Designing High-Bits ΔΣ Oscillator with Reduced Hardware Resource using Segmentation

Wenkang Zhou

(B.Eng. 2020)

Department of Electrical Engineering

McGill University, Montreal

November 2023

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

Abstract

This thesis explores the integration of segmentation in ΔΣ oscillators to enhance performance and reduce hardware costs. ΔΣ modulation has been a vital component of mixed-signal systems, delivering high-resolution analog-to-digital and digital-to-analog conversion. However, the need for cost-effective oscillator designs has prompted a reevaluation of traditional approaches. Therefore, the technique of segmentation has been proposed.

The thesis commences with the analysis and designs of ΔΣ modulators and oscillators. The mathematical foundations were first built for the design process, followed by simulations performed in MATLAB and Simulink to back them up. Then segmentation is introduced to the traditional ΔΣ modulators and oscillators, followed by simulations in Simulink to reach the same signal-to-noise (SNR) specification as the single paths. The core of the thesis lies in FPGA implementation. By implementing single-path and segmented ΔΣ modulators and oscillators within an FPGA, the research provides strong evidence of the benefits of segmentation, particularly in hardware cost reduction.

In summary, this thesis advances ΔΣ oscillator's practical implementation, with a specific focus on segmentation. The technique of segmentation contributes to the evolution of mixed-signal systems, enabling creative and cost-effective solutions for signal processing.

Résumé

Cette thèse explore l'intégration de la segmentation dans les oscillateurs ΔΣ pour améliorer les performances et réduire les coûts matériels. La modulation ΔΣ a été un élément essentiel des systèmes mixtes, permettant une conversion analogique-numérique et numérique-analogique de haute résolution. Cependant, la nécessité de concevoir des oscillateurs économiques a incité à réévaluer les approches traditionnelles. Par conséquent, la technique de segmentation a été proposée.

La thèse commence par l'analyse et la conception de modulateurs et d'oscillateurs ΔΣ. Les fondements mathématiques ont d'abord été posés pour le processus de conception, suivi de simulations réalisées dans MATLAB et Simulink pour étayer la théorie. Ensuite, la segmentation est introduite dans les modulateurs et les oscillateurs ΔΣ traditionnels, suivie de simulations dans Simulink visant à atteindre les mêmes spécifications de rapport signal/bruit (SNR) que les voies uniques. Le cœur de la thèse réside dans la mise en œuvre sur FPGA. En mettant en œuvre des modulateurs et des oscillateurs ΔΣ à voie unique et segmentés au sein d'un FPGA, la recherche apporte des preuves solides des avantages de la segmentation, en particulier en ce qui concerne la réduction des coûts matériels.

En résumé, cette thèse fait progresser de manière significative la mise en œuvre pratique des oscillateurs ΔΣ, en mettant particulièrement l'accent sur la technique innovante de la segmentation. L'adoption de la segmentation apporte des contributions importantes à l'évolution des systèmes mixtes, ouvrant la voie au développement de solutions de traitement du signal créatives et économiques.

Acknowledgments

First and foremost, I am deeply thankful to my supervisor, Professor Gordon Roberts, whose expertise and mentorship were instrumental in shaping this research. His patient guidance, encouragement, and unwavering support have been invaluable.

I would also like to acknowledge the invaluable contributions of Denis Romanov, a senior at McGill University. With his previous contribution to the research, I could catch up on with the concepts quicker and his insight really inspired me.

My heartfelt thanks extend to my colleagues and friends who provided me with a network of support and stimulating discussions. I would like to thank Muhammad Bilal Babar, Shaun Bradley, Jean-Christophe Couture and other staff in microelectronics laboratory. Their company gave me the courage to carry on with my study.

To my family, particularly my mom, Yan Zhou, who would support me financially and emotionally no matter where and when. I owe a debt of gratitude for her dedicated encouragement, love, and patience.

Finally, I am thankful for Mcgill University, which gives me the opportunity to conduct my research study these years. It is an amazing journey indeed.

Table of Contents

Abstract ..... **I**

Résumé ..... **II**

Acknowledgments ..... **III**

Table of Contents ..... **IV**

List of Tables ..... **VII**

List of Figures ..... **VIII**

List of Acronyms ..... **XI**

Chapter 1: Introduction **1**

1.1 Mixed-Signal Systems 1

1.1.1 Definition and Applications 1

1.1.2 Mixed-Signal Testing 2

1.2 Analog Sinewave Generation 3

1.2.1 Discrete Analog Oscillators 3

1.2.2 Digital Signal Generation 5

1.3 ΔΣ Modulation 6

1.3.1 History and Benefits of ΔΣ Modulation 7

1.3.2 ΔΣ Oscillators 8

1.3.3 Principles and Benefits of Segmentation 9

1.4 Thesis Objective 11

Chapter 2: Literature Review **13**

2.1 Principles of D/A conversion 13

2.2 DAC Performance Metrics 15

2.2.1 Gain and Offset Error 16

2.2.2 Monotonicity 18

2.2.3 Differential Non-Linearity (DNL) 18

2.2.4 Integral Non-Linearity (INL) 19

2.3 DAC Architectures 19

2.3.1 String DACs and Thermometer DACs 20

2.3.2 Binary-Weighted DACs 22

2.3.3 Pulse-Width Modulation DACs 22
2.3.4 ΔΣ DACs 24
2.3.5 Segmented DACs 25
2.4 Applications of Oscillators 27
2.5 Summary 29
Chapter 3: ΔΣ Modulators 30
3.1 Theory of ΔΣ Modulation 30
3.1.1 Structures of One-Bit ΔΣ modulators 30
3.1.2 Performance Characteristics of ΔΣ modulation 34
3.2 Design of an N-bit ΔΣ Modulator 35
3.2.1 Feasibility Study 36
3.2.2 Noise Transfer Function Structure 37
3.2.3 Transfer Function Design Method 38
3.3 One-Bit ΔΣ Modulator Simulation 40
3.4 Summary 45
Chapter 4: ΔΣ Oscillators 46
4.1 Theory and Design of ΔΣ oscillators 46
4.1.1 Digital Oscillator Analysis 46
4.1.2 ΔΣ Oscillator Structure 51
4.1.3 Noise Constraints 53
4.2 ΔΣ Oscillator Simulation 54
4.3 Summary 57
Chapter 5: Segmented ΔΣ Modulators 58
5.1 Segmentation introduction background 58
5.2 Design of Two-Segmented ΔΣ Modulators 59
5.2.1 Two-Segment ΔΣ Modulators Structure 59
5.2.2 Noise Constraints 62
5.2.3 Data Partition Impact 64
5.3 Two-Segment ΔΣ Modulator Simulation 66
5.4 Summary 69
Chapter 6: Segmented ΔΣ Oscillators 70

6.1 Segmented ΔΣ Oscillator Design 70
6.1.1 Segmented ΔΣ Oscillator Structure 70
6.1.2 Noise Constraints 72
6.2 Segmented ΔΣ Oscillator Simulation 73
6.3 Summary 78
Chapter 7: Experimental Validation 79
7.1 FPGA setup 79
7.1.1 Device Setup 79
7.1.2 Number System Setup 80
7.1.3 Arithmetic Operation 84
7.2 ΔΣ Modulator implementation 87
7.3 ΔΣ Oscillator Implementation 90
7.3.1 Single Path ΔΣ Oscillator 90
7.3.2 ΔΣ Oscillator with Segmentation 93
7.4 Hardware Costs 96
7.5 Summary 97
Chapter 8: Conclusion 98
8.1 Discussion of Results 98
8.2 Future Direction 99
Appendix 100
Verilog Source Code 100
Bibliography 108

List of Tables

Table. 3.1: The parameters for an N-bit unsigned ΔΣ modulator simulation example 41

Table. 4.1: Amplitude versus initial value of registers under different conditions 50
Table. 4.2: Amplitude versus initial value of registers under different conditions 55

Table. 5.1: Parameter settings for the two-segment modulator 66
Table. 5.2: SNR results of unsegmented ΔΣ modulator and two-segment ΔΣ modulator 69

Table. 6.1: Parameters of two-segment ΔΣ oscillator under 16-bits number system.. 74
Table. 6.2: Parameters of two-segment ΔΣ oscillator under 32-bits number system.. 76
Table. 6.3: System SNR and oscillation frequency under different circumstances 78

Table. 7.1: Theory, Simulink and FPGA results of SNR of the two-segment ΔΣ modulator..... 88
Table. 7.2: Theory, Simulink and FPGA results of oscillation frequency, amplitude and system SNR of the one-path ΔΣ Oscillator..... 92
Table. 7.3: Theory, Simulink and FPGA results of oscillation frequency, amplitude and system SNR of the one-path ΔΣ oscillator 95
Table. 7.4: Hardware costs for single path and two-segment ΔΣ modulator..... 96
Table. 7.5: Hardware costs for single path and two-segment ΔΣ oscillator 96

List of Figures

Fig. 1.1: The example of test set-up of a mixed signal device 2
Fig. 1.2: Configuration of the Colpitts oscillator 4
Fig. 1.3: Block diagram of a general direct digital frequency synthesis 5
Fig. 1.4: The structure of a lossless discrete resonator 6
Fig. 1.5: A comparison of the distance between the frequency band in green and the imaging 8
Fig. 1.6: The structure of a ΔΣ oscillator 9
Fig. 1.7: The structure of a K-segment DAC 11

Fig. 2.1: Diagram of an N-bit DAC 13
Fig. 2.2: Transfer curve for a 4-bit DAC 15
Fig. 2.3: Transfer curve of 4-bit DACs with gain error and offset error 16
Fig. 2.4: Example of best-fit line for a 4-bit DAC 17
Fig .2.5: Example of Endpoint-to-endpoint line for a 4-bit DAC 17
Fig. 2.6: Example of a 3-bit string DAC 20
Fig. 2.7: Example of a 3-bit Thermometer DAC 21
Fig. 2.8: Example of an 8-bit Binary-Weighted DAC 22
Fig. 2.9: Structure of a PWM DAC 23
Fig. 2.10: Structure of a ΔΣ DAC 24
Fig. 2.11: Noise distribution across frequency (left: a typical DAC; right: a ΔΣ DAC) 25
Fig. 2.12: An example of segmented DAC architecture 26
Fig. 2.13: Some applications of oscillators in DACs 28

Fig. 3.1: Block diagram of the general 1-bit delta-sigma modulator 31
Fig. 3.2: Linear model of the general 1-bit delta-sigma modulator 31
Fig. 3.3: Modified signal flow graph of the general delta-sigma modulator 32
Fig. 3.4: Structure of the general delta-sigma modulator with a unity STF 33
Fig. 3.5: Structure of 1-bit ΔΣ DAC 34
Fig. 3.6: Structure of DF-II for H(z) implementation 40
Fig. 3.7: The structure of a 1-bit unsigned ΔΣ modulator 40
Fig. 3.8: Magnitude response of the noise transfer function 42
Fig. 3.9: PSD of the outputs for sinusoidal inputs given the 3rd order inverse Chebyshev loop filter (without windowing) (a): Nyquist frequency (b): zoom in for in-band frequency 43
Fig. 3.10: PSD of the outputs for sinusoidal inputs given the 3rd order inverse

Chebyshev loop filter with Dolph–Chebyshev window (a): Nyquist frequency (b): zoom in for in-band frequency 44

Fig. 4.1: Block diagram of the lossless discrete resonator (LDR) 46
Fig. 4.2: Relationship between the oscillation frequency and the coefficients a1*a2 48
Fig. 4.3: Relationship between the amplitude and the initial value of the registers. Graph (a) illustrates the amplitude as a function of the register 1 given the other is 0. Graph (b) illustrates the amplitude as a function of the register 1 given the other is 1. Graph (c) and (d) illustrate the opposite situation of graph (a) and (b). 51
Fig. 4.4: The general structure of a ΔΣ oscillator 51
Fig. 4.5: The configuration of the ΔΣ oscillator (a): A general ΔΣ oscillator structure 52
Fig. 4.6: PSD of the outputs of ΔΣ Oscillator. (a): whole frequency; (b): in-band frequency 56

Fig. 5.1: Partition of the input signal 59
Fig. 5.2: Realization of partition of the input signal 60
Fig. 5.3 Overall structure of two-segment ΔΣ modulator 61
Fig. 5.4: An example of data partition: (a) full-scale input; (b) coarse path input; (c) fine path input 65
Fig. 5.5: Magnitude response of the fine path noise transfer function 67
Fig. 5.6: Theoretical and Simulink simulation output PSD of two-segment ΔΣ modulator 68

Fig. 6.1: General block diagram of a segmented ΔΣ oscillator 71
Fig. 6.2: Block diagram of a two-segment ΔΣ modulator 71
Fig. 6.3: Block diagram of two-segment ΔΣ modulator 73
Fig. 6.4: Theoretical and Simulink simulation output PSD of two-segment ΔΣ oscillator under 16 bits number system (a): Nyquist frequency (b): in-band frequency 75
Fig. 6.5: Theoretical and Simulink simulation output PSD of two-segment ΔΣ oscillator under 32 bits system (a): Nyquist frequency (b): in-band frequency 77

Fig. 7.1: Ideal (Theory, Simulink) and 4-bits fractional part realization (Theory, Simulink, FPGA implementation) of output PSD for a single path ΔΣ modulator 82
Fig. 7.2: Ideal (Theory, Simulink) and 16-bits fractional part realization (Theory, Simulink, FPGA implementation) of output PSD for a single path ΔΣ modulator 83
Fig. 7.3: Number representation in FPGA format 83
Fig. 7.4: Arithmetic operation structure 84
Fig. 7.5: Schematic view of the 1-bit modulator provided by QuartusII 87
Fig. 7.6: Schematic view of the two-segment ΔΣ modulator provided by QuartusII . 88
Fig. 7.7: Output PSD of Simulink simulation and FPGA implementation of two-segment ΔΣ modulator (a): Nyquist frequency (b): in-band frequency 89
Fig. 7.8: Schematic view of the single path ΔΣ oscillator provided by QuartusII 90
Fig. 7.9: Output PSD of Simulink simulation and FPGA implementation of the single

path ΔΣ oscillator (a): Nyquist frequency (b): in-band frequency..... 91
Fig. 7.10: Simulink and FPGA sinewave output of the single path ΔΣ oscillator..... 92
Fig. 7.11: Schematic view of the two-segment ΔΣ oscillator provided by QuartusII. 93
Fig. 7.12: Output PSD of Simulink simulation and FPGA implementation of the two-segment ΔΣ oscillator (a): Nyquist frequency (b): early frequency..... 94
Fig. 7.13: Simulink and FPGA sinewave output of the two-segment ΔΣ oscillator... 95

List of Acronyms

ADC	Analog-to-Digital Converter
ADM	Adaptive Delta Modulation
CAD	Computer Aided Design
DAC	Digital-to-Analog Converter
DDFS	Direct Digital Frequency Synthesis
DNS	Differential Non-Linearity
DSP	Digital Signal Processing
INL	Integral Non-Linearity
LDR	Lossless Discrete Resonator
LPF	Low-Pass Filter
LSB	Least Significant Bit
TLE	Total Logical Elements
BIST	Built-In-Self-Test
MSB	Most Significant Bit
NTF	Noise Transfer Function
PSD	Power Spectral Density
PWM	Pulse-Width Modulator
SNR	Signal-to-Noise Ratio
STF	Signal Transfer Function

Chapter 1: Introduction

In this chapter, the main concepts in relation to this thesis are introduced, including mixed signal testing, analog sinewave generation, ΔΣ modulation and the technique of segmentation. Additionally, the motivation behind this thesis is explored and elaborated upon.

1.1 Mixed-Signal Systems

1.1.1 Definition and Applications

In the field of electronics, one can classify circuits and signals as digital or analog in general. A mixed-signal circuit is a circuit that has both analog and digital components [1]. Mixed-signal integrated circuits (IC) are often used to convert analog signals to digital signals so that digital devices can process them or vice versa. Such devices are defined as the analog-to-digital converters (ADCs) or digital-to-analog converters (DACs).

The usage of mixed-signal circuits has experienced a significant increase due to the widespread use of smartphones, telecommunications, and vehicles equipped with digital sensors. Mixed-signal circuits or systems are commonly used as cost-efficient solutions in various domains, including industrial, medical, measurement, and space-related applications.

Due to the utilization of both digital signal processing and analog circuitry, mixed-signal ICs are generally designed for a specific purpose. The design demands significant proficiency and careful use of computer aided design (CAD) tools. Furthermore, specific design tools, such as mixed-signal simulators, and

description simulation languages, like VHDL-AMS, are employed during the design process. The testing process for the final chips can also be complex and challenging, as we will explore in the next subsection.

1.1.2 Mixed-Signal Testing

Nowadays, the testing techniques and procedures for digital chips have become well-established. However, the same cannot be said for mixed-signal ICs [2]. The cost associated with high-volume production of mixed-signal ICs is significantly impacted by their testing costs. The test cost associated with testing the analog portion of a mixed-signal device can cost up to 50% of the total product cost. With new technologies, the cost of analog testing in mixed-signal devices may become the predominant factor.

Fig. 1.1 depicts a typical arrangement used to perform analog measurements. The measurement process involves a sinusoidal signal generator with variable amplitude and frequency control, which excites the analog portion of the mixed-signal circuit. A true-RMS power meter operating over a very narrow, but tunable, frequency band can be used to extract the circuit response.

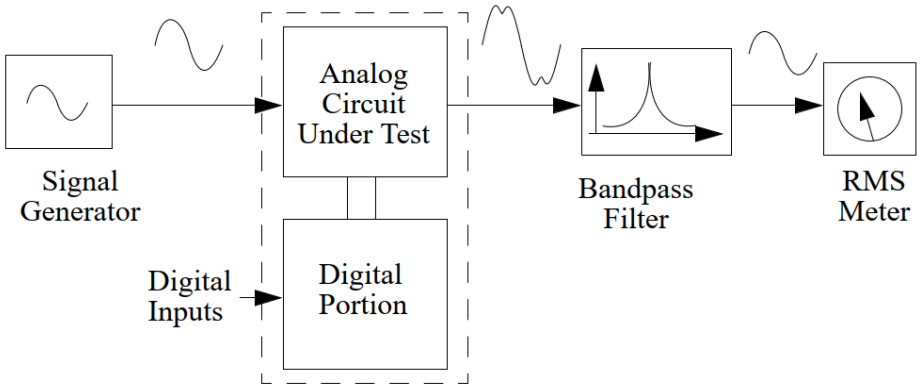


Fig. 1.1: The example of test set-up of a mixed signal device

To decrease testing costs, IC designers have put in considerable effort to produce circuits that can self-test, known as Built-In-Self-Test (BIST) [3]. BIST can significantly simplify the testing of digital ICs. However, the test challenge with BIST for analog circuits is far more complicated. Mixed signal BIST requires a high-precision analog signal source that needs to be fabricated on the same IC as the circuit-under-test. For practical purposes, realizing a high-precision analog test source on an IC without external calibration or trimming remains a challenge to this day.

In this thesis, a high-precision analog signal source for BIST application is to be investigated. This source is to exceed present day performance levels and silicon area requirements through the application of hardware resource reduction using segmentation.

1.2 Analog Sinewave Generation

In this section, different methods of analog signal generation are discussed with an emphasis on IC fabrication.

1.2.1 Discrete Analog Oscillators

Tuned oscillator circuits are traditional methods to produce oscillation using discrete components. They are also known as LC oscillators or resonant circuit oscillators [4].

According to the Barkhausen stability criteria [4], a circuit will sustain steady-state oscillations only when the loop gain $A\beta$ meets the following conditions:

$$| A\beta | = 1 \tag{1.1}$$

and

$$\angle A\beta = 0 \pm 2\pi k; k \; is \; an \; integer \tag{1.2}$$

Here we take the Colpitts Oscillator as an example. The configuration is shown in Fig. 1.2.

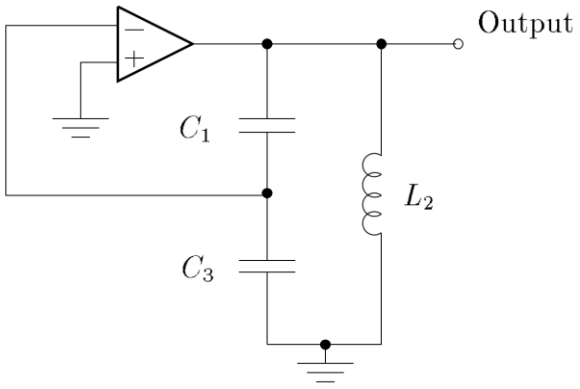


Fig. 1.2: Configuration of the Colpitts oscillator

The oscillation frequency of this structure is given as

$$\omega_o = \sqrt{\frac{1}{L_2}\left(\frac{1}{C_1} + \frac{1}{C_3}\right)} \tag{1.3}$$

However, purely analog oscillators are not suitable for mixed-signal BIST, as argued in [5]. The reasons are listed as followed:

- Integrated inductors needed in passive implementations have poor quality factors and are prohibitively large.
- Active implementations of analog oscillators using opamps are vulnerable to variations in the fabrication process and temperature drift.
- Integrated precision analog components require trimming, which adds to the fabrication cost.

In comparison, the operation of digital circuits is much less affected by these factors. Therefore, digital circuits with digital-to-analog converter are the preferred approach to realize an oscillating source in a mixed signal BIST.

1.2.2 Digital Signal Generation

Advancements in Digital-to-Analog and Analog-to-Digital converter technology have enabled IC designers to utilize Digital Signal Processing (DSP) techniques for processing analog signals. Here we will discuss three methods of digital signal generation: Direct digital frequency synthesis (DDFS), Lossless discrete resonator (LDR) and ΔΣ oscillator.

Direct digital frequency synthesis (DDFS) is a method generating analog sinusoidal signals using digital hardware and a DAC. As discussed in [4], the principle behind DDFS is to sequentially access the addresses of a ROM containing the sample points of a sinewave. The block diagram of DDFS is shown in Fig. 1.3. The phase accumulator enables the user to scan the ROM with different increments, denoted as F, to change the frequency. A multi-bit DAC and a low-pass filter (LPF) then transform the digital words from the ROM into an analog signal. The precision of the synthesizer will be dictated by this DAC.

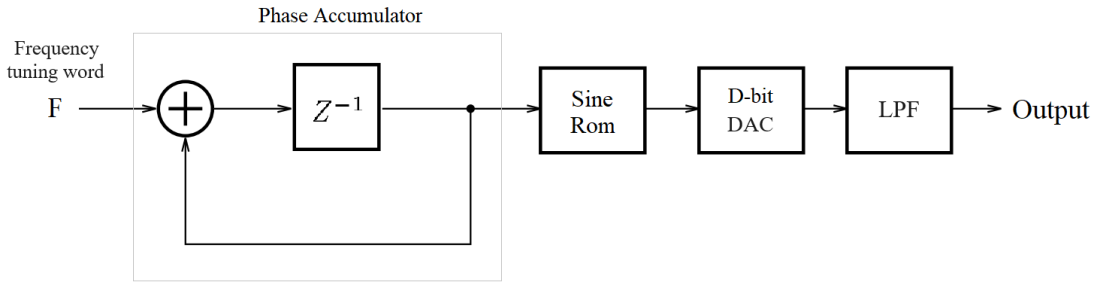


Fig. 1.3: Block diagram of a general direct digital frequency synthesis

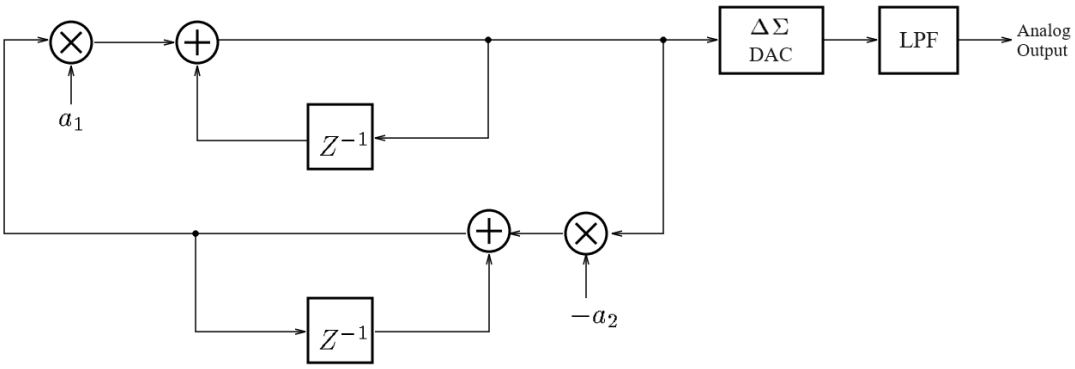


Fig. 1.4: The structure of a lossless discrete resonator

However, the multi-bit DAC and the ROM could take too much area resources as described in [4]. A more area-efficient method to generate a digital sinusoidal signal is the LDR approach. The passive LC ladder filters have exceptional sensitivity properties [6] and the LDR is based on this structure. Additionally, the multi-bit DAC can be replaced with a ΔΣ DAC to save more space. The general structure of the LDR approach is shown in Fig. 1.4.

To further reduce area usage, a ΔΣ oscillator is proposed [5]. The ΔΣ oscillator is the core of this thesis and the history and benefits will be discussed in detail in the next section.

1.3 ΔΣ Modulation

In addition to compatibility with VLSI technology, ΔΣ converters provide a high level of reliability and functionality and reduced chip cost. In this thesis, ΔΣ modulators are utilized for partial DAC realization. This section will explain the history and benefits of ΔΣ modulation. Moreover, a brief description of segmentation is introduced. The details of the performance characteristics and topologies of ΔΣ Modulation will be discussed in Chapter 3.

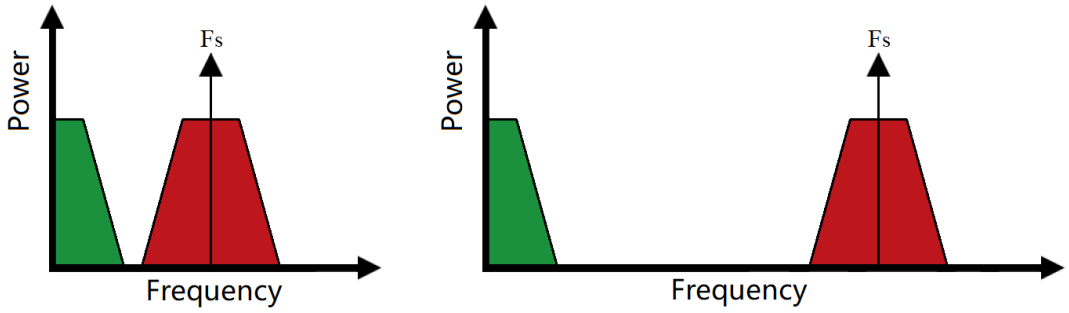
1.3.1 History and Benefits of ΔΣ Modulation

Delta-Sigma (ΔΣ) modulation is a technique for converting analog or digital signals into digital data that finds wide applications in all electronic systems. ΔΣ modulation provides high-resolution digital representations of analog signals, which makes it suitable for applications where fine details and precision are required. It represents an enhancement over the Delta Modulation (DM) [7]. DM is an analog-to-digital modulation technique widely employed in electronic systems, including digital telephony over the Internet (VoIP), digital wireless communication, and mobile communications. DM suffers from two significant drawbacks, namely slope overload distortion and granular noise [8]. To overcome these disadvantages, two approaches are commonly employed, namely the adaptive delta modulation (ADM) [9] and the ΔΣ modulation [7].

Utilizing ΔΣ DACs offers a significant advantage because they are predominantly digital, requiring only an LPF as the sole analog component. This attribute contributes to their compactness when implemented on a silicon chip.

In addition, ΔΣ modulation exhibits highly favorable characteristics in terms of noise-shaping. This is achieved by employing a clock with a sampling frequency that is at least twice the maximum signal frequency. An example using different sampling frequency is depicted in Fig. 1.5. As can be seen, the imaging frequencies are shifted away from the signal band using high sampling frequency, which minimizes interference caused by aliasing in the input spectrum and allows for a simpler LPF with less need for a sharp roll-off.

Moreover, ΔΣ modulators typically consume less power than other high-resolution analog-to-digital techniques, making them suitable for battery-powered or low-power applications.



(a) (b)

Fig. 1.5: A comparison of the distance between the frequency band in green and the imaging frequencies in red for (a) a low sampling frequency (b) a high sampling frequency

1.3.2 ΔΣ Oscillators

Different from other digital signal generation methods, ΔΣ oscillators stand at the crossroads of precision signal generation and advanced modulation techniques. Rooted in the principles of ΔΣ modulation, ΔΣ oscillators offer an innovative approach to achieving high-precision analog sinusoidal signals.

The history of ΔΣ oscillators is closely linked with the evolution of sigma-delta modulation. While ΔΣ modulation was first introduced for use in ADC, engineers recognized the potential to adapt these techniques for signal generation over time, paving the way for the development of ΔΣ oscillators [7]. These concepts therefore have reshaped the field of oscillator design and signal processing. ΔΣ oscillators can be seen as an improvement of the LDR approach as one multiplier is replaced by a two-input multiplexer, which further reduces area usage. The structure of a ΔΣ oscillator is shown in Fig. 1.6. Here, the only analog components are the 1-bit DAC and the low-pass filter (LPF), as all other components are digital.

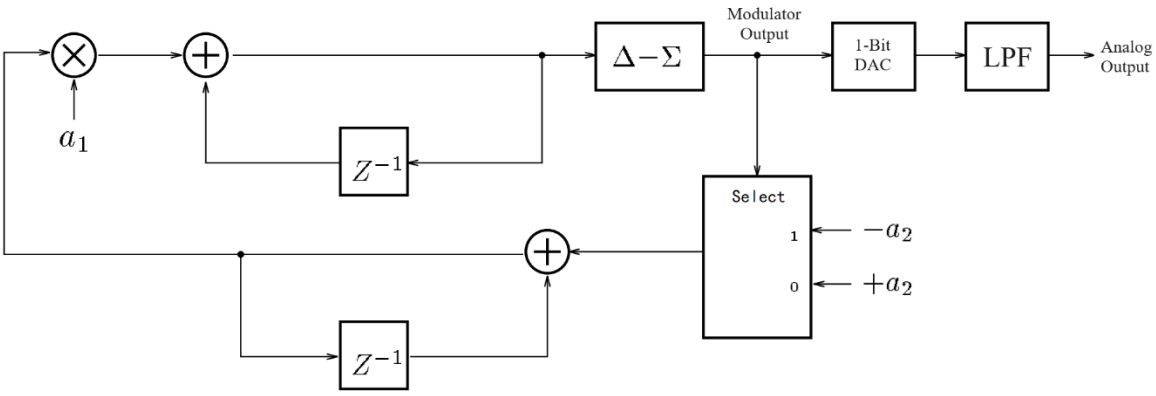


Fig. 1.6: The structure of a ΔΣ oscillator

ΔΣ oscillators also have many advantages. Because of the ΔΣ modulator inside the structure, they can deliver high-resolution output and inherent linearity, thus eliminating the need for complex calibration procedures. As a result, ΔΣ oscillators are used in applications where minimizing noise and achieving high signal-to-noise ratio (SNR) is vital [10].

The theories and topologies of ΔΣ oscillators will be covered in Chapter 4.

1.3.3 Principles and Benefits of Segmentation

When we are required to design a DAC to meet the specific performance requirement, it may well be that no single architecture is ideal. In such cases, two or more DACs may be combined to form a single DAC to achieve the required performance. In principle, the process of segmentation in DACs typically involves the partitioning of the input digital data into *K* segments, and each segment is processed individually, then all combined into a single output signal. The first path handles the *N1* most significant bits (MSB), the next path handles the *N2* MSBs, etc. The pattern continues for all *K* segments. The outputs of the individuals DACs are then combined to reconstruct the final converted signal.

The structure of a K-segment DAC is shown in Fig. 1.7.

Segmentation produces important benefits. First, segmentation can enhance resolution and reduce mismatch errors. By dividing the modulator into K segments, each segment will deal with a smaller range of input signals, thus reducing the weight mismatch between the MSBs and LSBs. This results in an improvement in resolution and accuracy, making segmentation a useful strategy for achieving high-resolution ΔΣ modulators.

The second advantage segmentation brings is that it also optimizes the power efficiency and area usage in ΔΣ modulators. Segmentation divides the processing task among multiple smaller units or segments, each working on a narrower input range. This approach can significantly reduce the power consumption and silicon area required for the modulator, making it more practical for integrated circuit design [11]. This thesis focuses on the realization of a ΔΣ oscillator with a two-segment ΔΣ modulator. The segmentation utilization will be covered in Chapter 5 for segmented ΔΣ modulators and Chapter 6 for segmented ΔΣ oscillators.

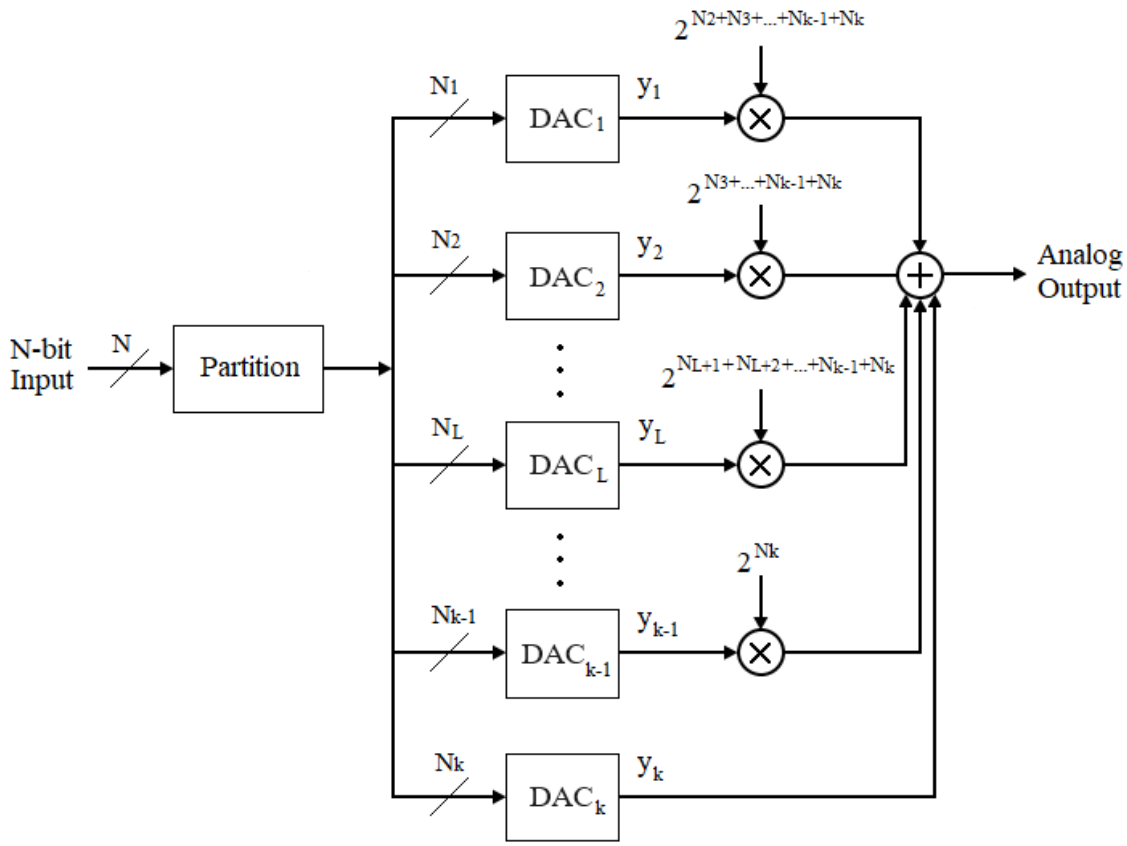


Fig. 1.7: The structure of a K-segment DAC

1.4 Thesis Objective

This thesis focuses on the development of a high-precision analog signal source using segmentation. The analog signal source shall follow the constraints below:

- The generation of high-precision test signals that meets a desired signal-to-noise ratio (SNR).
- Programmability should be supported by the source.
- External calibration or trimming is to be avoided, necessitating the source to be insensitive to process variations.
- The circuit should occupy less silicon area and be mostly digital.
- Segmentation shall be introduced to reduce hardware costs.

To achieve a high SNR for ΔΣ modulation, there is a significant trade-off in terms of increased hardware complexity and area costs. A higher order modulator will require far more hardware when compared to one with the same data path width but a lower loop filter order. Similarly, expanding the width of the data path also leads to increased hardware requirements. Therefore, by introducing segmentation into ΔΣ modulation, less silicon area is required. Notably, Ahmed Emara's previous research [12] introduced segmentation into ΔΣ modulation and realized segmented ΔΣ modulators through IC implementation. Moreover, Denis Romanov [11] demonstrated the mapping of segmented modulators into a digital realization using FPGA.

In the context of generating high-precision analog signals, Albert Lu's work [13] has also made substantial contributions by developing the concept of ΔΣ oscillators. With the benefits of segmentation in mind, this thesis aims to investigate the incorporation of segmentation into ΔΣ oscillators using digital realization via FPGA.

Chapter 2: Literature Review

As has been stated in the previous chapter, the Digital-to-Analog Converter (DAC) is a vital component of analog signal generation. Within this chapter, fundamental principles and characteristics of DACs are elucidated, alongside an exploration of the various DAC architectures documented in existing literature. Finally, the use of segmentation and its application is presented.

2.1 Principles of D/A conversion

This section will provide background information of DACs, including the principle of D/A conversion and the key metrics used to evaluate their performance.

The Digital-to-Analog Converter (DAC) is a system that converts a digital signal into an analog signal. A DAC can reconstruct the original signal from the sampled data provided that its bandwidth meets the requirements of the Nyquist sampling theorem [14].

Fig. 2.1 shows the structure of a typical N-bit DAC. It can be characterized as a decoding device that takes an integer input value and generates an output in the form of an analog quantity such as voltage or current.

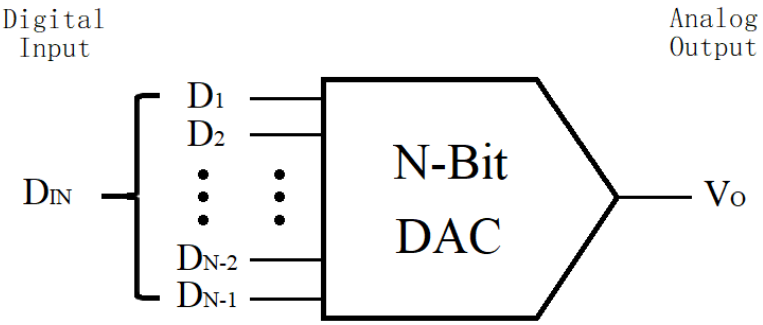


Fig. 2.1: Diagram of an N-bit DAC

The operation of the DAC can be expressed as an approximation of multiplying by a factor G to convert between analog and digital equivalents. Assume the input is D_{IN} and the output is V_o, we can write:

$$V_o = G \cdot D_{IN} \tag{2.1}$$

where the factor G is a real-valued constant. Since D_{IN} corresponds to data originating from a digital system, it is commonly represented as an N-bit wide base-2 unsigned integer. Therefore, the value of D_{IN} ranges from 0 to $2^N - 1$ and its expression can be given as:

$$D_{IN} = D_0 + 2^1 \cdot D_1 + 2^2 \cdot D_2 + \cdots + 2^{N-1} \cdot D_{N-1} \tag{2.2}$$

where the coefficients D_0, D_1,…, D_{N-1} have either a 0 or 1 value. Coefficient D_0 is referred to as the least significant bit (LSB) because it exerts the least influence on D_{IN}, whereas D_{N-1} is regarded as the most significant bit (MSB) since it has the greatest impact.

From Eqn. (2.1), we can deduce the smallest change at the input as 1 LSB. Thus, the smallest voltage change at the output is given as:

$$V_{LSB} = G \cdot 1\ LSB \tag{2.3}$$

Fig. 2.2 demonstrates an example of transfer curve of a 4-bit DAC, where the factor G is set to 0.2 V. For each input digital word, we could see there is one-to-one mapping and the LSB step size $V_{LSB} = 0.2$ V.

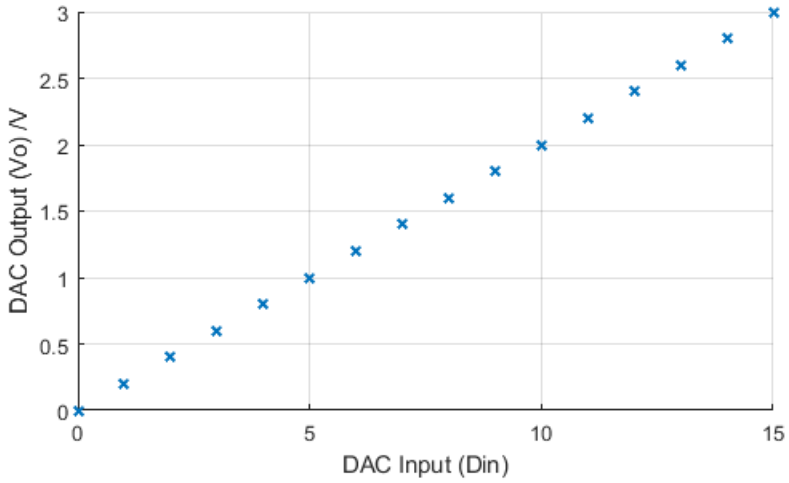


Fig. 2.2: Transfer curve for a 4-bit DAC

Alternatively, one can design the gain of the DAC (G) as the ratio of the range of output values to the range of input values as:

$$G = \frac{V_{FS+} - V_{FS-}}{D_{IN,MAX} - D_{IN,MIN}} \tag{2.4}$$

where V_{FS+} and V_{FS-} represents the maximum and minimum output voltage values, respectively. If we define the difference between V_{FS+} and V_{FS-} as V_{FSR} and denote the input integer range as $2^N - 1$. Thus, the DAC gain and the LSB step size are given as follows

$$G = \frac{V_{FSR}}{2^N - 1} \tag{2.5}$$

and

$$V_{LSB} = \frac{V_{FSR}}{2^N - 1} \tag{2.6}$$

2.2 DAC Performance Metrics

Within the realm of DACs, various metrics are utilized to assess their operation. In this section, we introduce DAC errors and elaborate on three

linearity metrics: Monotonicity, Differential Non-Linearity (DNL), and Integral Non-Linearity (INL). These characteristics are previously discussed in [15].

2.2.1 Gain and Offset Error

The transfer curve for the ideal 4-bit DAC provided earlier, as shown in Fig. 1.7, assumed that the DAC is ideal, which means gain error and offset error do not exist. We denote the ideal gain as G_{ideal}. In practical terms, however, the gain G_{actual} is not necessarily equal to the ideal gain and thus introduces a gain error. Gain error percentage can be expressed as:

$$GairError\% = \frac{G_{actual} - G_{ideal}}{G_{ideal}} \times 100\% \tag{2.7}$$

Moreover, the value of V_{FS-} is not always equal to zero, leading to the presence of offset error. Fig. 2.3 represents the transfer curves of two DACs with a gain error and an offset error, respectively, when comparing to the ideal DAC.

To compensate for the gain and offset error, two reference lines were introduced, namely best-fit line and endpoint-to-endpoint line [16].

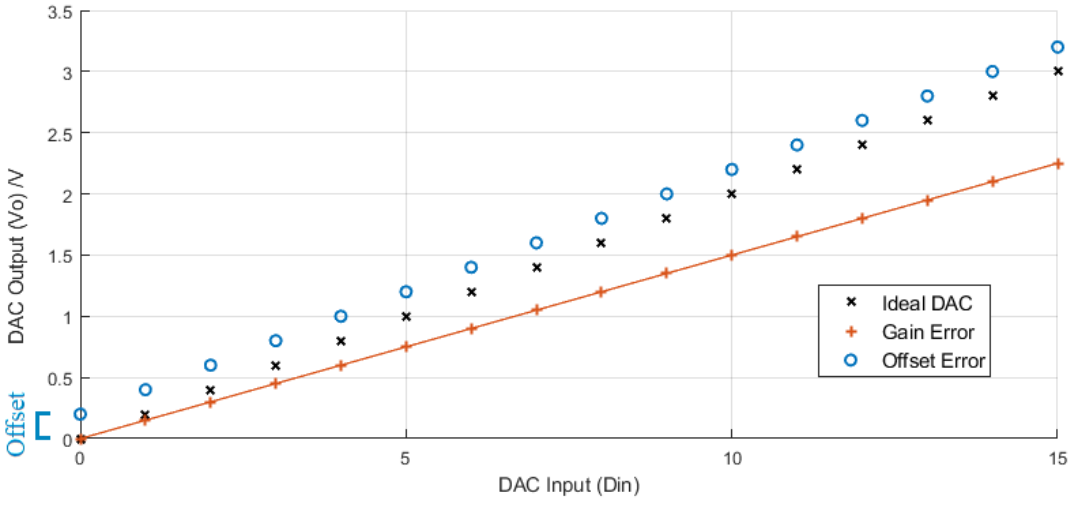


Fig. 2.3: Transfer curve of 4-bit DACs with gain error and offset error

The best-fit line is obtained by applying linear regression to the data points of a DAC transfer curve, ensuring the line closely matches the data. The endpoint-to-endpoint reference line is simply drawn by connecting the two endpoints. Figure 2.4 illustrates a best-fit line for a 4-bit DAC. On the other hand, the endpoint-to-endpoint reference line is simply drawn by connecting the two endpoints of the transfer curve, which is demonstrated in Fig 2.5. These two reference lines serve as a basis for calculating the linearity metrics DNL and INL.

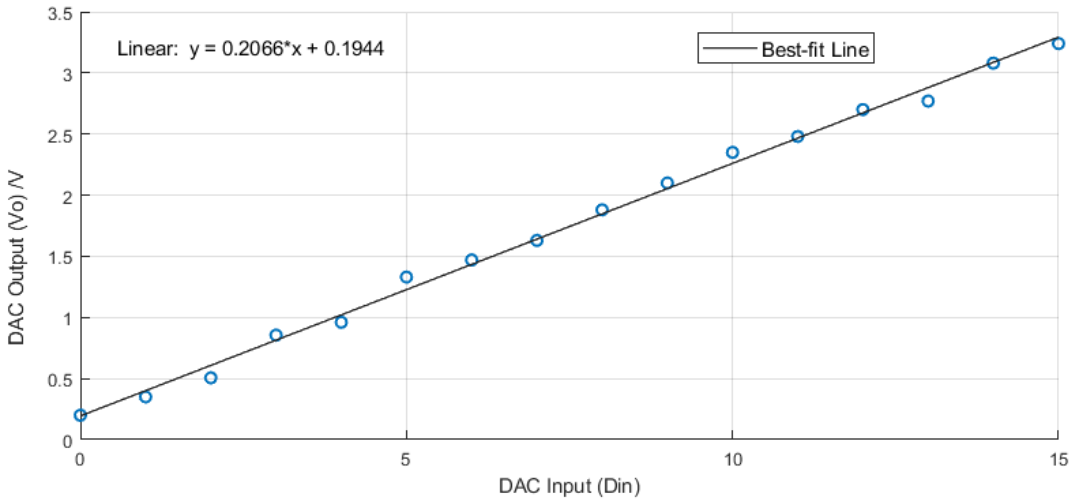


Fig. 2.4: Example of best-fit line for a 4-bit DAC

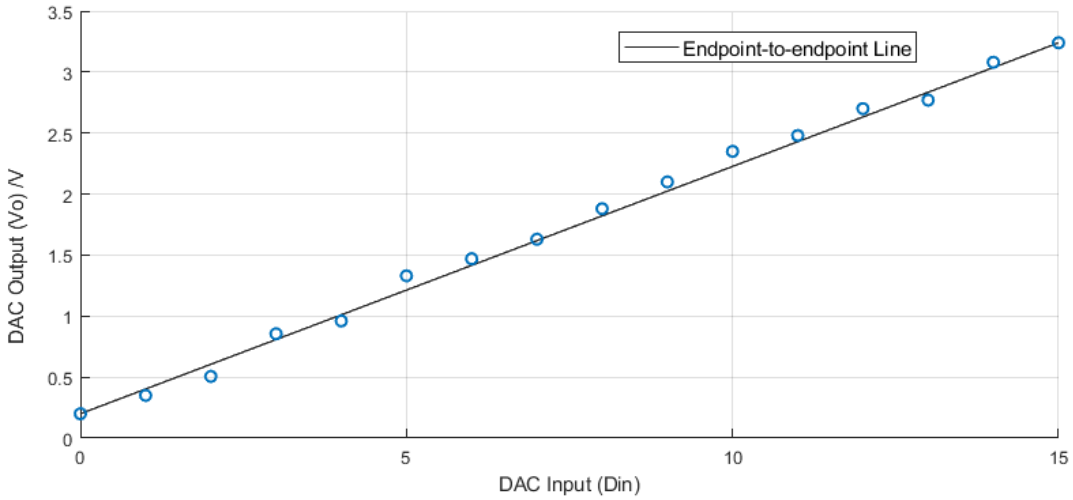


Fig .2.5: Example of Endpoint-to-endpoint line for a 4-bit DAC

2.2.2 Monotonicity

In an ideal DAC, the output voltage obtained from a specific code is always greater than the output voltage generated by the preceding input code. This property, known as monotonicity, can be assessed by calculating the discrete first derivative of the transfer curve. The output voltage for a given code is denoted as $S(i)$, while the output for the next code is denoted as $S(i+1)$. To test for monotonicity, we analyze the discrete first derivative of the transfer curve, denoted as $S'(i)$ in this context:

$$S'(i) = S(i+1) - S(i) \tag{2.8}$$

If all the derivatives are positive when the input is a rising ramp or negative when the input is a falling ramp, it indicates that the DAC is monotonic.

2.2.3 Differential Non-Linearity (DNL)

In an ideal DAC, V_{LSB} represents the difference between two adjacent output levels. DNL measures the deviation from the ideal step size between two adjacent output levels. It indicates the DAC's ability to accurately reproduce small changes in the input signal. The DNL curve illustrates the error in each step size, expressed as fractions of an LSB. DNL is computed in a similar manner to the discrete first derivative, using the following formula:

$$DNL(i) = \frac{S(i+1) - S(i) - V_{LSB}}{V_{LSB}} LSB \tag{2.9}$$

The determination of LSB calculations depends on the reference line employed, whether it is the ideal line, the best-fit line, or the endpoint-to-endpoint reference line [16]. Consequently, the V_{LSB} values derived from each reference line differ from one another, while despite the variations, the results

obtained from these different reference lines will be relatively similar.

2.2.4 Integral Non-Linearity (INL)

The INL curve, very similar to DNL, quantifies the deviation of the DAC's output from a reference line. It provides information about the overall linearity performance of the DAC. The INL curve is normalized to the LSB step size. The INL curve is calculated by subtracting the reference DAC line from the actual DAC curve then normalized to be expressed as a fraction of the average LSB step:

$$INL(i) = \frac{S(i) - S_{REF}(i)}{V_{LSB}} LSB \tag{2.10}$$

If the absolute maximum value of the INL is less than 1 LSB, it indicates that the DAC has achieved the desired resolution. The calculation of INL relies on the choice of reference line used [16]. It is also worth mentioning that the INL curve is the integral of the DNL curve. While DNL quantifies the error in step sizes between consecutive codes, INL represents the accumulated error across all the step sizes.

2.3 DAC Architectures

In this section, introduction of the commonly used DAC architectures is provided, followed by the advantages and disadvantages associated with each architecture. The information of these architectures was mainly taken from [17].

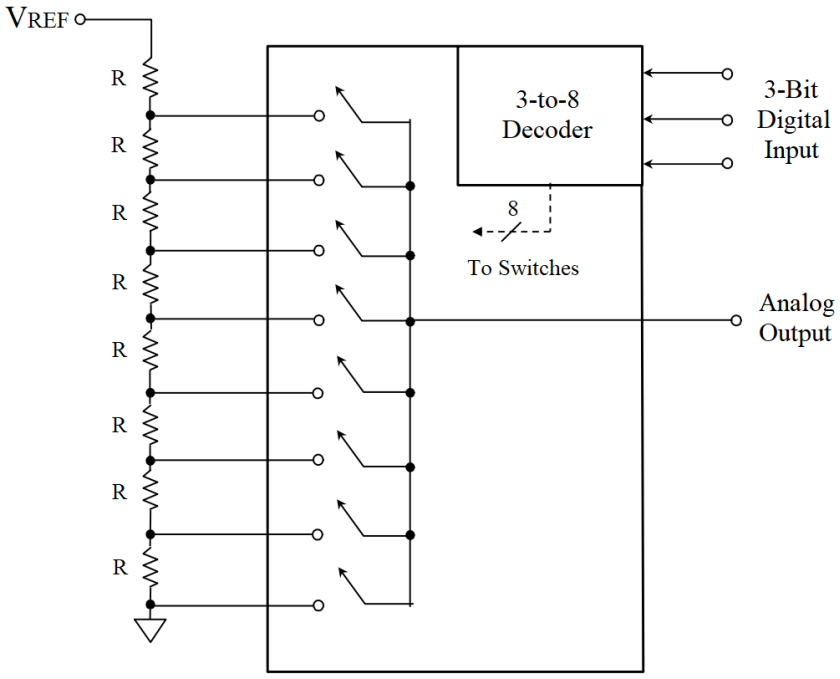


Fig. 2.6: Example of a 3-bit string DAC

2.3.1 String DACs and Thermometer DACs

The simplest DAC structure of all is the Kelvin divider or string DAC [17]. Fig. 2.6 presents the structure of a 3-bit string DAC. As can be seen, an N-bit string DAC consists of three parts, including a $N - to - 2^N$ decoder, 2^N switches (mostly CMOS) and 2^N equal resistors. The switches and resistors are allocated such that there is one for each node of the chain and output. The decoder takes the N bit input and converts it into 2^N separate signals of which only one is active. The output is taken from the appropriate tap by closing just one of the switches.

The architecture itself is quite simple, and the digital circuitry involved in the decoder is quite cheap to implement. In addition, the voltage output is inherently monotonic, as even a short-circuit in one of the resistors cannot cause any output code's voltage to exceed the output of the following code.

The major downside lies in the heavy use of analog components. For a linear increase in resolution, one must incur an exponential cost in components. Analog components such as resistors can take up a considerable amount of space and generate a non-negligible amount of heat if they become numerous.

Thermometer-coded DACs [17] use 2^{N-1} equal size number of elements (current sources, resistors, or capacitors). There is a current-output DAC analogous to a string DAC that consists of 2^{N-1} switchable current source (which may be resistors and a voltage reference or may be active current sources) connected to an output terminal, which must be at, or close to, ground. This architecture is commonly referred to as a "thermometer" or "fully decoded" DAC.

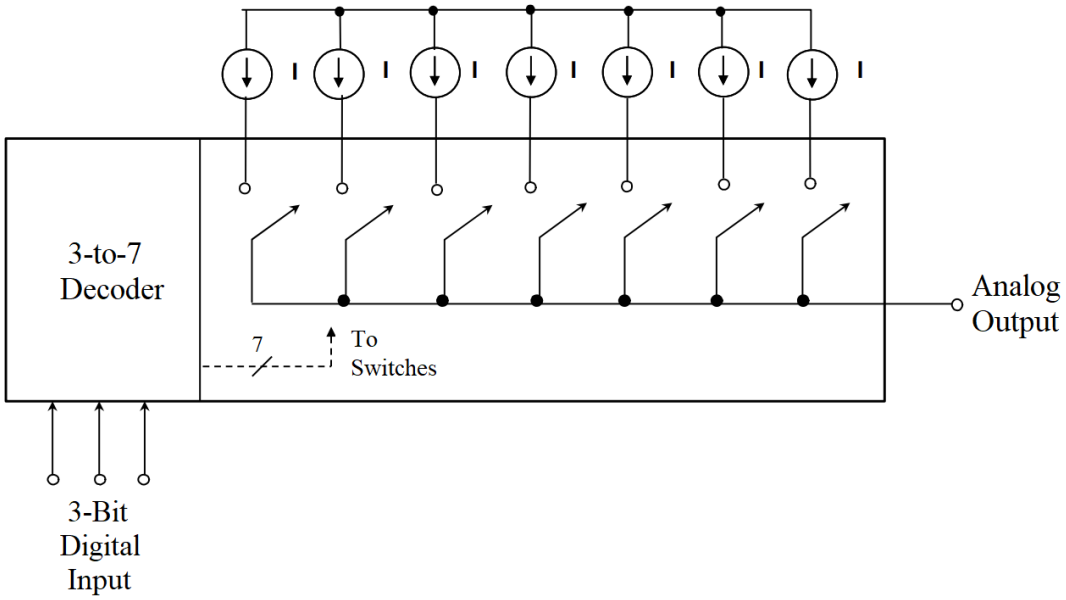


Fig. 2.7: Example of a 3-bit Thermometer DAC

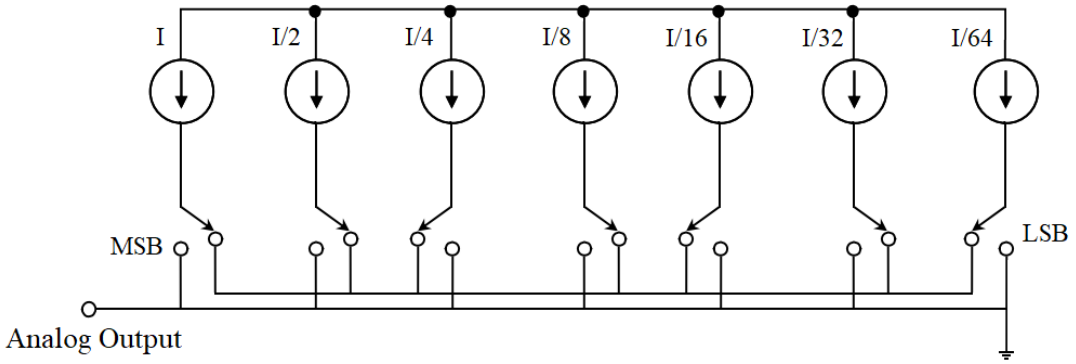


Fig. 2.8: Example of an 8-bit Binary-Weighted DAC

2.3.2 Binary-Weighted DACs

Binary-weighted DACs use binary-weighted number of elements [17], which are employed to generate the desired analog value by selectively switching these sources on or off based on the digital input code. Fig. 2.8 presents an 8-bit current mode binary-weighted DAC.

Compared to thermometer-coded architecture as shown in Fig. 2.7, binary-weighted DAC is significantly more area efficient as it has much higher resolution with the same number of current sources being used.

However, this architecture faces a notable drawback when aiming for resolutions of 10 bits or higher. The weights of the most significant bits (MSBs) and least significant bits (LSBs) exhibit a substantial difference, resulting in high sensitivity of the output to mismatch errors. To address this issue, various calibration circuits and techniques [17] are required for error correction.

2.3.3 Pulse-Width Modulation DACs

Fig. 2.9 illustrates the configuration of a pulse-width modulated (PWM)

DAC. The circuit operates based on a sampling clock, causing an N-bit counter to iterate through its complete range of possible values, in comparison to the digital input. When the input exceeds the counter's value, the output produces a high voltage; otherwise, it generates a low voltage. As a result, a rectangular wave is generated with 2N sampling periods and a duty cycle corresponding to the input. To obtain the desired analog output, this output bitstream needs to undergo reconstruction by passing it through a low-pass filter (LPF).

PWM DACs offer a notable advantage in the form of high-linearity achieved using a 1-bit DAC. Single-bit DACs exhibit perfect linearity and can be regarded as ideal DACs. Consequently, PWM DACs eliminate the need for calibration, making them highly suitable for testing purposes. Furthermore, PWM implementation predominantly relies on cost-effective digital components, which are efficient in terms of their area utilization.

However, a significant drawback arises from the analog LPF employed in PWM DACs, particularly when designing high-resolution DACs. The LPF consumes a substantial amount of silicon area, which becomes more pronounced as higher resolution is desired. As the output always repeats at intervals of 2N sampling periods, the LPF must be capable of attenuating frequencies that are increasingly closer to twice the input frequency range with each additional bit of resolution.

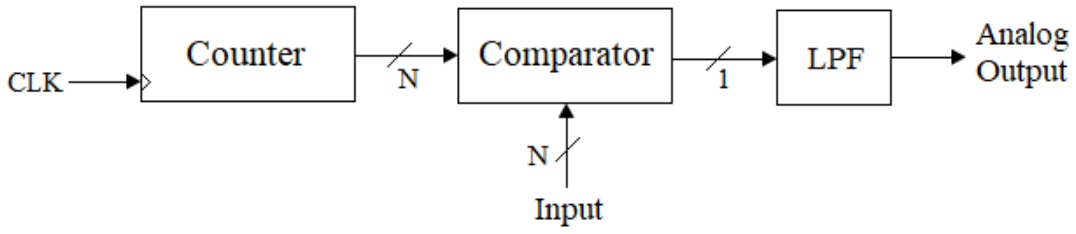


Fig. 2.9: Structure of a PWM DAC

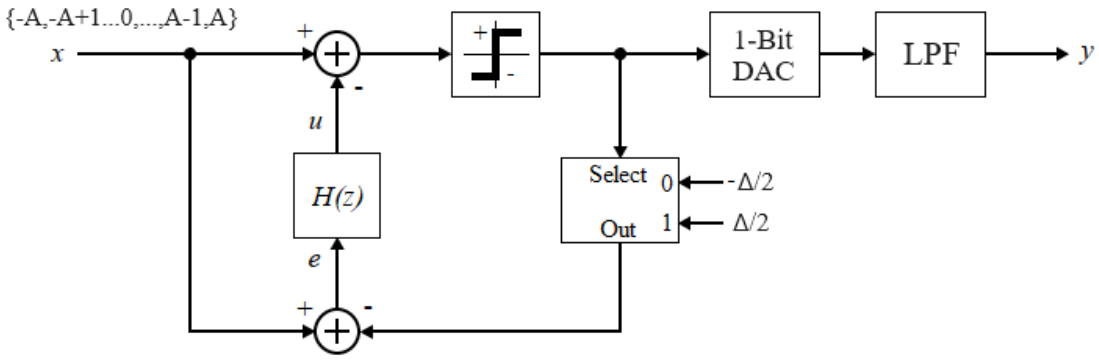


Fig. 2.10: Structure of a ΔΣ DAC

2.3.4 ΔΣ DACs

Much like PWM DACs, ΔΣ DACs [17] produce a bitstream at the output of a digital circuit which is then passed through a LPF to produce the intended signal. Since the only two analog component are the 1-bit DAC and the LPF, ΔΣ DAC is considered as an almost all-digital DAC. The structure of a typical ΔΣ DAC is shown in Fig. 2.10.

In a typical DAC which operates at a sampling frequency f_s, the quantization noise tends to be concentrated in the lower frequencies, making it challenging to adequately address using a low pass filter. However, when oversampling is employed, this same quantization noise gets spread out across a wider frequency range up to $KF_s/2, (K > 1)$, resulting in a significant reduction of the noise observed within the input signal band. Additionally, the noise-shaping characteristics of the ΔΣ modulator, facilitated by the filter in its feedback loop, further push the noise towards higher frequencies, effectively moving it outside of the input signal band. The quantization noise distribution across frequencies of the typical DAC and ΔΣ DAC are shown in Fig. 2.11.

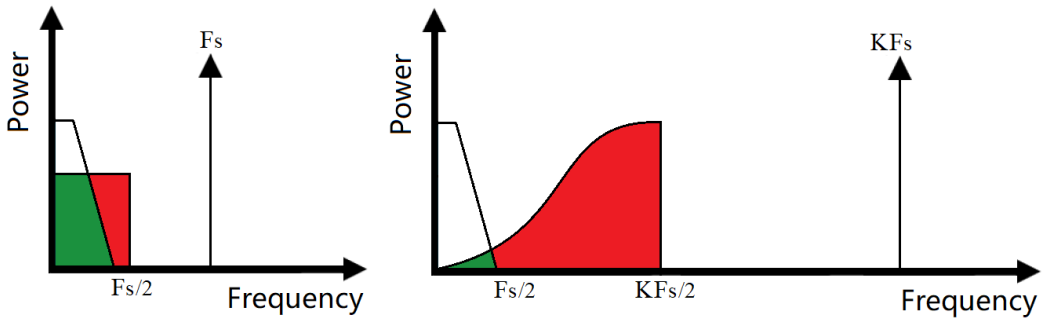


Fig. 2.11: Noise distribution across frequency (left: a typical DAC; right: a ΔΣ DAC)

These unique properties make the ΔΣ modulator an excellent choice for achieving high-resolution results at a lower cost. Furthermore, its low power consumption and bandwidth make it well-suited for applications involving voiceband and general audio signal processing.

Nonetheless, the size of the filter remains a primary factor determining the amount of silicon area utilized by the ΔΣ DAC. Consequently, additional efforts are necessary to further reduce the size of the filter. Moreover, the substantial number of memory elements needed to store a single DAC input code poses a limitation in memory-based ΔΣ DACs, particularly when developing high-resolution DACs.

2.3.5 Segmented DACs

To enhance DAC resolution and minimize hardware costs, the concept of segmentation is introduced, which is the core idea of this thesis. A segmented DAC [17] refers to a type of Digital-to-Analog converter that divides the digital input word into distinct "segments" and processes them individually. This approach may involve utilizing specific DAC architectures for each segment,

leveraging their individual strengths and mitigating their respective weaknesses to optimize overall performance.

Fig. 2.12 depicts a segmented string DAC architecture [18], illustrating a 6-bit configuration composed of two 3-bit string DACs. The voltages at each tap within the DAC are labeled for clarity. In this design, the resistors within the two strings must be identical, except for the top resistor in the MSB string, which should be smaller ($1/2^k$ of the others). Furthermore, the Least Significant Bit (LSB) string comprises $2^k - 1$ resistors instead of the standard 2^k. Notably, as there are no buffers, the LSB string is connected in parallel with the resistor in the MSB string it is switched across, thus introducing a load. This results in a voltage drop of precisely 1 LSB from the LSB DAC, as required by the design.

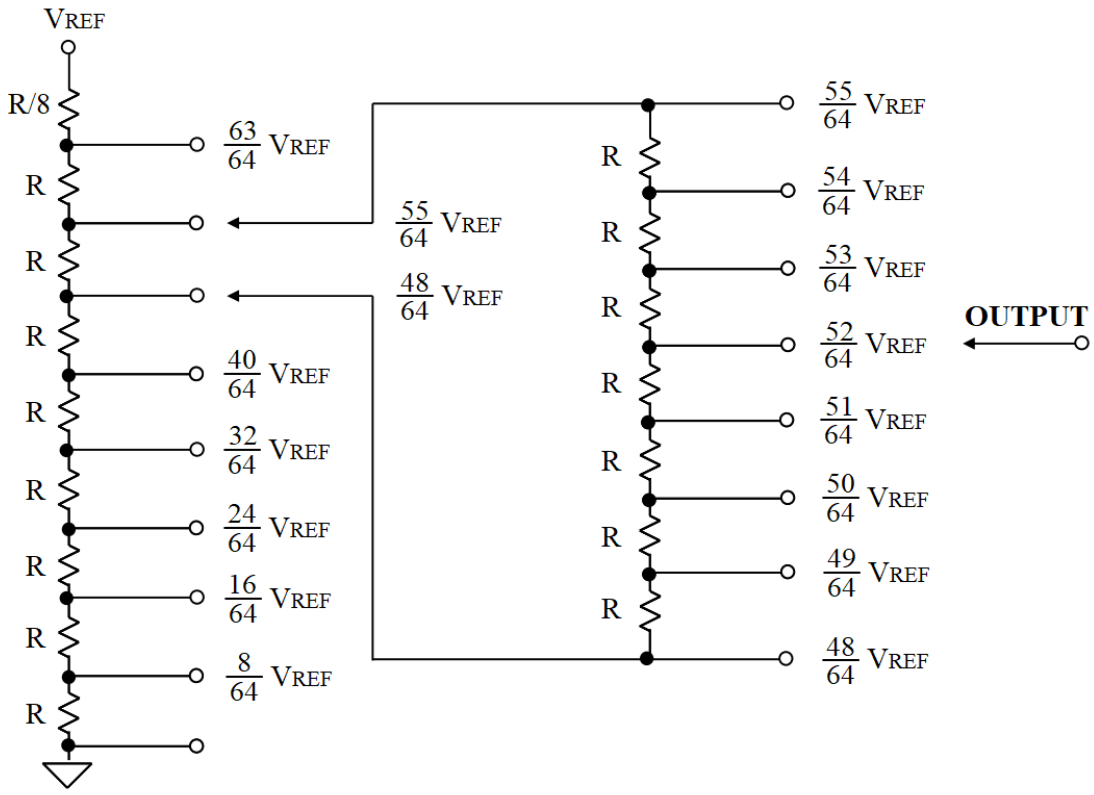


Fig. 2.12: An example of segmented DAC architecture

A designer might opt to utilize a segmented DAC for various reasons. One possible reason is that the specific performance criteria they need to meet may not be achievable with a single architecture alone. In such cases, a segmented DAC can offer the necessary flexibility to meet those requirements. Another consideration is the hardware complexity associated with achieving a linear increase in the number of bits. As the number of bits increases, the hardware required to implement a DAC grows exponentially. To mitigate this hardware complexity and save on resources, segmenting the DAC may be a viable solution. Hence, the decision to employ a segmented DAC can stem from the need to address specific performance metrics and optimize hardware utilization.

2.4 Applications of Oscillators

Recent literature underscores the significance of both digital and analog oscillators due to advances in integrated circuit technologies. Digital oscillators, often in the form of numerically controlled oscillators (NCOs), are gaining attention for their accuracy, programmability, and compatibility with digital signal processing (DSP) algorithms. Research focuses on improving NCO resolution and spectral purity for applications such as software-defined radios and radar systems [19].

Analog oscillators remain vital in scenarios where precision, phase noise, and rapid response are paramount. Recent studies explore novel architectures and topologies to address challenges like power efficiency, phase noise mitigation, and integration with analog functions [20]. Integrating analog oscillators with digital compensation techniques also aims to achieve superior

performance [21].

Moreover, oscillators are intimately linked with DACs in various applications [4]. Some ideas of how they are connected are shown in Fig. 2.13.

In summary, oscillators are vital components in various DAC applications, providing clock signals, driving ΔΣ modulators, serving as references for PLLs, and enabling precise frequency synthesis. The stability, precision, and spectral purity of the oscillators have a direct impact on the performance of DACs and DDFS systems. A high-quality oscillator can significantly enhance the overall performance.

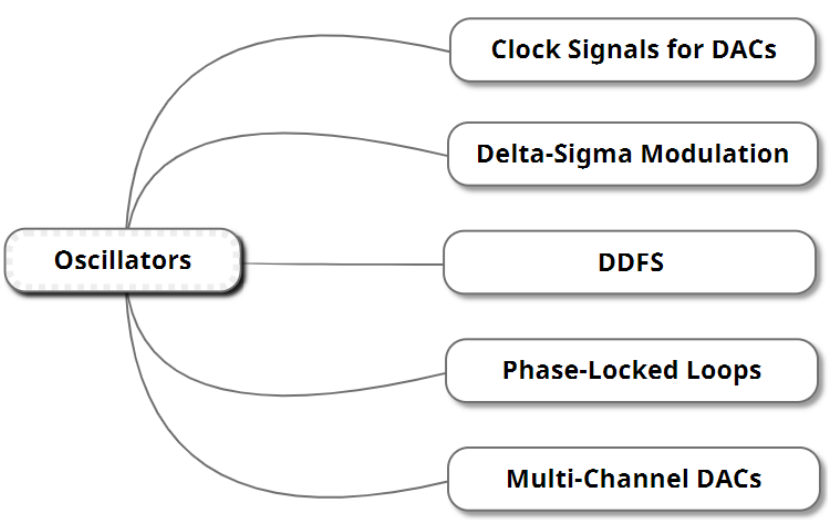


Fig. 2.13: Some applications of oscillators in DACs

2.5 Summary

Within this chapter, an extensive literature review relevant to the thesis has been conducted. Initially, the principle of digital-to-analog (D/A) conversion is elucidated, followed by the introduction of DACs and their associated performance metrics. Subsequently, various DAC architectures documented in the literature are presented, accompanied by a comprehensive examination of their respective advantages and disadvantages.

Chapter 3: ΔΣ Modulators

This chapter provides an introduction to the theory of ΔΣ modulation, including the structure and performance characteristics of ΔΣ modulators. The chapter then delves into a detailed design of a one-bit ΔΣ modulator, followed by a comprehensive discussion of a MATLAB/Simulink simulation.

3.1 Theory of ΔΣ Modulation

3.1.1 Structures of One-Bit ΔΣ modulators

Assuming no loss of generality, the block diagram shown in Fig. 3.1 can represent any high-order one-bit ΔΣ modulator. A linear model is shown in Fig. 3.2.

The input x could be a digital or analog signal while the output y of the modulator is a 1-bit digital signal. There are three main blocks: Two linear filters, H_1 and H_2, and a one-bit quantizer. A delta-sigma modulator employs a feedback loop where the band-limited input signal, x, undergoes filtering via H_1, and the resulting signal is then subtracted by the feedback signal from H_2, resulting in e, the input signal to the quantizer. The quantizer output is utilized directly as the modulator output, y, while also being filtered by H_2 to generate the feedback signal. This configuration creates a non-linear system with feedback, which characterizes a delta-sigma modulator.

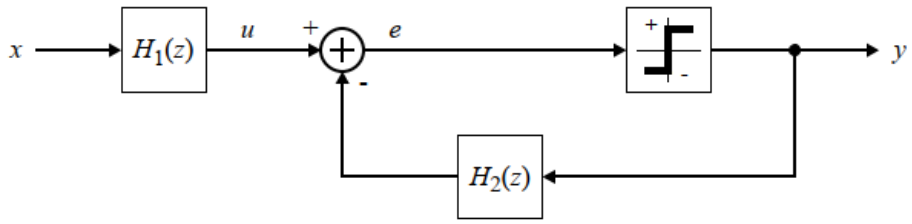


Fig. 3.1: Block diagram of the general 1-bit delta-sigma modulator

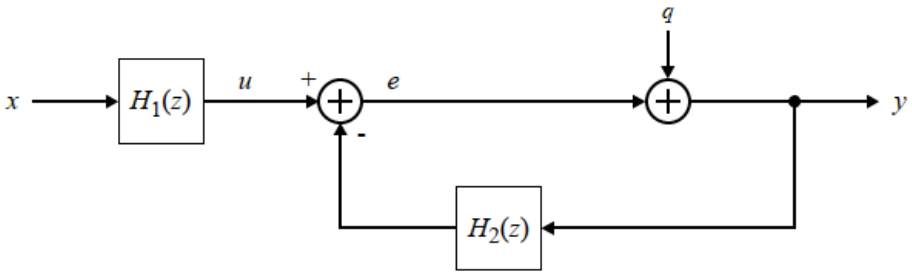


Fig. 3.2: Linear model of the general 1-bit delta-sigma modulator

The quantizer can be modelled as a source of noise as shown in Fig. 3.2. Therefore, the output of the quantizer y can be expressed as the sum of the initial quantizer input, e, and the quantization noise, q, generated by the additional adder. Using this noise model, it is possible to perform a z-transform on the system, resulting in the equation

$$Y(z) = \frac{H_1(z)}{1 + H_2(z)} \cdot X(z) + \frac{1}{1 + H_2(z)} \cdot Q(z) \tag{3.1}$$

We can deduce that the signal transfer function is the factor multiplied by $X(z)$, given as

$$STF(z) = \frac{H_1(z)}{1 + H_2(z)} \tag{3.2}$$

while the noise transfer function is the factor multiplied by $Q(z)$, given as

$$NTF(z) = \frac{1}{1 + H_2(z)} \tag{3.3}$$

The two transfer functions can fully describe the spectral characteristics of the sigma-delta modulator. Therefore, the z-transform of the output signal y can be expressed using these two transfer functions as follow:

$$Y(z) = STF(z) \cdot X(z) + NTF(z) \cdot Q(z) \tag{3.4}$$

The noise source $Q(z)$ can be assumed as white noise, which is reasonably precise for modulators that are higher than the first order.

From the noise transfer function expressed in Eqn. (3.3), we can conclude that $H_2(z)$ must have a significant value in the relevant bandwidth to minimize the noise at the output. However, considering the signal transfer function presented in Eqn. (3.2), we must ensure that $H_1(z)$ has a large value to offset the effect of $H_2(z)$. Ultimately, in the ideal scenario, $STF(z)$ should equal one, and $NTF(z)$ should be zero.

Therefore, in this thesis, we will utilize a modified model different from the linear model shown in Fig. 3.2 to achieve a unity gain, while also simplifying the overall topology. Given the definition of STF in Eqn. (3.2), if the STF is unity, we can derive that:

$$H_1(z) = 1 + H_2(z) \tag{3.5}$$

Since we derived a quite simple relation between the two filters H_1 and H_2. We could use one single filter $H(z)$ instead of $H_2(z)$. The modified topology [1] is shown in Fig. 3.3. We could further simplify the modulator structure as shown in Fig. 3.4.

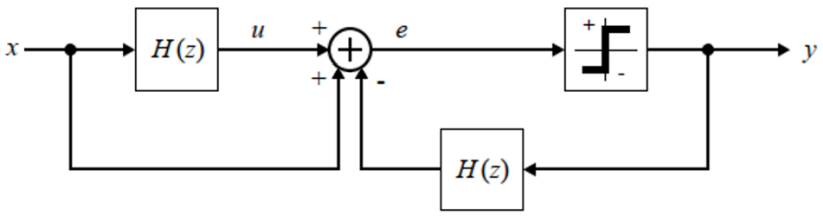


Fig. 3.3: Modified signal flow graph of the general delta-sigma modulator

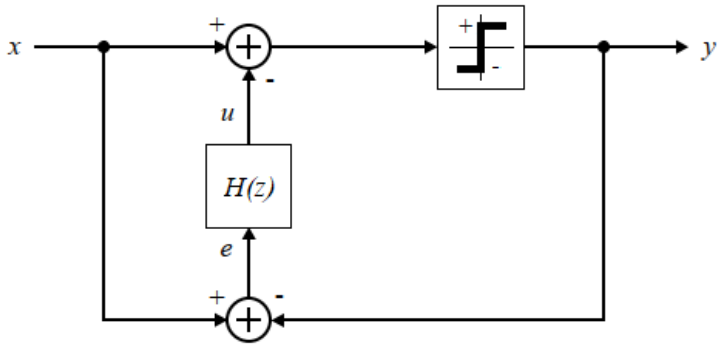


Fig. 3.4: Structure of the general delta-sigma modulator with a unity STF

Substituting the quantizer with its linear model results in a direct path from the input signal to the output, which then loops back to the filter input. Additionally, the input signal is fed forward to the filter input, effectively canceling out the signal feedback. Therefore, we have a unity gain STF, expressed as:

$$STF = \frac{1}{(1+H)-H} = 1 \tag{3.6}$$

The NTF now is as follow:

$$NTF(z) = \frac{1}{1+H(z)} \tag{3.7}$$

As a result, Fig. 3.3 shows the structure of a unity gain ΔΣ modulator which we will use in the later chapters.

Based on different needs, ΔΣ modulator can take digital or analog inputs while producing a 1-bit digital output stream containing all the information of the input signal. We can fulfil A/D and D/A conversion by using a ΔΣ modulator.

Fig. 3.4 shows the structure of a ΔΣ modulator if the input signal is analog. As for ΔΣ DAC, we could feed the 1-bit output stream into a 1-bit DAC and a low pass filter. The structure of a 1-bit ΔΣ DAC is shown in Fig. 3.5.

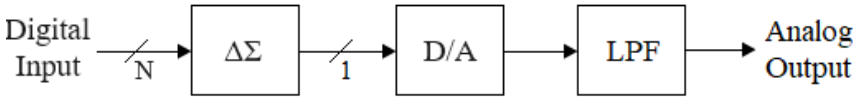


Fig. 3.5: Structure of 1-bit ΔΣ DAC

In this thesis, the focus is on analog oscillators with segmentation. Thus, the input of the ΔΣ modulator will be a digital input.

3.1.2 Performance Characteristics of ΔΣ modulation

The performance characteristics of a ΔΣ modulator are heavily influenced by several factors: the order, the amount of quantization noise generated by the quantizer, and the LPF's bandwidth and its order. An error source is introduced due to the quantization process carried out by the quantizer, which has a significant impact on the output signal quality after passing through the LPF.

To estimate the quantization noise, the Power Spectral Density (PSD) is taken to be a white noise with an average power as

$$P_Q = \frac{\Delta^2}{12} \tag{3.8}$$

where Δ is the interval of quantization. Assume the PSD of the input $X(z)$ is defined as $S_X(f)$, and the sampling frequency is denoted as f_s. According to Eqn. (3.4), we can derive the PSD of the output $Y(z)$ as

$$S_Y(f) = S_X(f) \mid STF\left(e^{j2\pi\left(\frac{f}{f_s}\right)}\right)\mid^2 + S_Q \mid NTF\left(e^{j2\pi\left(\frac{f}{f_s}\right)}\right)\mid^2 \tag{3.9}$$

Since the magnitude of the STF is equal to one and the quantizer's PSD is equal to the average power P_Q spread across the sampling frequency f_s, we could simplify the previous equation as

$$S_Y(f) = S_X(f) + | NTF\left(e^{j2\pi\left(\frac{f}{f_s}\right)}\right) |^2 . \frac{1}{f_s}\frac{\Delta^2}{12} \tag{3.10}$$

Assume that the incoming bandwidth of the signal $x(t)$ is denoted as BW. Further, as the sampling rate f_s is significantly higher than BW, we can define the ratio of the Nyquist frequency of the sampling process to input signal bandwidth as the "oversampling ratio" (OSR), which is expressed as

$$OSR = \frac{f_s/2}{BW} \tag{3.11}$$

The output of the system consists of both noise and signal components, and hence the output signal-to-noise ratio (SNR) calculated over the LPF bandwidth can be expressed as

$$SNR_{Mod} = \frac{\int_0^{BW} S_X(f)df}{\frac{\Delta^2}{12f_s}\int_0^{BW} | NTF(f) |^2 \, df} \tag{3.12}$$

SNR is a rather important characteristic in the design of a ΔΣ modulator. It evaluates the quality of a modulator. Assume the amplitude of the input signal is A , the power spectral density for the input signal would be

$$S_X(f) = \frac{1}{4} A^2(\delta(f-\omega) + \delta(f+\omega)) \tag{3.13}$$

Now by combining Eqn. (3.12) and (3.13), one can state the output SNR defined over the bandwidth LPF as

$$SNR_{Mod} = \frac{6f_s A^2}{\Delta^2 \int_0^{BW} | NTF(f) |^2 \, df} \tag{3.14}$$

3.2 Design of a One-bit ΔΣ Modulator

Figure 3.3 presents the general topology of a ΔΣ modulator. The essence of this design centers around the loop filter $H(z)$, which significantly influences

the noise shaping at the output stage. The subsequent section will delve into an in-depth exploration of the design process.

3.2.1 Feasibility Study

The primary constraint of the ΔΣ modulators is stability. Due to the presence of the one-bit quantizer, which is a highly non-linear element, the stability of these circuits cannot be predicted solely using methods applicable to linear systems. Nevertheless, according to an essential stability requirement outlined in [22], we can achieve absolute stability for the modulator by ensuring that the magnitude of the Noise Transfer Function remains below 2.0 across all frequencies, i.e.,

$$\max\left(\left| NTF(e^{j2\pi f}) \right|_{0<f<\frac{1}{2}}\right) < 2.0 \tag{3.15}$$

It's also important to note that the stability criterion for discrete-time linear systems is a fundamental consideration in system analysis and design. This criterion dictates that the poles of the system must lie within the unit circle in the z-plane.

Besides the stability issue, for an N-bit ΔΣ modulator to be realizable, each feedback loop within it must include a minimum of one unit delay. As shown in Figure 3.3, the transfer function $H(z)$ must be causal, the zeros of $H(z)$ are fewer than its poles, which means the coefficient of the denominator of $H(z)$ is 0. This constraint can be stated as

$$NTF(z)|_{z=\infty} = 1 \tag{3.16}$$

3.2.2 Noise Transfer Function Structure

Given a desired SNR, designing the NTF is generally prioritized to fulfill the noise requirements according to Eqn. (3.14). One can obtain the constraints on NTF as

$$\int_0^{BW} |\, NTF(f)\,|^2\, df \leq \frac{6 f_s A^2}{\Delta^2 SNR_{Mod,max}} \tag{3.17}$$

Assume the NTF to be of the general form

$$NTF(z) = \frac{b_N z^N + b_{N-1} z^{N-1} + \cdots + b_1 z + b_0}{z^N + a_{N-1} z^{N-1} + \cdots + a_1 z + a_0} \tag{3.18}$$

For the 1-bit ΔΣ modulator to be realizable, the constraint described in Eqn. (3.16) suggest that $b_N = 1$. Thus, the NTF is written as

$$NTF(z) = \frac{z^N + b_{N-1} z^{N-1} + \cdots + b_1 z + b_0}{z^N + a_{N-1} z^{N-1} + \cdots + a_1 z + a_0} \tag{3.19}$$

Now we will proceed to the design and selection of loop filter $H(z)$. Based on Eqn. (3.7), we can derive the filter block transfer function, according to

$$H(z) = \frac{1}{NTF(z)} - 1 \tag{3.20}$$

By plugging this into Eqn. (3.13), the filter transfer function becomes

$$H(z) = \frac{(a_{N-1} - b_{N-1}) z^{N-1} + \cdots + (a_1 - b_1) z + (a_0 - b_0)}{b_N z^N + b_{N-1} z^{N-1} + \cdots + b_1 z + b_0} \tag{3.21}$$

Additionally, to optimize SNR, it is crucial for the zeros of NTF, which correspond to the poles of $H(z)$ to be positioned precisely on the unit circle within the denoted bandwidth. The design of the poles of the NTF can be accomplished using dedicated tools such as DSMOD, which are documented in prior works [23].

3.2.3 Transfer Function Design Method

As stated in the previous section, we could derive the NTF given a specific SNR. In this section, we will address the methods of implementing a modulator with a given NTF. The task of realizing a modulator involves determining the structure coefficients that realize the intended NTF. To achieve this, each term in the numerator and denominator of the NTF is formulated as a function of the structure coefficients. With most simulation tools, floating-point formats are employed for coefficients, accumulators, and output variables of the loop filter. However, when transitioning to a physical implementation, such as an FPGA, fixed-point representation becomes essential. This transition from simulation to hardware introduces quantization challenges. Specifically, the reduction in precision for loop filter coefficients can lead to deviations from the intended operational behavior. Thus, the topology of mapping the transfer function needs to be carefully chosen to minimize quantization error.

Given the format of Eqn. (3.20), an IIR filter needs to be implemented. In prior work, the Resonator Cascade topology (RC) was used [11][23]. Another common digital filter topology that can be used is the Direct Form II (DF-II) structure.

The RC structure requires additional delay elements for each second-order section, which can increase memory requirements compared to DF-II structures. In addition, the modular nature of cascade structures can introduce some additional complexity in managing and processing the various filter sections. On the other hand, DF-II structure can be numerically sensitive [11], especially when dealing with high-order filters or very small filter coefficients. This sensitivity can lead to issues such as coefficient quantization errors and round-

off noise, affecting the overall filter performance.

In this thesis, DF-II structures for $H(z)$ is chosen. A prominent advantage of the DF-II structure lies in its ease of implementation, making it more FPGA-friendly. The RC structure often involves multiple stages with intricate interconnections, which can present challenges in its design and analysis, whereas DF-II exhibits a more straightforward and efficient mapping to FPGA hardware. Secondly, our specific FPGA implementation allocates high precision in coefficient mapping, which significantly mitigates the quantization errors associated with the DF-II structure. Moreover, FPGA devices have been advancing, providing more resources and better precision for fixed-point arithmetic. Newer FPGAs often have improved DSP slices and fixed-point arithmetic units, which can result in better numerical accuracy [25]. The overall effectiveness of this choice is evident in Chapter 7.1, where we demonstrate that the implementation of DF-II structures in FPGA can yield accurate results.

The structure is simply based on the difference equation by using inverse z-transform of $H(z)$. Eqn. (3.20) can be rewritten as:

$$y(n-N) + b_{N-1}y(n-(N-1)) + \cdots + b_1 y(n-1) + b_0 y(n) = (a_{N-1} - b_{N-1})x(n-(N-1)) + \cdots + (a_1 - b_1)x(n-1) + (a_0 - b_0)x(n) \quad (3.22)$$

The diagram of DF-II structure for the loop filter is shown in Fig. 3.6.

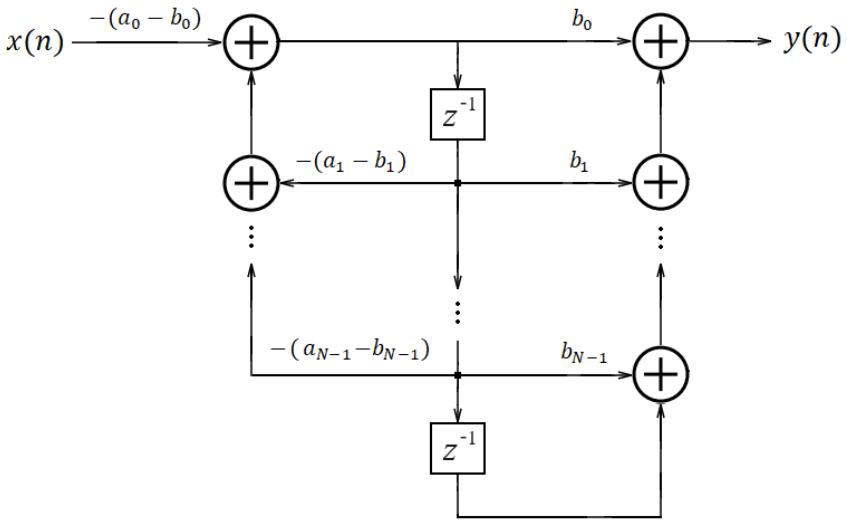


Fig. 3.6: Structure of DF-II for H(z) implementation

3.3 One-Bit ΔΣ Modulator Simulation

In this section, the one-bit delta-sigma modulator is designed and simulated in MATLAB/Simulink, assuming unsigned integer number precision. A given modulator design can be simulated at various signal levels to determine the maximum SNR and the corresponding input level, as well as the available dynamic range. The structure of the ΔΣ modulator is given in Fig. 3.7, where the input is a digital sinusoidal signal ranging from $-A$ to A, with a frequency of f_0.

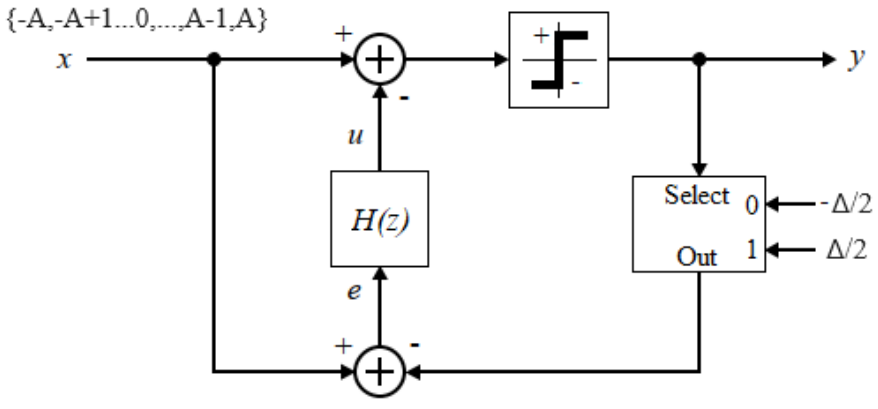


Fig. 3.7: The structure of a 1-bit unsigned ΔΣ modulator

The parameters for ΔΣ modulator simulation are summarized in Table 3.1. They are to be implemented under a 32 bits number system.

Parameters	A	f_0	f_s	$\Delta/2$	N	OSR
Values	2^8	$0.0017Hz$	$1Hz$	2^9	32	256

Table. 3.1: The parameters for an N-bit unsigned ΔΣ modulator simulation example

Assume the desired maximum SNR is greater than 80 dB. Given the parameters shown in Table. 3.1, we can derive the inequality for the noise power according to Eqn. (3.17)

$$\int_0^{BW} |\, NTF(f)\,|^2\, df \leq\ 4.7434 \times 10^{-10} \tag{3.23}$$

To meet this specification shown in Eqn. (3.23), we choose a 3rd order inverse-Chebyshev NTF with a stopband of 0.005 Hz and a stopband attenuation of -75 dB, assuming the sampling frequency is 1 Hz. The transfer function of NTF is found using MATLAB, which is given by

$$NTF(z) = \frac{z^3 - 2.9993z^2 + 2.9993z - 1}{z^3 - 2.1659z^2 + 1.6460z - 0.4295} \tag{3.24}$$

The magnitude response of $NTF(f)$ is shown in Fig. 3.8.

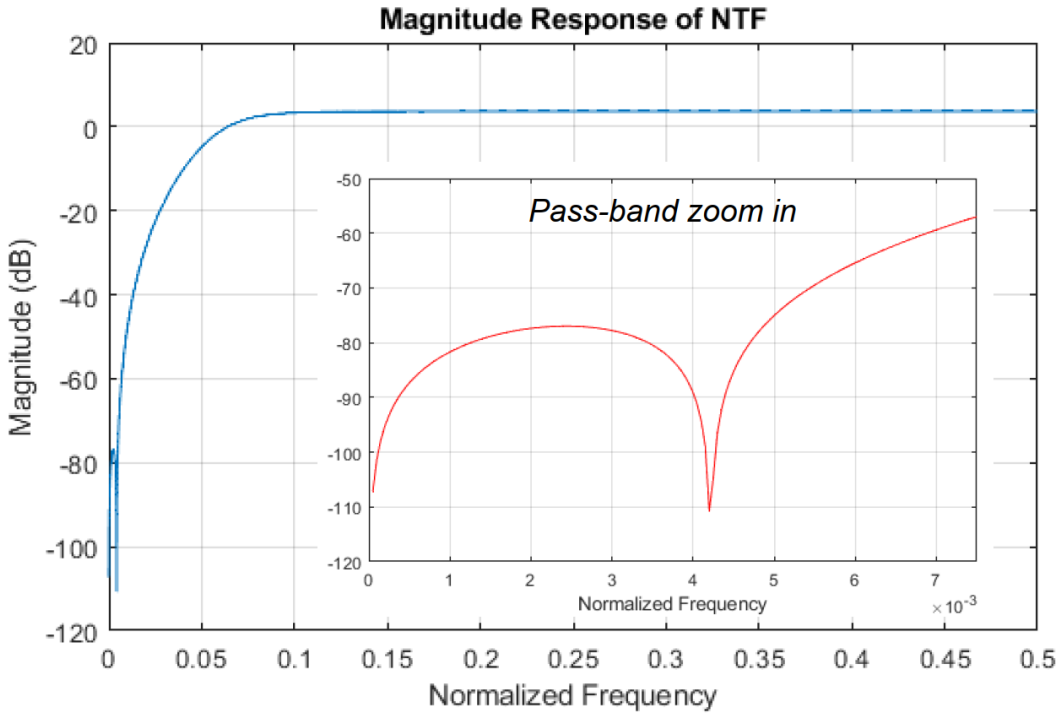


Fig. 3.8: Magnitude response of the noise transfer function

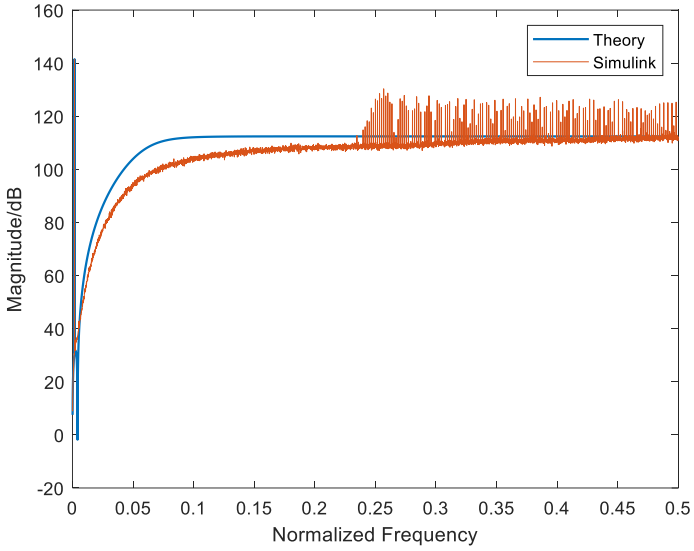
By integrating $|\,NTF(f)\,|^2$ from 0 Hz to BW, we can calculate that $\int_0^{BW} |\,NTF(f)\,|^2\,df = 9.0805 \times 10^{-11}$, which satisfies our 80 dB SNR specifications. Using Eqn. (3.12), the theoretical SNR is then calculated to be 96.1592 dB.

The loop filter transfer function of the modulator can be calculated using Eqn. (3.22) and (3.26), which leads to

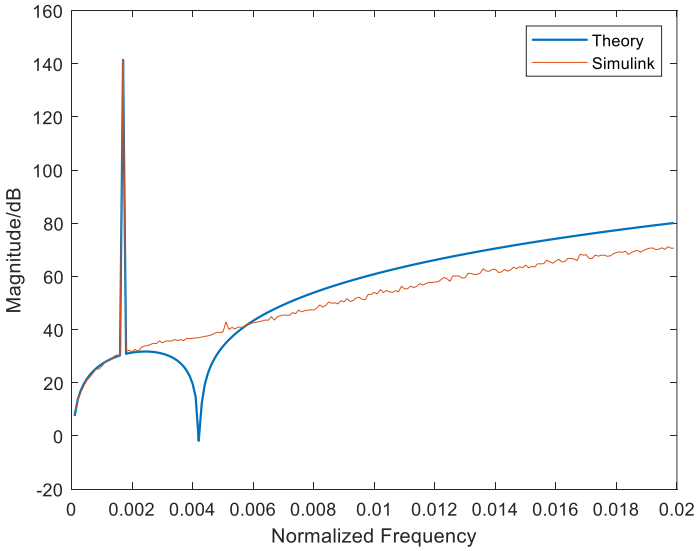
$$H(z) = \frac{0.8334z^2 - 1.3533z + 0.5705}{z^3 - 2.9993z^2 + 2.9993z - 1} \tag{3.27}$$

By utilizing the parameters shown in Table. 3.1 and the loop filter transfer function from Eqn. (3.27), the power spectral density (PSD) of the modulator output in Simulink versus theoretical results are shown in Fig. 3.9. In consideration of comparison in the future, the amplitude and quantizer threshold are scaled up by 2^{16}. We could see the simulation output is very close to the

theoretical one. The output contains the information of a sinusoidal signal with a frequency of 0.0017 Hz and an amplitude of 2^{24}. However, we could also spot a slight spectral leakage near the side lobe.



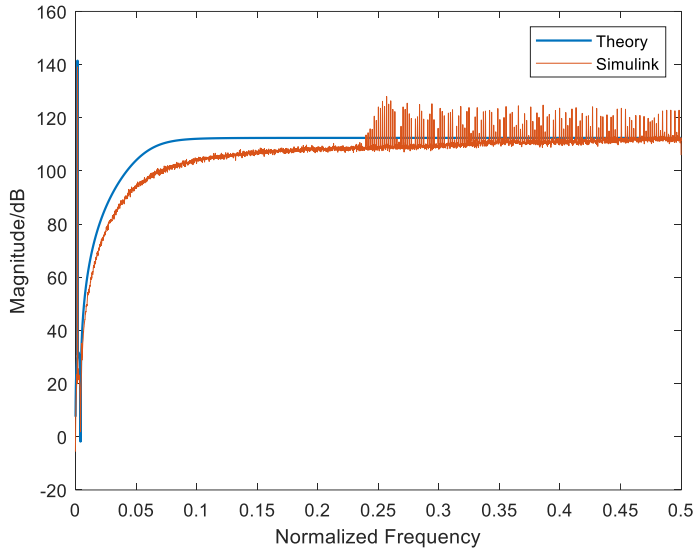
(a)



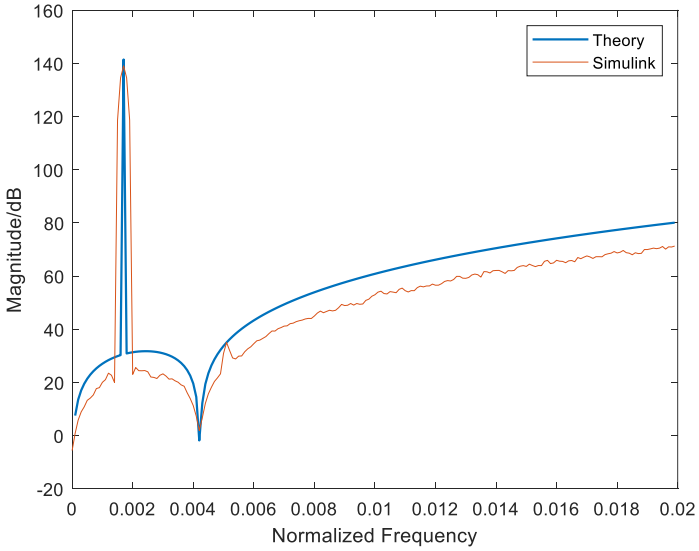
(b)

Fig. 3.9: PSD of the outputs for sinusoidal inputs given the 3rd order inverse Chebyshev loop filter (without windowing) (a): Nyquist frequency (b): zoom in for in-band frequency

To mitigate against spectral leakage during transform functions and reduces the frequency smearing due to the unavoidable incoherence of the delta-sigma modulator output [23], the resulting output was processed using the Dolph–Chebyshev window described in [26]. This window has sidelobes that decay rapidly in the frequency domain. The PSD of modulator output with windowing is then shown in Fig. 3.10.



(a)



(b)

Fig. 3.10: PSD of the outputs for sinusoidal inputs given the 3rd order inverse Chebyshev loop filter with Dolph–Chebyshev window (a): Nyquist frequency (b): zoom in for in-band frequency

As can be seen in Fig. 3.10, windowing improves the overall performance and gives better noise shaping. The in-band SNR is calculated to be 96.90 dB, which is much greater than the desired SNR of 80 dB.

3.4 Summary

This chapter mainly presents three parts. To start with, the ΔΣ modulator topology with a unity Signal Transfer Function (STF) is presented and the performance characteristics are described, which is crucial for its application in delta-sigma oscillators. Next, the noise constraints and design for the modulator is discussed and various transfer function topology are provided. Due to ease of implementation, more FPGA resources and high accuracy, DF-II structure is chosen for transfer function mapping. Finally, simulations using MATLAB/Simulink are provided and the validity of these designs are confirmed.

Chapter 4: ΔΣ Oscillators

In this chapter, a ΔΣ oscillator is introduced that is designed to generate high-precision analog sinusoidal signals. The chapter begins by explaining the principle of oscillation and providing a mathematical analysis. Following that, the structure of a ΔΣ oscillator is explained and presented, followed by a detailed MATLAB/Simulink simulation.

4.1 Theory and Design of ΔΣ oscillators

4.1.1 Digital Oscillator Analysis

The Lossless Discrete Resonator (LDR) introduced in Section 1.2.2 can be available for designing digital filters using LC ladder networks [14]. Typically, the resulting structure of the digital filter comprises a group of coupled first-order integrators, forming a second-order resonator. These resonators are created by looping two integrators in a cascade. The block diagram of the LDR is shown in Fig. 4.1.

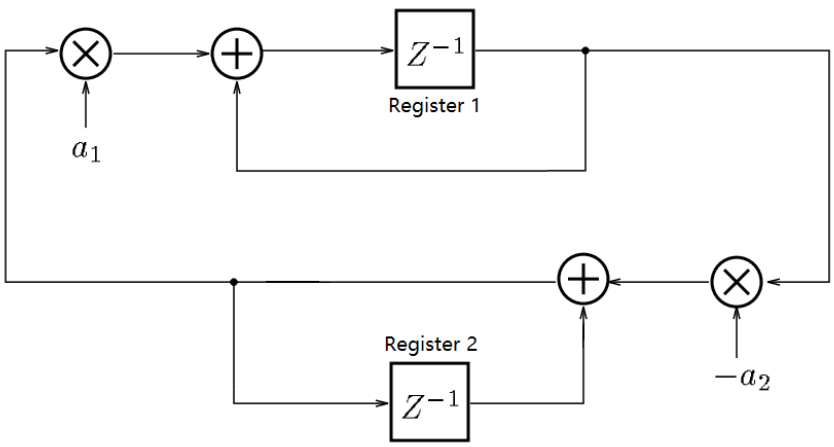


Fig. 4.1: Block diagram of the lossless discrete resonator (LDR)

Eliminating damping in the filter can lead to the realization of a digital oscillator. By employing this method, the resonant circuit depicted in Figure 4.1 can be utilized as a digital oscillator. The analog counterpart of this circuit is an LC-tank circuit [14]. There are two characteristics about LC-tank circuit: (1) Capacitor and inductor values shift the oscillation frequency while they cannot stop the circuit from oscillating; (2) Initial conditions imposed on the capacitor and inductor decide the oscillation amplitude.

Similarly in Fig. 4.1, the oscillation frequency can be controlled by parameters a_1 and a_2 while the oscillation amplitude can be adjusted by changing the initial values in the registers.

Assume the values in registers 1 and 2 in Fig 4.1 are $x_1(n)$ and $x_2(n)$ at time $t = nT$. We can derive the two difference equations as

$$x_1(n+1) = x_1(n) + a_1 x_2(n+1) \tag{4.1}$$

and

$$x_2(n+1) = -a_2 x_1(n) + x_2(n) \tag{4.2}$$

Applying z-transform to Eqn. (4.1) and Eqn. (4.2), we can eliminate $X_2(z)$ and derive one single equation as

$$z^2 X_1(z) + (a_1 a_2 - 2) z X_1(z) + X_1(z) = 0 \tag{4.3}$$

then the characteristic equation is

$$z^2 + (a_1 a_2 - 2) z + 1 = 0 \tag{4.4}$$

The roots of the quadratic equation can determine the circuit poles as

$$z_{1,2} = \left(1 - \frac{a_1 a_2}{2}\right) \pm \frac{1}{2}\sqrt{a_1 a_2 (a_1 a_2 - 4)} \tag{4.5}$$

According to Eqn. (4.5), if the product $a_1 a_2$ is larger than 4, the system can no longer oscillate because the two roots are real. Therefore, the circuit only oscillates when $0 < a_1 a_2 \leq 4$. We could rewrite the equation as

$$z_{1,2} = \left(1 - \frac{a_1 a_2}{2}\right) \pm j\sqrt{1 - \left(1 - \frac{a_1 a_2}{2}\right)^2} \tag{4.6}$$

If $0 < a_1 a_2 \leq 2$, the two roots are in the right-half plane, as for $2 < a_1 a_2 < 4$, the two roots are in the left-half plane, which could be all written as

$$z_{1,2} = e^{\pm j cos^{-1}\left(1 - \frac{a_1 a_2}{2}\right)} \tag{4.7}$$

Thus, the oscillation frequency ω_o can be derived from Eqn. (4.7). Assume the sampling frequency is $f_s = 1/T$, one finds

$$f_o = f_s \cos^{-1}\left(1 - \frac{a_1 a_2}{2}\right) \qquad \text{for } 0 < a_1 a_2 \leq 4 \tag{4.8}$$

Fig. 4.2 illustrates the relationship between the coefficients $a_1 a_2$ and the oscillation frequency. For values of $a_1 a_2$ between 0 and 4, the oscillation frequency varies continuously between 0 and $f_s/2$. Nevertheless, due to the coefficients being restricted to discrete values (as is the case in a finite precision implementation), the available selectable oscillation frequencies are also limited to discrete values. However, it is important to note that since the poles of the circuit always remain on the unit circle for all values of $a_1 a_2$ between 0 and 4, the circuit is guaranteed to oscillate even in a finite precision implementation.

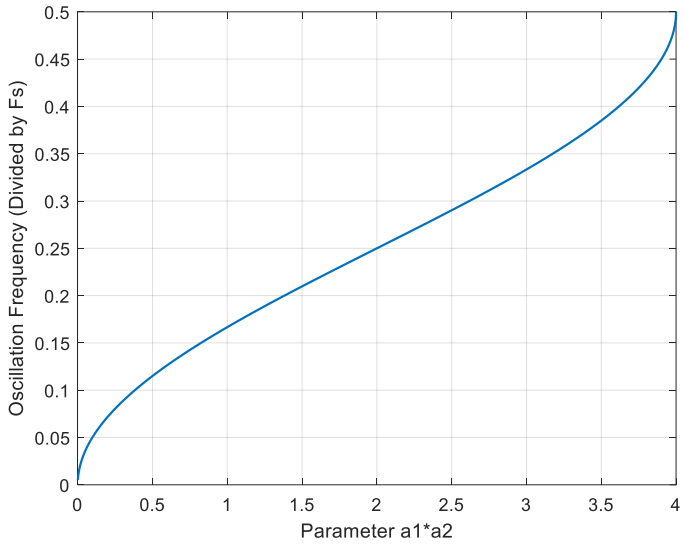


Fig. 4.2: Relationship between the oscillation frequency and the coefficients a1*a2

As for the oscillation amplitude, one can refer to the LC-tank analogy, where the amplitude of oscillation can be regulated by the initial conditions of the capacitor and inductor. Similarly, the initial values for the registers, $x_1(0)$ and $x_2(0)$ control the oscillation amplitude. Take another look at Eqn. (4.1) and (4.2), from the analysis in the previous section, it is evident that the solution to these equations is a pure tone with a frequency of ω_o. We could assume the solution is of the form

$$x_1(n) = A\sin(\omega_o nT + \phi) \tag{4.9}$$

According to Eqn. (4.9), by setting the sampling instants $n = 0$, one can obtain the relationship of the initial register value $x_1(0)$ and the signal amplitude and initial phase as

$$x_1(0) = A\sin\phi \tag{4.10}$$

And by setting $n = 1$, one can write that

$$x_1(1) = A\sin(\omega_o T + \phi) \tag{4.11}$$

To get rid of $x_1(1)$ in Eqn. (4.11), one can combining Eqn. (4.1) and (4.2) and express $x_1(1)$ in terms of the known parameters a_1, a_2 and register initial values $x_1(0)$, $x_2(0)$, which can be written as

$$x_1(1) = (1 - a_1a_2)x_1(0) + a_1x_2(0) \tag{4.12}$$

By merging Eqn. (4.10), (4.11), and (4.12), we can determine the values of the two unspecified constants A and ϕ. The resulting equations are presented as

$$A = \frac{x_1(0)}{\sin\phi} \tag{4.13}$$

and

$$\phi = \tan^{-1}\left(\frac{x_1(0)\sin(2\pi f_o T)}{(1 - a_1a_2 - \cos(2\pi f_o T))x_1(0) + a_1x_2(0)}\right) \tag{4.14}$$

Here we denote the amplitude of the signal as a positive value, then we can rewrite the amplitude of the signal as

$$A = \sqrt{1 + \frac{1}{\tan^2 \phi}} \tag{4.15}$$

By merging Eqn. (4.14), and (4.15), we can get rid of the parameter ϕ. The resulting amplitude is

$$A = \sqrt{x_1(0)^2 + \frac{((1 - a_1 a_2 - \cos(2\pi f_o T))x_1(0) + a_1 x_2(0))^2}{\sin^2(2\pi f_o T)}} \tag{4.16}$$

Amplitude vs $x_2(0)$	$x_1(0) = 0$	$x_1(0) \neq 0$
A	$A = \lvert C_1 x_2(0) \rvert$	$A = \sqrt{C_2 + C_3 x_2(0)^2}$
Amplitude vs $x_1(0)$	$x_2(0) = 0$	$x_2(0) \neq 0$
A	$A = \lvert C_4 x_1(0) \rvert$	$A = \sqrt{C_5 x_1(0)^2 + C_6 x_1(0) + C_7}$

Table. 4.1: Amplitude versus initial value of registers under different conditions

Eqn. (4.16) showcases the oscillation amplitude. To better demonstrate the amplitude versus the initial value of registers, Table. 4.1 simplifies Eqn. (4.16) under different conditions. Here $C_k (k = 1,2, \dots ,7)$ are constant values. To demonstrate more vividly, Fig. 4.3 shows the relationship between the amplitude and the initial value of one register given other parameters are set, i.e., $a_1 = 2^{-3}, a_2 = 2^{-13}, F_s = 1$. We could conclude that when the initial value of one register is set to 0, the amplitude is strictly linear in relation to the initial value of the other register, as also shown in Fig. 4.3 (a) and (c). However, if the initial value of one register is not zero, the oscillation amplitude is nearly linear when the initial value of the other register is high, however, it may be nonlinear when the value is low, as also shown in Fig. 4.3 (b) and (d).

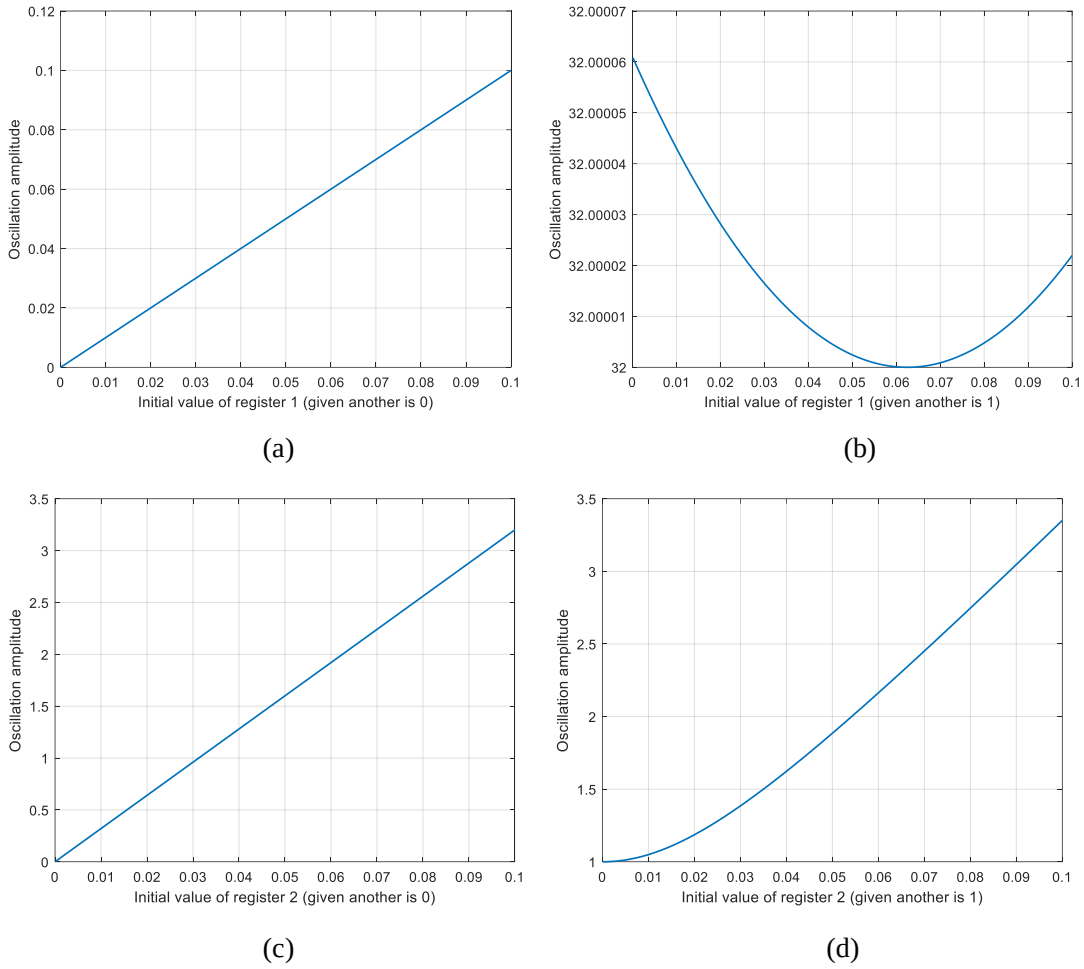


Fig. 4.3: Relationship between the amplitude and the initial value of the registers. Graph (a) illustrates the amplitude as a function of the register 1 given the other is 0. Graph (b) illustrates the amplitude as a function of the register 1 given the other is 1. Graph (c) and (d) illustrate the opposite situation of graph (a) and (b).

4.1.2 ΔΣ Oscillator Structure

As stated in Chapter 3, we could configure a ΔΣ DAC using a ΔΣ modulator, a 1-bit DAC and an LPF. By passing the output of the digital oscillator into a ΔΣ modulator and an LPF, the general structure of a ΔΣ oscillator can be shown in Fig. 4.4.

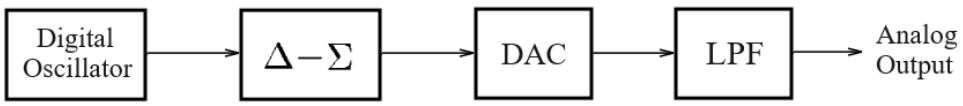
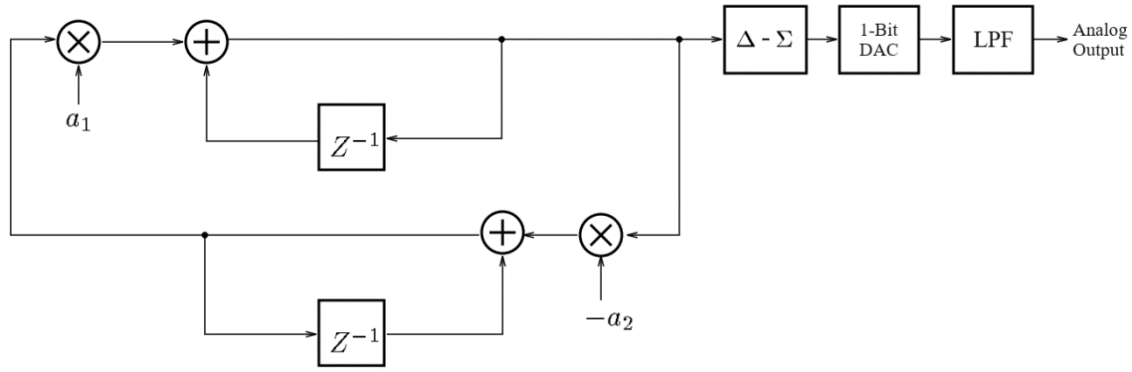
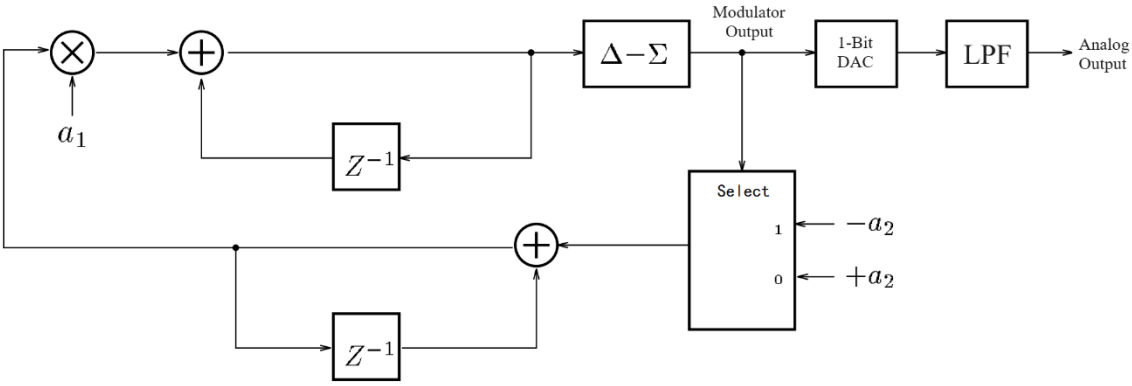


Fig. 4.4: The general structure of a ΔΣ oscillator

The configuration of a ΔΣ oscillator is shown in Fig. 4.5 (a). Additionally, as can be seen in Eqn. (4.8), the oscillation frequency is decided by the product $a_1 a_2$. Since the ΔΣ modulator produces an output of $\pm\Delta/2$, the multiplication by a_2 can be achieved using a two-input multiplexer. Therefore, by placing the delta-sigma modulator inside the resonator loop and replacing the two multi-bit multiplications with a two-to-one multiplexer, a more area-efficient implementation of the ΔΣ oscillator is shown in Fig. 4.5 (b).



(a)



(b)

Fig. 4.5: The configuration of the ΔΣ oscillator (a): A general ΔΣ oscillator structure (b): A more area efficient design of the ΔΣ oscillator

4.1.3 Noise Constraints

As can be seen from Fig. 4.5, the dominant source of noise at the output level is the quantization error originating from the ΔΣ modulator. That being said, to generate a sinewave given the specific number of bits, the loop filter $H(z)$ inside the ΔΣ modulator needs to be properly designed. Similarly in Chapter 3 and as given in Eqn. (3.14), (3.15) and (3.16), we have the constraints as follows for an N-bit oscillator as

$$\int_0^{BW} |\, NTF(f)\,|^2\, df = \frac{12 f_s \int_0^{BW} S_X(f) df}{\Delta^2 SNR_{Osc}} \tag{4.17}$$

and

$$H(z) = \frac{1}{NTF(z)} - 1 \tag{4.18}$$

In consideration of the FPGA implementation in the future, fractional numbers will introduce more calculation complexity into the system. Thus, we can adjust the value of the quantizer output of the ΔΣ modulator to $\Delta/2$ instead of 1. The change of quantizer output brings changes to Eqn. (4.8), (4.14) and (4.16), and we could rewrite the oscillation frequency and amplitude as

$$f_o = f_s \cos^{-1}\left(1 - \frac{a_1 a_2}{\Delta}\right) \qquad \text{for } 0 < \frac{a_1 a_2}{\Delta} \leq 4 \tag{4.19}$$

and

$$A = \sqrt{x_1(0)^2 + \frac{\left(\left(1 - \frac{2a_1 a_2}{\Delta} - \cos(2\pi f_o T)\right) x_1(0) + a_1 x_2(0)\right)^2}{\sin^2(2\pi f_o T)}} \tag{4.20}$$

The initial phase is calculated as

$$\phi = \sin^{-1} \frac{x_1(0)}{A} \tag{4.21}$$

Therefore, given a desired SNR, oscillation frequency ω_o and amplitude A, we need to properly choose the parameters to meet these 3 specifications. The unknown parameters are $F_s, a_1, a_2, \Delta, x_1(0), x_2(0)$ and the NTF. Based on Eqn. (4.17), (4.19) and (4.20), we now have 7 unknown parameters and 3 equations, which result in a degree of design freedom of 4 (i.e., if parameters can be set arbitrarily).

4.2 ΔΣ Oscillator Simulation

In this chapter, the delta-sigma oscillator presented in Fig. 4.5 (b) is simulated in MATLAB/Simulink, assuming signed integer number precision. Throughout this thesis unless otherwise stated, the Simulink simulations will be performed with 32-bit precision.

Assume we need to meet the following specifications: a sinewave signal is required to be generated by the oscillator with an amplitude of $A = 5181.5$ and oscillation frequency of $f_o = 7.8641$ Hz. The desired maximum SNR shall be higher than 80 dB. The bandwidth is set to $BW = 10\ Hz$.

By setting $F_s = 2000Hz,\ a_1 = 1, a_2 = 5$, Table. 4.2 gives all the parameters for the oscillator. The first row is the desired performance characteristics, the second row is defined by the designer and the third row is calculated using Eqn. (4.17), (4.19) and (4.20).

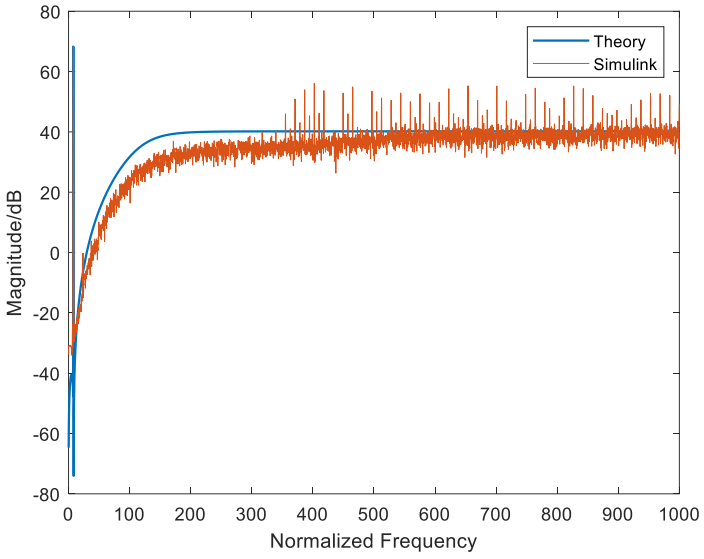
Required specifications	$\boldsymbol{A}$	$\boldsymbol{f_o}$	$\boldsymbol{SNR}$	
	5181.5	7.8641 Hz	$\geq 80dB$	
Parameters choose	$\boldsymbol{a_1}$	$\boldsymbol{a_2}$	$\boldsymbol{F_s}$	$\boldsymbol{x_1(0)}$
	1	5	2000 Hz	128
Parameters calculated	$\boldsymbol{\Delta/2}$	$\boldsymbol{x_2(0)}$	$\int_0^{BW} \mid NTF(f) \mid^2 df$	
	2^{13}	128	$\leq 6.0009 \times 10^{-9}$	

Table. 4.2: Amplitude versus initial value of registers under different conditions

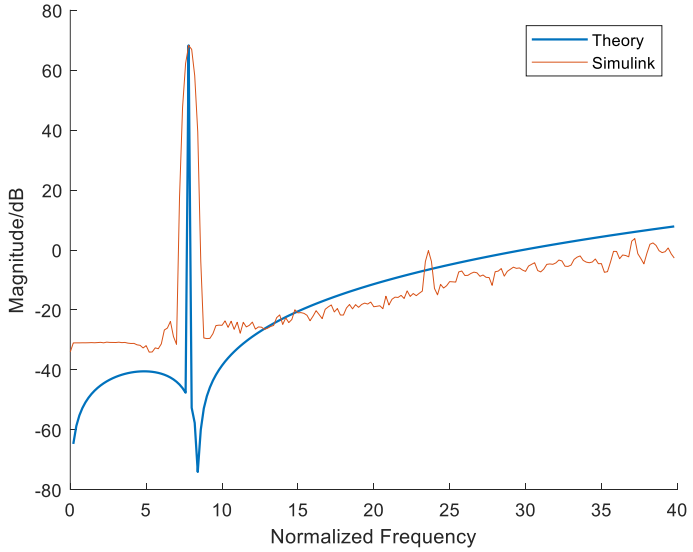
As we know from Chapter 3, the in-band noise integral for the 3rd order inverse-Chebyshev NTF produce a noise power of 9.0805×10^{-11}, while the stopband normalized frequency is 0.005 Hz, which in this case would be calculated as 10 Hz given the sampling frequency is set to 2000 Hz. This meets the noise power constraints listed in Table. 4.2. As reported in Chapter 3, the loop filter transfer function is given as:

$$H(z) = \frac{0.8334z^2 - 1.3533z + 0.5705}{z^3 - 2.9993z^2 + 2.9993z - 1} \tag{4.22}$$

The resulting modulator output was processed using the Dolph–Chebyshev window and the power spectral density of the simulation outputs is shown in Fig. 4.6. We could see the simulation closely aligns with the theoretical expectations.



(a)



(b)

Fig. 4.6: PSD of the outputs of ΔΣ Oscillator. (a): whole frequency; (b): in-band frequency

The results of the simulation are shown in Table. 4.2 in comparison with the theoretical values. Note that there is some drop in the actual maximum SNR but the desired SNR requirement is still satisfied. Also, the oscillation frequency is identical to that predicted by the theory. Likewise, the oscillation amplitude is also really close to what theory predicts.

	A	f_o	SNR
Theoretical	5181.5	7.8641 Hz	91.2114 dB
Simulation	5158.6	7.8641 Hz	86.9023 dB

Table. 4.2: Theoretical and simulation results of oscillation frequency, amplitude and system SNR

4.3 Summary

This chapter commences by introducing the concept of a second-order resonator, primarily employed in digital filter design. Building upon this foundation, the resonator's potential for use as a digital oscillator is explored. By diving into the mathematical background and making necessary adaptations, an improved and area-efficient circuit known as the ΔΣ oscillator is described. The chapter further substantiates the proposed design by providing MATLAB/Simulink simulations that confirm its validity and practicality in generating high-precision analog signals.

Chapter 5: Segmented ΔΣ Modulators

In this chapter, the fundamental concepts of segmentation and its integration into ΔΣ modulation are outlined. Additionally, the principles of a two-segment ΔΣ design are discussed in detail. Finally, the equation for attaining maximum SNR is derived in the concluding subsection, followed by a detailed MATLAB/Simulink simulation.

5.1 Segmentation introduction background

Most single-stage multibit modulators typically employ quantizers with six bits of resolution or less [24]. Increasing the number of quantizer bits enhances the SNR by 6 dB and improves modulator stability, allowing for more effective noise shaping. However, in certain cases, the performance requirements of high-resolution modulators cannot be met by a single modulator design.

To address this challenge, a possible solution is to use segmentation. As covered in the previous work by Emara Ahmed [12], by dividing the ΔΣ DAC into multiple segments, each handling a portion of the digital input, segmentation enables higher precision without causing an exponential increase in resources. The ΔΣ DACs used for different paths can be of different types and may not have matching resolutions. Hence, the properties influencing the coarse (MSB) and fine (LSB) sections need not to be identical or correlated.

5.2 Design of Two-Segment ΔΣ Modulators

5.2.1 Two-Segment ΔΣ Modulators Structure

The majority of segmented DACs typically consist of two sub-DACs, specifically the coarse and fine components. Assume that the input signal is $x(t)$, which has a total of N bits. The most significant bits (MSB) of the signal are allocated to the coarse input signal, denoted as $x_c(t)$. The least significant bits (LSB) of the input signal are allocated to the fine signal, denoted as $x_f(t)$. Assume that the digital input of the coarse word is NC bits wide and the fine word is NF bits wide, so that

$$N = NC + NF \tag{5.1}$$

Therefore, the input signal ranges from 0 to $2^{NC+NF} - 1$, while the coarse signal and the fine signal have ranges of from 0 to $2^{NC} - 1$, and 0 to $2^{NF} - 1$, respectively. The bit partition of the input signal $x(t)$ into components $x_c(t)$ and $x_f(t)$ is demonstrated in Fig. 5.1.

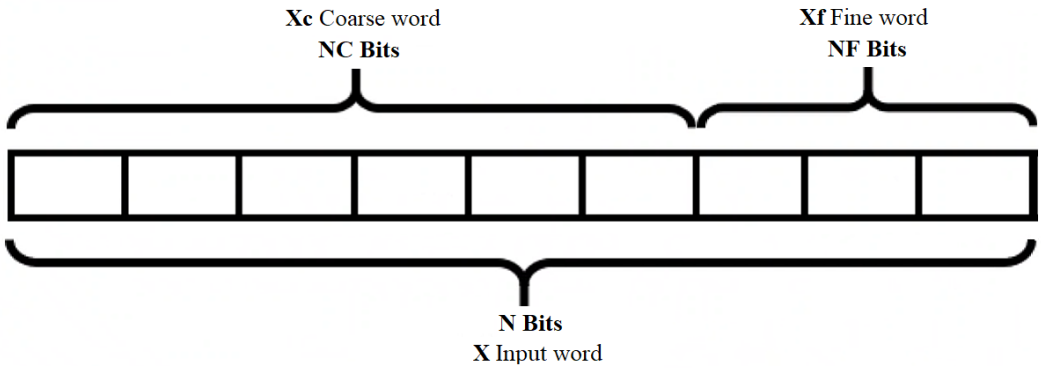


Fig. 5.1: Partition of the input signal

The coarse signal $x_c(t)$ can be described as:

$$x_c(t) = \left\lfloor \frac{x(t)}{2^{NF}} \right\rfloor \tag{5.2}$$

where $\lfloor w \rfloor$ is the function that takes a real number w as input and gives the greatest integer less than or equal to w as output. Consequently, the fine signal can be denoted as:

$$x_f(t) = x(t) - 2^{NF} \left\lfloor \frac{x(t)}{2^{NF}} \right\rfloor \tag{5.3}$$

By using a quantizer, the structure of the partition operation used to generate the coarse and fine components of the input signal is shown in Fig. 5.2. The digital input $x(t)$ is segmented into the coarse input $x_c(t)$ and the fine input $x_f(t)$.

The general structure of two-segment ΔΣ modulator is shown in Fig. 5.3. As can be seen, data partition is utilized in the structure. The coarse and fine input are then fed into two different ΔΣ modulators. Subsequently, the coarse modulator output is scaled by 2^{NF} while the fine modulator output remains unscaled. By combining the scaled outputs together and feeding it through the 1-bit DAC and a reconstruction LPF, the output analog signal is formed.

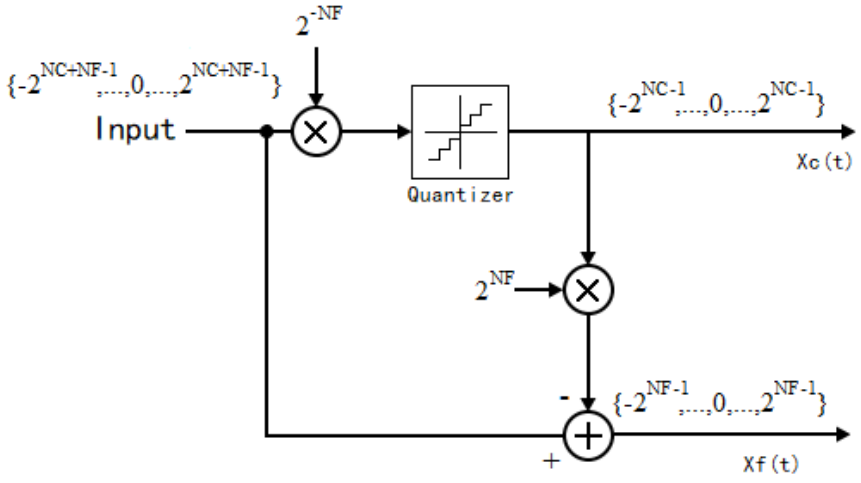


Fig. 5.2: Realization of partition of the input signal

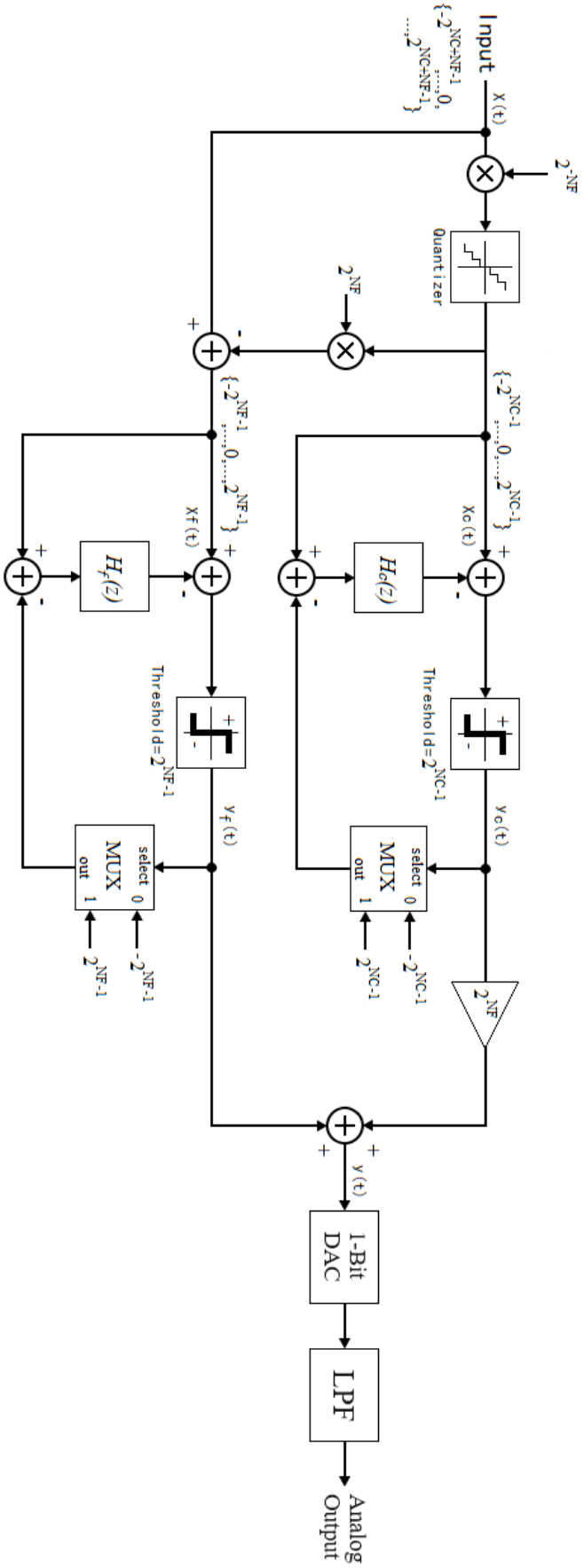


Fig. 5.3 Overall structure of two-segment ΔΣ modulator

5.2.2 Noise Constraints

As shown in Fig. 5.3, the two-segment ΔΣ modulator is segmented into two paths, the coarse and the fine. Denote the output before the reconstruction LPF as $y(t)$, the output of the coarse and fine modulator is $y_c(t)$ and $y_f(t)$. Δ_c and Δ_f is the interval of quantization for coarse and fine modulator. We can derive that

$$y(t) = 2^{NF} y_c(t) + y_f(t) \tag{5.4}$$

Assume the PSD of the input $X(z)$ is defined as $S_X(f)$, and the sampling frequency is f_s. Similarly, as in Section 3.1.2, as the STF of both the coarse and fine path is unity, the power spectral density of the coarse and fine output are

$$S_{Y_c}(f) = S_{X_c}(f) + | NTF_C(f) |^2 \frac{1}{f_s} \frac{{\Delta_c}^2}{12} \tag{5.5}$$

and

$$S_{Y_f}(f) = S_{X_f}(f) + | NTF_f(f) |^2 \frac{1}{f_s} \frac{{\Delta_f}^2}{12} \tag{5.6}$$

From Eqn. (5.2) and (5.3), we can derive that

$$S_X(f) = 2^{2NF} S_{X_c}(f) + S_{X_f}(f) \tag{5.7}$$

by combining Eqn. (5.4), (5.5), (5.6) and (5.7), the power spectral density of the output can be shown to be

$$S_Y(f) = S_X(f) + 2^{2NF} \, | NTF_C(f) |^2 \frac{1}{f_s} \frac{{\Delta_c}^2}{12} + | NTF_f(f) |^2 \frac{1}{f_s} \frac{{\Delta_f}^2}{12} \tag{5.8}$$

Here $\Delta_c = 2^{NC} - 1 \approx 2^{NC}$ and $\Delta_f = 2^{NF} - 1 \approx 2^{NF}$.

From Eqn. (5.1) and (5.8), the power spectral density of the output can be simplified as

$$S_Y(f) = S_X(f) + | NTF_c(f) |^2 \frac{1}{f_s} \frac{2^{2N}}{12} + | NTF_f(f) |^2 \frac{1}{f_s} \frac{2^{2NF}}{12} \tag{5.9}$$

Therefore, the SNR over the bandwidth of the LPF can be expressed as

$$SNR_{Mod,2-Seg} = \frac{\int_0^{BW} S_X(f)df}{\frac{2^{2NF}}{12f_s}\int_0^{BW} 2^{2NC} \mid NTF_c(f) \mid^2 + \mid NTF_f(f) \mid^2 df} \quad (5.10)$$

As can be seen from Eqn. (5.10), the total noise is associated with both the coarse and fine path. However, if the number of coarse path bit resolution NC is high, we can generally write

$$2^{2NC} \mid NTF_c(f) \mid^2 \gg \mid NTF_f(f) \mid^2 \quad (5.11)$$

Eqn. (5.10) can then be further simplified to

$$SNR_{Mod,2-Seg} \approx \frac{\int_0^{BW} S_X(f)df}{\frac{1}{12f_s}\int_0^{BW} 2^{2N} \mid NTF_c(f) \mid^2 df} \quad (5.12)$$

Denote the input amplitude as A , we could rewrite Eqn. (5.12) as

$$SNR_{Mod,2-Seg} = \frac{6f_s A^2}{2^{2N}\int_0^{BW} \mid NTF_c(f) \mid^2 df} \quad (5.13)$$

For an unsegmented ΔΣ modulator with an input of N bits wide, the quantization interval square would be $\Delta^2 = 2^{2N}$. If the coarse modulator in the two-segment modulator has the same loop filter as the unsegmented modulator, Eqn. (5.13) would be almost identical to Eqn. (3.14), namely

$$SNR_{Mod,2-Seg} \approx SNR_{Mod,UnSeg} \quad (5.14)$$

Eqn. (5.13) implies that in a segmented design, the overall output SNR may remain the same as the unsegmented design, although it may degrade slightly due to the extra noise power introduced by the fine segment.

For the design of noise transfer function, we could simply follow the design procedure as in the unsegmented ΔΣ modulator. The constraint on the NTF can be obtained if the required SNR is given. By simply rearrange Eqn. (5.13), we can impose the design constraint:

$$\int_0^{BW} |\, NTF_c(f) \,|^2 \, df \leq \frac{6 f_s A^2}{2^{2N} \cdot SNR_{2-SegMod,max}} \tag{5.15}$$

As we know that the coarse term is dominant as it as it contributes 2^{2NC} times the power of its fine counterpart. For dominance, assuming a k-times magnitude change

$$|\, NTF_c(f) \,|^2 \geq k \,|\, NTF_f(f) \,|^2 \tag{5.16}$$

where $k \ll 2^{2NC}$.

Now given the sampling frequency and target SNR, by combining Eqn. (5.15) and (5.16), the fine segment constraint is given by

$$\int_0^{BW} |\, NTF_f(f) \,|^2 \, df \leq \frac{6}{k} \frac{f_s A^2}{2^{2N} \cdot SNR_{2-SegMod,max}} \tag{5.17}$$

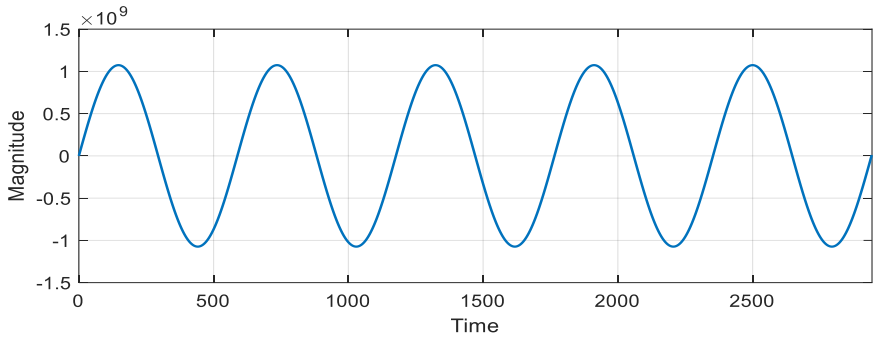
5.2.3 Data Partition Impact

As can be seen in Fig. 5.3, the input signal $x(t)$ is divided into $x_c(t)$ and $x_f(t)$, denoted as the coarse input and the fine input separately. However, the partition of input signal might introduce extra noise into the system and may even cause the system to go unstable. In this subsection, we will investigate the impact of data partition on the operation of two-segment ΔΣ modulators and try to minimize these side effects.

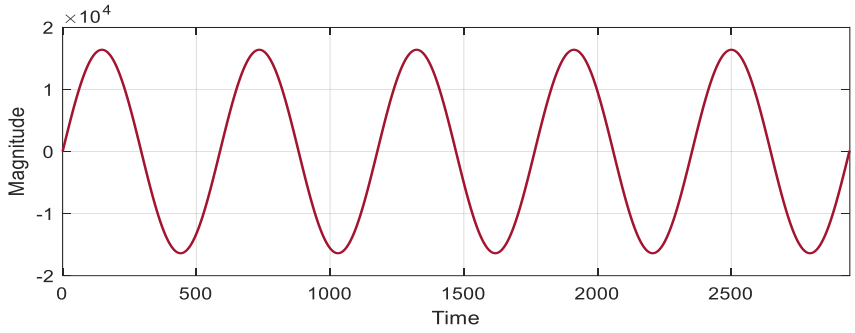
Fig. 5.5 presents an example of data partition simulated in MATLAB/Simulink. The 32-bits digital input signal $x(t)$ is segmented into two 16-bits signal, the coarse input $x_c(t)$ and the fine input $x_f(t)$, as shown in Fig. 5.5. Note that the amplitude of $x(t)$ is set to $A = 2^{29}$. As can be seen, although the coarse path and fine path are both 16 bits wide, the fine path input signal occupies almost 100% of the input range of 16 bits while it's not the case for the

coarse path. Note that this is always the case unless the full-scale signal amplitude can be represented using 16 bits or less.

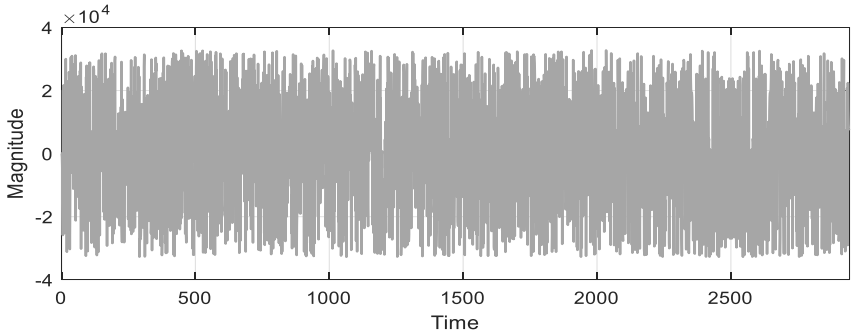
Therefore, the fine modulator needs to be properly designed so that the fine path is stable. According to Eqn. (5.11), the first segment path is the dominant part of the noise contribution. The fine segment noise contribution is 2^{Nc} times less significant than the noise contribution from the coarse path. As a result, a high order modulator in the fine path is unnecessary. Contrarily, a low order modulator should be used to ensure stability.



(a)



(b)



(c)

Fig. 5.4: An example of data partition: (a) full-scale input; (b) coarse path input; (c) fine path input

In the previous study carried out by Romanov [11], at least four or more bits should be allocated to the coarse path for proper operation. Additionally, it was shown [11] that the maximum SNR of the segmented modulator approaches the maximum SNR of the unsegmented modulator as more bits are allocated to the coarse path. Thus, segmentation sacrifices some SNR for the benefit of less resource usage. The hardware cost and the advantages of segmentation will be later discussed in Chapter 7.

5.3 Two-Segment ΔΣ Modulator Simulation

The two-segment ΔΣ modulator is simulated using MATLAB/Simulink in this section. The parameter settings for the simulation are provided in Table. 5.1.

Total bits N	N_c	N_f	A	F_s	f_0	SNR_{max}
32	16	16	2^{N-3}	1 Hz	0.0017	$\geq 80\ dB$

Table. 5.1: Parameter settings for the two-segment modulator

The ΔΣ modulator is designed for a maximum SNR greater than 80 dB. As given in Eqn. (5.14) and (5.15), the previous section's conclusion highlights that the maximum SNR of a two-segment modulator closely approximates that of the unsegmented modulator, with the coarse path primarily determining the overall noise level. By setting the normalized bandwidth to 0.005 Hz, we can employ the same 3rd-order inverse-Chebyshev Noise Transfer Function (NTF) in the coarse path, as previously described in Section 3.3 as

$$H_c(z) = \frac{0.8334z^2 - 1.3533z + 0.5705}{z^3 - 2.9993z^2 + 2.9993z - 1} \tag{5.18}$$

Regarding the fine path filter design, following Eqn. (5.17), we choose the value that $k = 600$, which satisfies $k \ll 2^{2NC}$. This enables us to impose the noise constraint for the fine modulator as

$$\int_0^{BW} | NTF_f(f) |^2 \, df \leq 6.711 \times 10^{-3} \tag{5.19}$$

As we discussed in Section 5.2.3, the loop filter in the fine path should be of first order to ensure stability, i.e.

$$H_f(z) = \frac{1}{z-1} \tag{5.20}$$

The noise transfer function of the fine path is calculated as

$$NTF_f(f) = \frac{z-1}{z} \tag{5.21}$$

The magnitude response of $NTF_f(f)$ is shown in Fig. 5.5. The integral of $|NTF_f(f)|^2$ from zero to the edge of its bandwidth is calculated to be 2.7761×10^{-6}, which meets the constraint calculated in Eqn. (5.19).

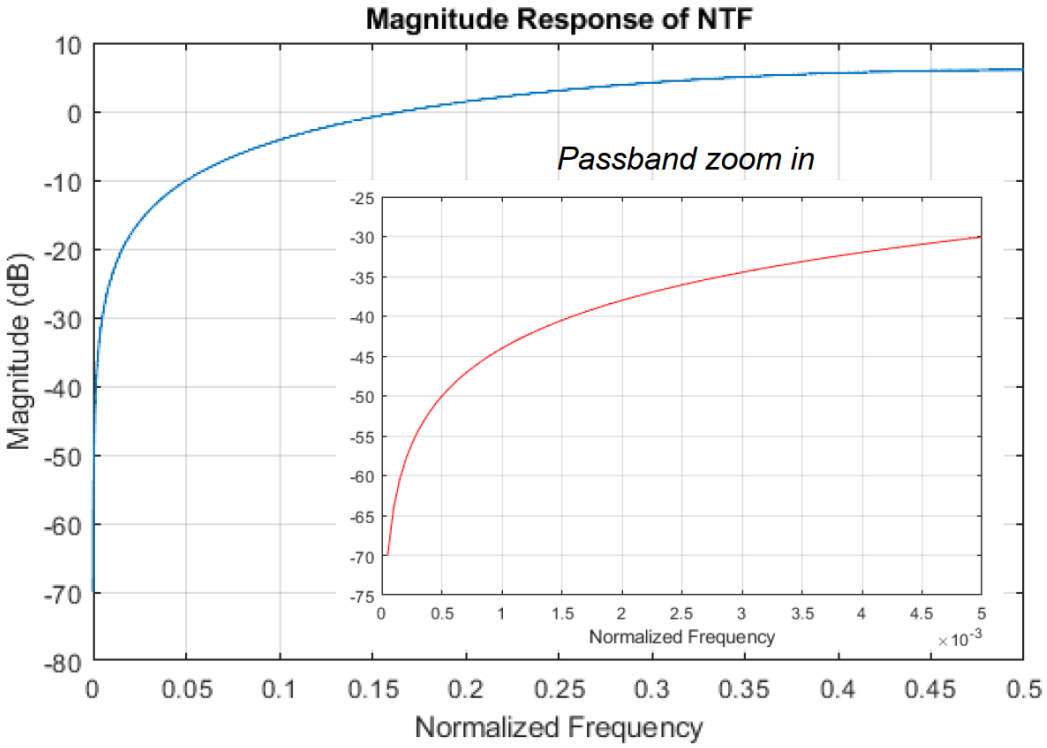
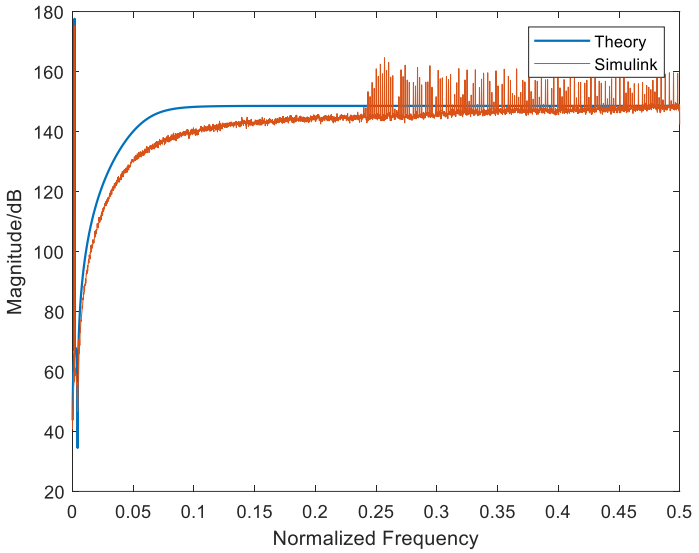
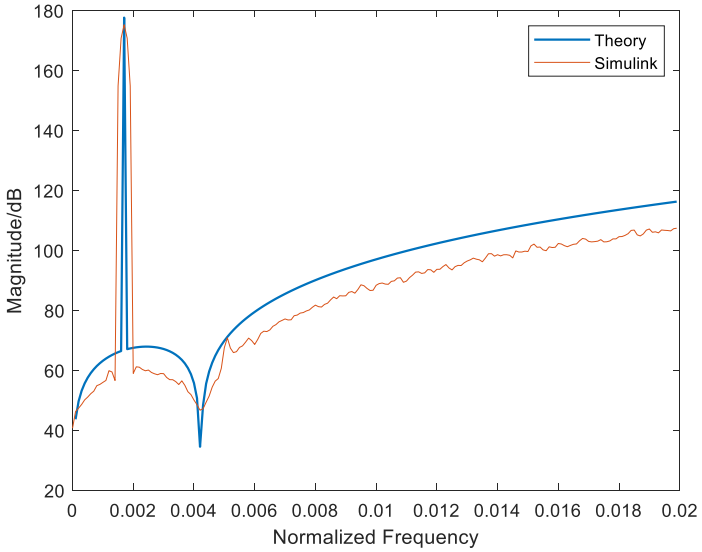


Fig. 5.5: Magnitude response of the fine path noise transfer function

By using the coarse filter described in Eqn. (5.18) and the fine filter described in Eqn. (5.20), the PSD results of the output before the LPF predicted by theory and that generated by a MATLAB/Simulink simulation are shown in Fig. 5.6. Note that the resulting output was processed using the Dolph–Chebyshev window, which is the same window as we used in Chapter 3.



(a)



(b)

Fig. 5.6: Theoretical and Simulink simulation output PSD of two-segment ΔΣ modulator
(a): whole frequency (b): in-band frequency

For the two-segment simulation, the SNR is calculated to be 96.05 dB theoretically and the Simulink simulation SNR is measured at 96.56 dB. Since we are using the same NTF for the coarse path as we did for single path modulator in Section 3.3, one can compare the SNR results of the unsegmented ΔΣ modulator to the two-segment ΔΣ modulator. This is done in Table. 5.2. We could see that by using a two-segment, the overall SNR of the ΔΣ modulator dropped slightly. Fortunately, this decrease can be tolerated in most cases.

	Theoretical	**Simulink**
Unsegment	96.1592 dB	96.8973 dB
two-segment	96.0508 dB	96.5575 dB

Table. 5.2: SNR results of unsegmented ΔΣ modulator and two-segment ΔΣ modulator

5.4 Summary

This chapter delves into the design and simulation of segmented ΔΣ modulators. It begins by introducing the segmentation concept and its background. The chapter then dissects two-segmented ΔΣ modulators, outlining the structure and noise constraints equations. Finally, the chapter concludes with MATLAB/Simulink simulation of a two-segment ΔΣ modulators and compared to a single path ΔΣ modulator. The results were comparable. This chapter offers a comprehensive understanding of the segmented ΔΣ modulator design and simulation, which provides a foundation for the segmented ΔΣ oscillators of the next chapter.

Chapter 6: Segmented ΔΣ Oscillators

In this chapter, the oscillator circuit established in Chapter 4 is repurposed with the aim of producing highly accurate analog sinusoidal signals using the segmented ΔΣ modulator developed in Chapter 5. The chapter begins by explaining the motivation and principle of the segmented ΔΣ oscillator design. Following that, the circuit's structure is presented, followed by a detailed MATLAB/Simulink simulation.

6.1 Segmented ΔΣ Oscillator Design

The analog signal source plays a vital part in a mixed-signal BIST scheme. Chapter 4 presents the design and simulation of ΔΣ oscillators. To further enhance the high-resolution performance and efficiency of an oscillator, we will map the two-segment ΔΣ modulator into the ΔΣ oscillator design and test its feasibility.

6.1.1 Segmented ΔΣ Oscillator Structure

To further reduce the hardware resources use of the ΔΣ oscillator, the goal of this chapter is to replace the unsegmented modulator with a two-segment modulator realization in the ΔΣ oscillator design. Thus, we can derive the structure of the segmented ΔΣ Oscillator in Fig. 6.1. For the general case of two-segment modulator, the subsystem of the segmented modulator is shown in Fig. 6.2.

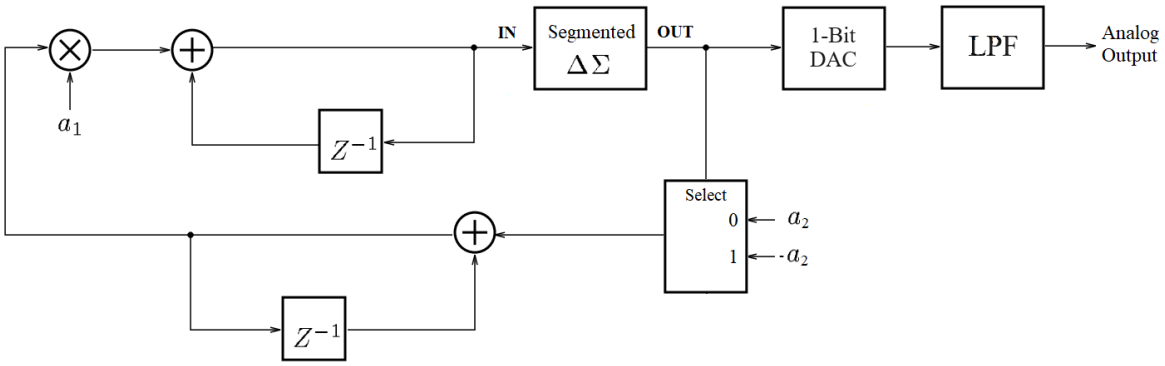


Fig. 6.1: General block diagram of a segmented ΔΣ oscillator

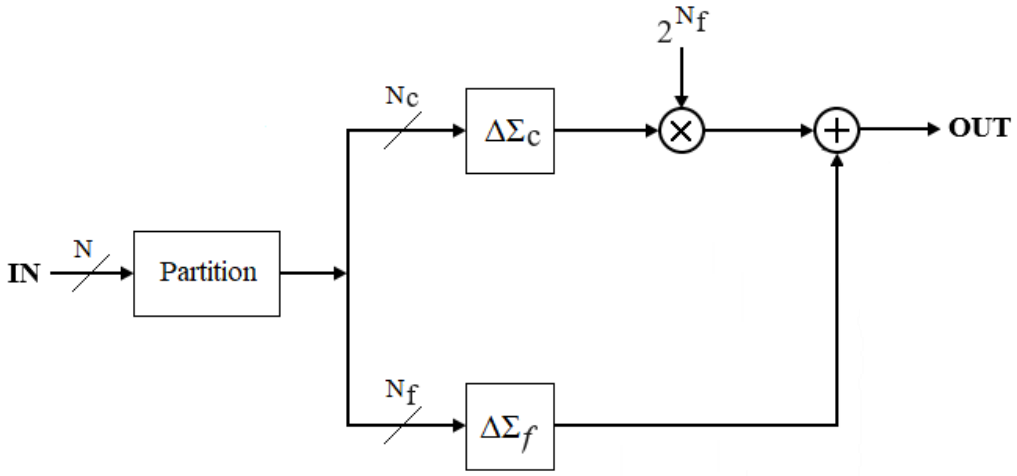


Fig. 6.2: Block diagram of a two-segment ΔΣ modulator

Since the modulator does not affect the oscillation amplitude and frequency, the output sinewave takes on the general format:

$$x_1(n) = A\sin(\omega_o nT + \phi) \tag{6.1}$$

In consideration of the fractional numbers' implementation in the FPGA, the amplitude of the output signal, denoted as OUT in Fig. 6.1 is adjusted to $\Delta/2$, where Δ is the quantization interval, equal to 2^N.

Thus, the oscillation amplitude, frequency and phase are given as

$$f_o = f_s \cos^{-1}\left(1 - \frac{a_1 a_2}{\Delta}\right) \qquad \text{for } 0 < \frac{a_1 a_2}{\Delta} \leq 4 \tag{6.2}$$

and

$$A = \sqrt{x_1(0)^2 + \frac{(\left(1 - \frac{2a_1a_2}{\Delta} - \cos(2\pi f_o T)\right) x_1(0) + a_1 x_2(0))^2}{\sin^2(2\pi f_o T)}} \tag{6.3}$$

The initial phase is presented as

$$\phi = \sin^{-1} \frac{x_1(0)}{A} \tag{6.4}$$

6.1.2 Noise Constraints

The noise constraints on a segmented ΔΣ oscillator could be combined with the constraints we described in Chapter 4 and Chapter 5. Given a desired SNR, the design process is the same as we described in Section 5.3.2.

As concluded in Section 4.1.3, the dominant source of noise at the output of the oscillator is the quantization error originating from the ΔΣ modulator. Therefore, for a two-segment ΔΣ oscillator within N bits number system, with the coarse path having NC bits and the fine path having NF bits, where

$$N = NC + NF \tag{6.5}$$

The noise power of coarse path is given as

$$\int_0^{BW} | NTF_c(f) |^2 \, df \approx \frac{12 f_s \int_0^{BW} S_X(f) df}{2^{2N} SNR_{Osc,2-Seg,max}} \tag{6.6}$$

And noise power associated with the fine path is given as

$$| NTF_f(f) |^2 \, df \leq \frac{1}{k} | NTF_c(f) |^2 \tag{6.7}$$

where $k \ll 2^{2NC}$.

The loop filter transfer function is given as

$$H_L(z) = \frac{1}{NTF_L(z)} - 1 \qquad L \in \{c, f\} \tag{6.8}$$

Since there is more than one signal path in the modulator, the loop filter design should follow Eqn. (6.6), (6.7) and (6.8) as described above given a desired SNR. To realize a specific oscillation frequency f_o and amplitude A, we need to properly choose the parameters. The unknown parameters are $f_s, a_1, a_2, x_1(0), x_2(0)$. Based on Eqn. (6.2), (6.3), we now have 5 unknown parameters and 2 equations, which result in a degree of design freedom of 3 to realize the oscillator.

6.2 Segmented ΔΣ Oscillator Simulation

In this section, we follow the same simulation procedure as we did in the previous chapters. To maintain coherency, we will use the two-segment ΔΣ modulator as we did in Section 5.4.2 inside of the oscillator.

Specifically, the diagram of the two-segment ΔΣ modulator, which is the subsystem described in Fig. 6.1 is presented in Fig. 6.3.

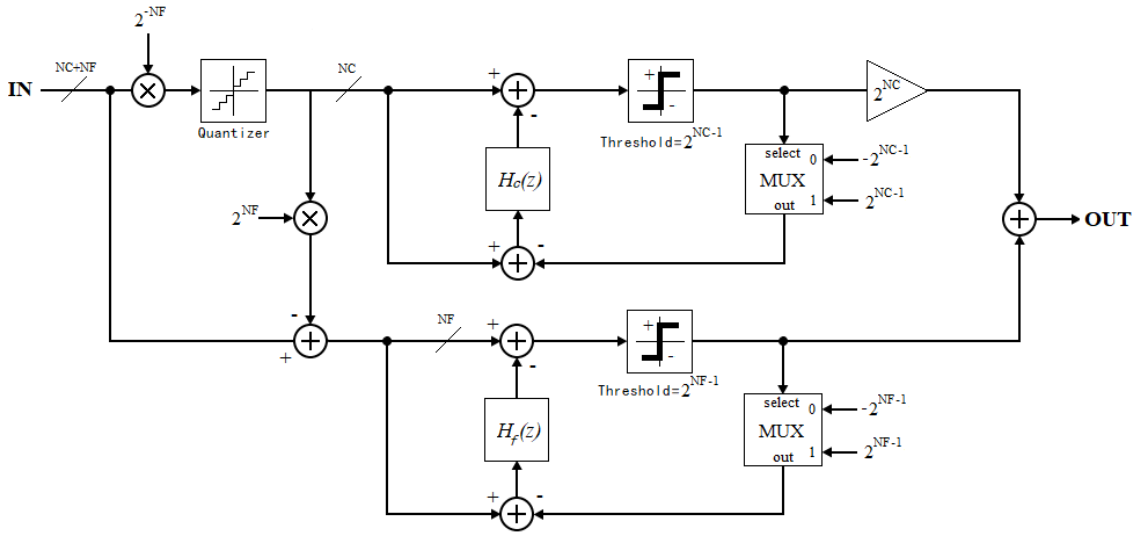


Fig. 6.3: Block diagram of two-segment ΔΣ modulator

Assuming 32-bit signed integer number precision, the system is simulated using Matlab/Simulink. The SNR requirement is the same as we desired in Section 5.4, which is made greater than 80 dB. Assume the design specification is the same as we performed in Section 4.2, the two-segment ΔΣ oscillator needs to generate a sinewave signal with an amplitude of $A = 5181.5$ and oscillation frequency of $f_o = 7.8641$ Hz. The parameters we designed to meet the amplitude and frequency are shown in Table. 6.1.

Required specifications	$\boldsymbol{A}$	$\boldsymbol{f_o}$	$\boldsymbol{SNR}$	
	5181.5	7.8641 Hz	$\geq 80dB$	
Parameters choose	$\boldsymbol{a_1}$	$\boldsymbol{NF = NC}$	$\boldsymbol{F_s}$	$\boldsymbol{x_1(0)}$
	1	8	2000 Hz	128
Parameters calculated	$\boldsymbol{a_2}$	$\boldsymbol{x_2(0)}$	$\int_0^{BW} (2^{2N} \mid NTF_c(f) \mid^2 + 2^{2NF} \mid NTF_c(f) \mid^2) df$	
	20	128	$\leq 1.1070 \times 10^{11}$	

Table. 6.1: Parameters of two-segment ΔΣ oscillator under 16-bits number system

The loop filter in the fine path should be of low order to ensure stability. We reuse the first-order filter for the fine path in Chapter 5, represented as

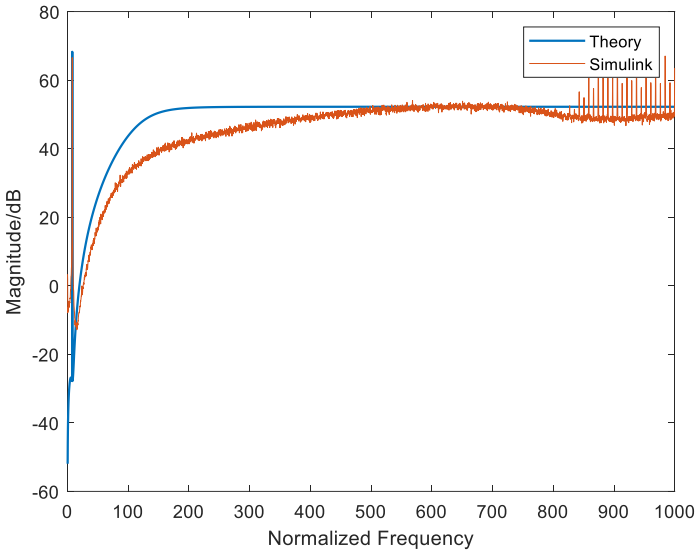
$$H_f(z) = \frac{1}{z-1} \tag{6.9}$$

As for the coarse filter, we reuse the 3rd inverse-Chebyshev filter in Chapter 3 as the coarse NTF. The transfer function of the coarse loop filter is given as

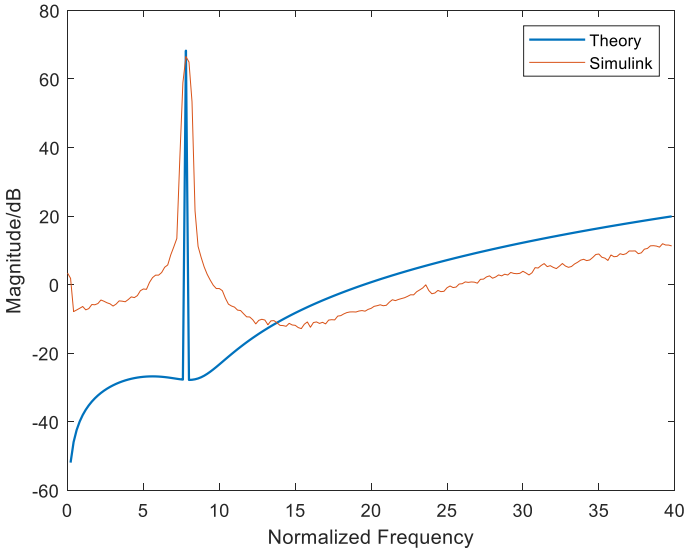
$$H_c(z) = \frac{0.8334z^2 - 1.3533z + 0.5705}{z^3 - 2.9993z^2 + 2.9993z - 1} \tag{6.10}$$

By using the parameters in Table. 6.1, the fine filter in Eqn. (6.9), and the coarse filter in Eqn. (6.10), the PSD results of the output before the LPF predicted

by theory and that generated by a MATLAB/Simulink simulation are depicted in Fig. 6.5. Note that the resulting output was processed using the Dolph–Chebyshev window. As we can see, the noise level of the simulation results remains high. One way to suppress the noise would be to increase the number of bits we use, to generate a signal with the same frequency but a higher amplitude.



(a)



(b)

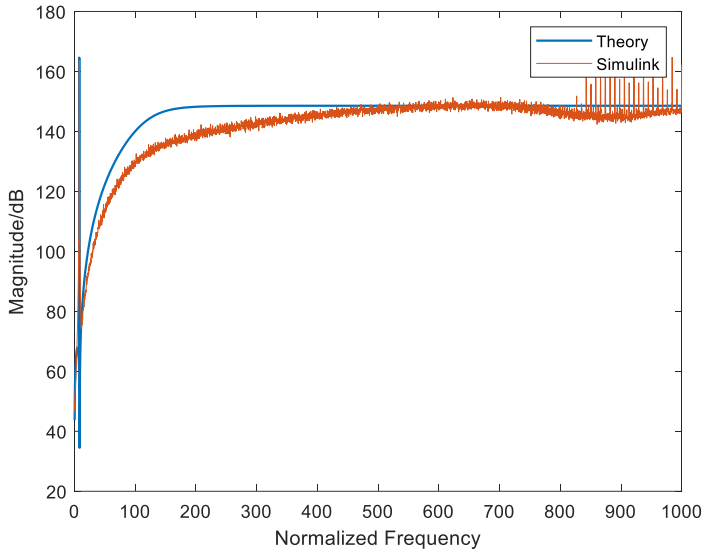
Fig. 6.4: Theoretical and Simulink simulation output PSD of two-segment ΔΣ oscillator under 16 bits number system (a): Nyquist frequency (b): in-band frequency

To improve the noise shape and bring down the SNR, we increase the number system to 32 bits. This requires that $NF = NC = 16$. Meanwhile, to maintain coherency, the new design has the same oscillation frequency as the previous one. The parameters for the two-segment oscillator under 32-bits number system are shown in Table. 6.2. Note that the oscillation amplitude A may vary, but the final output signal amplitude can be adjusted with analog signal processing (using proper LPF).

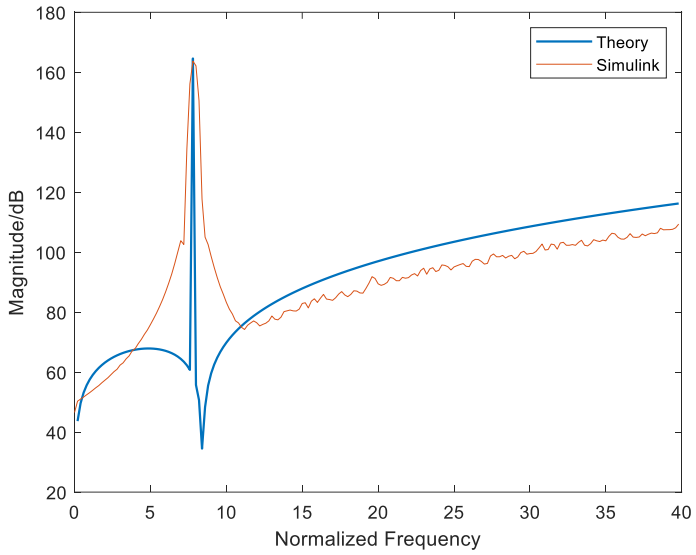
Required specifications	$\boldsymbol{NF = NC}$	$\boldsymbol{f_o}$	$\boldsymbol{SNR}$	
	16	7.8641 Hz	$\geq 80dB$	
Parameters choose	$\boldsymbol{a_1}$	$\boldsymbol{x_2(0)}$	$\boldsymbol{F_s}$	$\boldsymbol{x_1(0)}$
	1	8	2000 Hz	128
Parameters calculated	$\boldsymbol{a_2}$	$\boldsymbol{A}$	$\int_0^{BW} (2^{2N} \mid NTF_c(f) \mid^2 + 2^{2NF} \mid NTF_c(f) \mid^2) df$	
	1310720	3.3957×10^8	$\leq 1.1070 \times 10^{11}$	

Table. 6.2: Parameters of two-segment ΔΣ oscillator under 32-bits number system

By using the parameters provided in Table. 6.2, we repeat the simulation with the fine filter described in Eqn. (6.9), and the coarse filter described in Eqn. (6.10). Fig. 6.6 provides the PSD results of the output before the LPF predicted by theory and that generated by a MATLAB/Simulink simulation under the 32 bits number system.



(a)



(b)

Fig. 6.5: Theoretical and Simulink simulation output PSD of two-segment ΔΣ oscillator under 32 bits system (a): Nyquist frequency (b): in-band frequency

Due to more bits being used in the new system, the simulation result achieves a better noise shaping. The system SNR under different circumstances are displayed in Table. 6.3. We could see the two-segment ΔΣ oscillator under 32 bits system has an SNR of 81.62 dB, which meets the SNR requirement.

Comparing the result with the single path simulation as shown in Table. 5.3, it can be noted that there is a slight drop of SNR for the two-segment oscillator comparing to the single path ΔΣ oscillator. This is due to the introduction of fine segments. The oscillation frequency of the simulation results closely aligns with that predicted by theory.

	SNR	f_o
Theory of two-segment	91.0075 dB	7.8641 Hz
Two-segment (Total: 16-bits)	58.1208 dB	7.8641 Hz
Two-segment (Total: 32-bits)	81.6227 dB	7.8641 Hz

Table. 6.3: System SNR and oscillation frequency under different circumstances

6.3 Summary

This chapter explores the design and simulation of segmented ΔΣ oscillators. The segmented ΔΣ oscillators design process is explained first and then a MATLAB/Simulink simulation is conducted. Simulation results indicate that using the same parameter settings, the two-segment ΔΣ oscillator produces a signal with the same frequency as the single path oscillator. Furthermore, the SNR of the two-segment ΔΣ oscillator simulation is higher when more bits are used in the number system. By using a 32 bits number system, the two-segment ΔΣ oscillator simulation meets the maximum SNR requirements.

Chapter 7: Experimental Validation

In the previous chapters, we established the theoretical foundations of ΔΣ modulators and ΔΣ oscillators, along with the introduction of segmentation into ΔΣ modulators and ΔΣ oscillators. Furthermore, we developed the necessary design principles crucial for their implementation, followed by building them in MATLAB/Simulink and conducting the simulations to evaluate their performance.

In this chapter, we will map the previous simulation examples into hardware implementation to validate the previous findings and further enhance the principles of the designs.

Finally, we will evaluate and compare the hardware costs associated with the design, considering both segmented and non-segmented configurations.

7.1 FPGA setup

7.1.1 Device Setup

Synthesizing an HDL description of a circuit to a Field-Programmable-Gate-Array (FPGA) can enable prototyping, where the register lengths can be adjusted until a prototype with minimal hardware and desired SNR performance is achieved.

In this chapter, an ALTERA DE1 Board is utilized [25]. Verilog was chosen as the HDL description language. The Verilog code for each module to implement the modulator is provided in the Appendix.

In ΔΣ modulators and ΔΣ oscillators, the output of the modulator, namely

the 1-bit data stream contains all the information of the analog signal. They were sampled using DPO 3032 Digital Phosphor oscilloscope. The data collected is then processed using Matlab to compare with the results from simulation in the previous chapters. Finally, for the ΔΣ oscillator implementation, the sinewave that the oscillator generates is displayed by running the testbench file in Quartus II, which is also provided in Appendix.

7.1.2 Number System Setup

FPGA configuration utilizes the binary number system, which uses two symbols, 0 and 1, to represent numerical values. For this thesis, we built up four different systems. First, we studied the ΔΣ modulators and ΔΣ modulators with segmentation. Then we switched the topic to ΔΣ oscillator then studied the behavior of ΔΣ oscillators with segmentation. For ΔΣ modulators and ΔΣ oscillators, the output of the modulator contains all the information of the input signal, which is a 1-bit word wide signal. Thus, we use one pin on the FPGA to represent it.

In the previous study covered by Romanov [11], a 1-bit two-segment ΔΣ DAC was implemented in FPGA. It employed an unsigned digital input within the range of 0 to $2^N - 1$ and the system was built using the unsigned number system. However, when it comes to the oscillator implementation, the input signal generated by the oscillator ranges from -2^{N-1} to 2^{N-1}. This poses a challenge for our design. If we want to apply the use of segmented ΔΣ modulator into an oscillator, we could either change the design of the oscillator by introducing a DC bias or build the implementation based on signed number system. The former approach would involve massive mathematical calculations

for the oscillators. As a result, we opted for the implementation of segmented ΔΣ oscillators under a signed number system.

Since the loop filter inside of the modulator is based on floating point numbers, we must represent them properly in an FPGA. Every floating number can be divided into the integer part and fractional part. The integer part can be easily represented using a binary number system and the arithmetic operation simply follows the rules of integer number arithmetic operations. However, the arithmetic operation of fractional part could cause overflow and therefore needs to be carefully taken care of, which will be discussed in the next subsection. Moreover, a slight error in the fractional part could bring a catastrophic effect on the modulator's output.

Take the ΔΣ modulator in Fig. 3.4 as an example. Assume there are 32 bits available for use. The input sinewave amplitude is set to 2^8 and the input frequency is 0.0017 Hz. The loop filter transfer function is given as

$$H(z) = \frac{0.8334z^2 - 1.3533z + 0.5705}{z^3 - 2.9993z^2 + 2.9993z - 1} \tag{7.1}$$

If there are 4 bits allocated for the fractional part representation, the loop filter transfer function will be represented as

$$H_{4-bit}(z) = \frac{\frac{13}{16}z^2 - (1 + \frac{5}{16})z + \frac{9}{16}}{z^3 - (2 + \frac{15}{16})z^2 + (2 + \frac{15}{16})z - 1} \tag{7.2}$$

As we can see in Eqn. (7.2), the fractional part representation is a bit different from the floating representation in Eqn. (7.1). The output power spectral density of ideal realization (Theory, Simulink) and 4 bits allocated for fractional part representation (Theory, Simulink, FPGA implementation) is plotted in Fig. 7.1. Note that the output of the modulator is scaled up by 2^{16}.

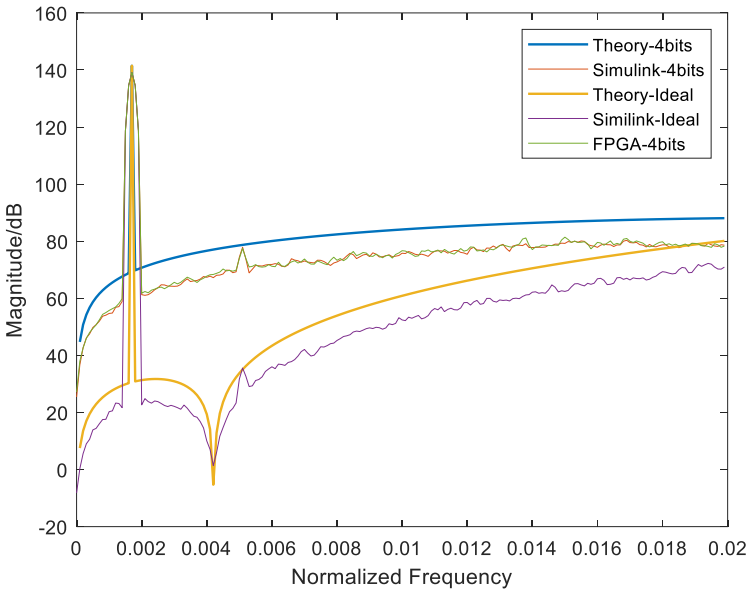


Fig. 7.1: Ideal (Theory, Simulink) and 4-bits fractional part realization (Theory, Simulink, FPGA implementation) of output PSD for a single path ΔΣ modulator

There are a few things we could conclude from Fig. 7.1. The FPGA result correlates well with the Simulink simulation. However, only 4 bit allocation for the fractional part significantly increases the in-band noise level by about 40 dB. Therefore, more bit precision is needed to achieve a higher SNR.

Similarly, if there are 16 bits allocated for fractional part representation, the loop filter transfer function will be represented as

$$H_{16-bit}(z) = \frac{\frac{54617}{65536}z^2 - (1 + \frac{23153}{65536})z + \frac{37388}{65536}}{z^3 - (2 + \frac{65490}{65536})z^2 + (2 + \frac{65490}{65536})z - 1} \tag{7.3}$$

As can be seen in Eqn. (7.3), the fractional part representation using 16 bits is close to the floating representation shown in Eqn. (7.1). The output PSD of ideal realization (Theory, Simulink) and 16 bits allocated for fractional part representation (Theory, Simulink, FPGA implementation) is shown in Fig. 7.2.

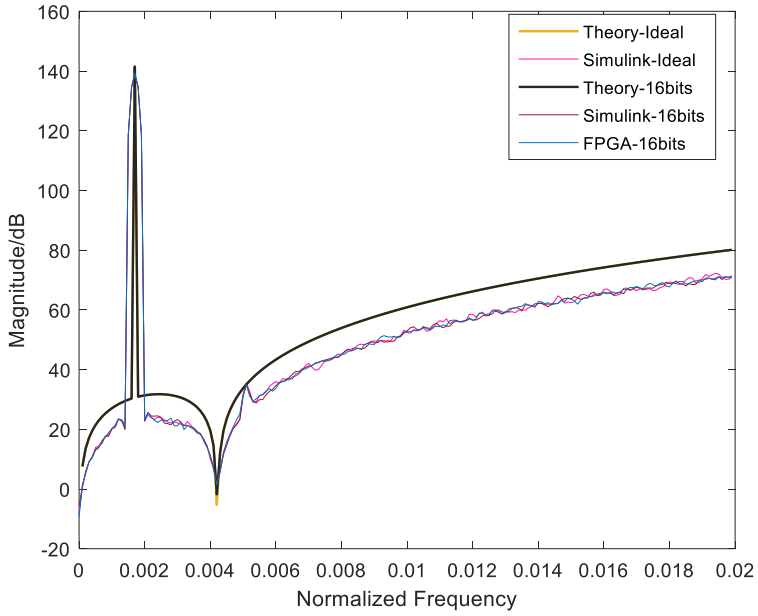


Fig. 7.2: Ideal (Theory, Simulink) and 16-bits fractional part realization (Theory, Simulink, FPGA implementation) of output PSD for a single path ΔΣ modulator

Here, we can see with 16-bits precision for fractional number gives a PSD curve close to floating point resolution. With this result in mind, the systems implemented in FPGA will adopt 16 bits-precision for the fractional numbers, which will be further discussed in later sections. Take the parameter 2.9993 in the denominator in Eqn. (7.1) as an example. The binary number representation can be shown in Fig. 7.3.

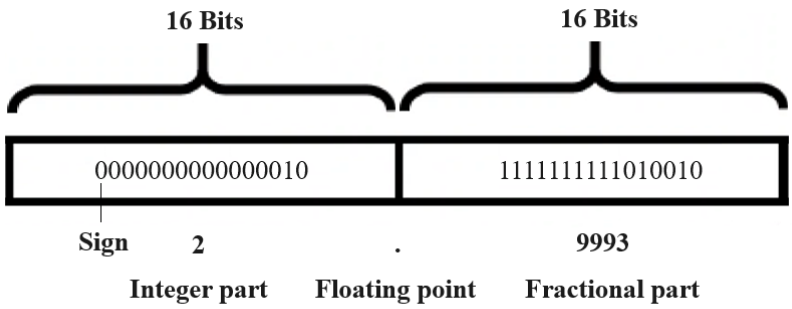


Fig. 7.3: Number representation in FPGA format

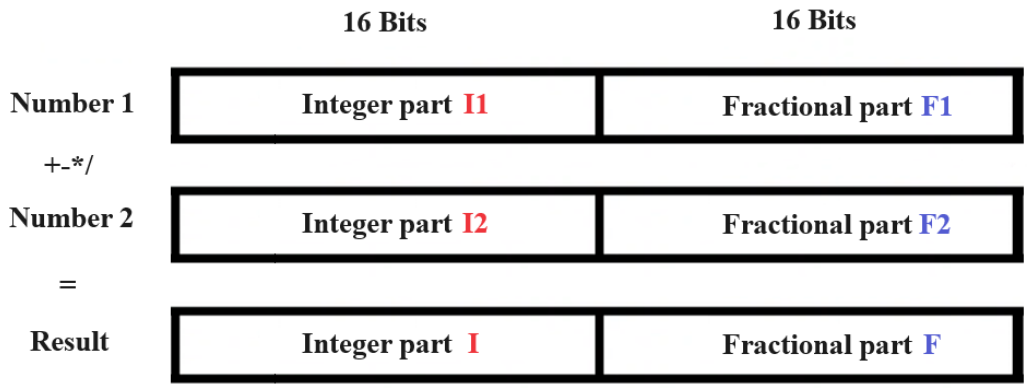


Fig. 7.4: Arithmetic operation structure

7.1.3 Arithmetic Operation

As interpreted in Fig. 7.3, every number we mapped to FPGA has an integer part and a fractional part, which makes the arithmetic operation different from that in decimal number system.

The arithmetic operation includes addition, subtraction, multiplication, and division. Assume we are doing one operation on *Number 1* and *Number 2*, and the output is *Result*. Fig. 7.4 shows the structure of this operation while using symbols to demonstrate each number. *I1, I2* and *I* represent the integer part of *Number 1, Number 2* and *Result*, respectively. Contrarily, *F1, F2* and *F* denote the fractional part of *Number 1, Number 2* and *Result*, respectively. Generally, one can see a number as the combination of the integer and fractional part. For example, the result can be written as $Result = (IF)$.

For addition and subtraction, the operation could be performed separately. The fractional part of the result can be simply calculated as

$$F_{\pm} = F1 \pm F2 \tag{7.4}$$

The integer part of the result therefore is given as

$$I_{\pm} = \begin{cases} I1 \pm I2, \ if\ no\ overflow\ in\ fraction\ part \\ I1 \pm I2 \pm 1, if\ overflow\ in\ fractional\ part \end{cases} \quad (7.5)$$

where the result of addition or subtraction can be denoted as

$$Result_{\pm} = (IF)_{\pm} \quad (7.6)$$

When it comes to division, *Number 2* is always an integer, which would simplify our calculation. However, different from addition or subtraction, division involves 32-bits calculation. One can use *Number 1* as a 32-bits number as a whole. *Number 2* is a 16-bits number while the result shall be a 32-bits number with both the integer part and fractional part. The division *Result* can be calculated as

$$Result_{\div} = (IF)_{\div} = \lfloor (I1F1)_{32bits} \div (00000000000000I2)_{32bits} \rfloor \quad (7.7)$$

where $\lfloor w \rfloor$ is the function that takes a real number w as input and gives the greatest integer less than or equal to w as output.

Multiplication is the most complicated arithmetic operation and could easily cause register overflow which needs to be taken more care of. *Number 1* times *Number 2* can be simply written as

$$I1 \times I2 + I1 \times F2 + F1 \times I2 + F1 \times F2 \quad (7.8)$$

The first term $I1 \times I2$ is a 16-bits times 16-bits integer multiplication. The result should also be a 16-bits integer which only contribute to the integer part of Result. Thus, only one 16-bits register is needed.

The last term $F1 \times F2$ is a 16-bits times 16-bits number multiplication, the result is a 32-bits number which only contribute to the fractional part of the multiplication *Result*. Both $F1$ and $F2$ are fractional numbers, however, the result of this term should be a 16-bits fractional number. Thus, we need one 32-

bits register to store this term. By right shifting 16 bits, we could obtain the result of $F1 \times F2$. Note that the redundant terms after shifting will be discarded due to 16-bits precision. Denote this result as FF, one can write that

$$FF = \lfloor (F1 \times F2)_{32bits} / 2^{16} \rfloor \tag{7.9}$$

The outcome of two terms in the middle involve both integers and fractional parts. Therefore, we need two 32-bits registers to store the value of $I1 \times F2$ and $F1 \times I2$. The first 16 bits of $I1 \times F2$ and $F1 \times I2$ are integer results while the later 16 bits are fractional results.

By adding the integer parts and fractional parts individually of the four terms, we can derive the integer $I_\times$ and fractional part $F_\times$ of *Result*, which is the result of *Number 1* times *Number 2*

$$F_\times = FF + \lfloor (I1 \times F2) \rfloor_{32bits}[15:0] + \lfloor (I1 \times F2) \rfloor_{32bits}[15:0] \tag{7.10}$$

and

$$I_\times = \lfloor (I1 \times I2) \rfloor_{16bits} + \lfloor (I1 \times F2) \rfloor_{32bits}[31:16] + \lfloor (I1 \times F2) \rfloor_{32bits}[31:16] \tag{7.11}$$

the result is given as

$$Result_\times = (IF)_\times \tag{7.12}$$

The arithmetic operation rules described above would be strictly performed under 32 bits number system setup in FPGA. All the implementation in the later sections utilize this setup.

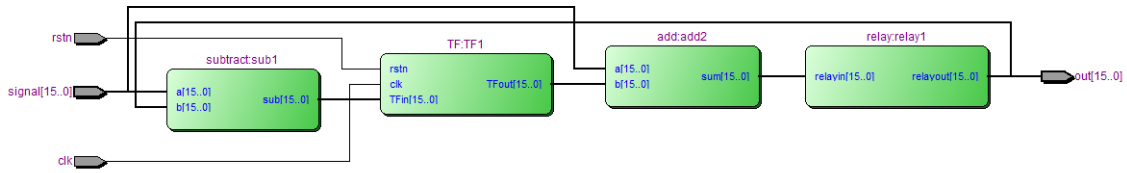


Fig. 7.5: Schematic view of the 1-bit modulator provided by QuartusII

7.2 ΔΣ Modulator implementation

Firstly, we map the 1-bit ΔΣ modulator in Chapter 3.4 into hardware. As can be seen from Fig. 3.5, the structure of a 1-bit unsigned ΔΣ modulator consist of two adders, the transfer function module and a relay module. The schematic view of the 1-bit modulator provided by QuartusII is shown in Fig. 7.5.

To avoid quantization error caused by the input sinewave, the input for the modulator implemented with FPGA is directly taken from Simulink. By building up the arithmetic operation in Section 7.1.3, the in-band PSD of the modulator output of the FPGA implementation and Simulink simulation are displayed in Fig. 7.2. As we are using 16 bits precision for fractional number representation, it is evident that the modulator output generated from FPGA strongly matches the one from MATLAB/Simulink generation.

Given the same signal amplitude and frequency, we further moved forward to two-segment ΔΣ modulator implementation. To maintain coherency in Section 5.4, we use the same parameters we used there as the ones in the FPGA. The schematic view of the two-segment ΔΣ modulator provided by QuartusII is shown in Fig. 7.6. Note that the coarsemod and finemod module is reusing the single path ΔΣ modulator we built up in Fig. 7.5.

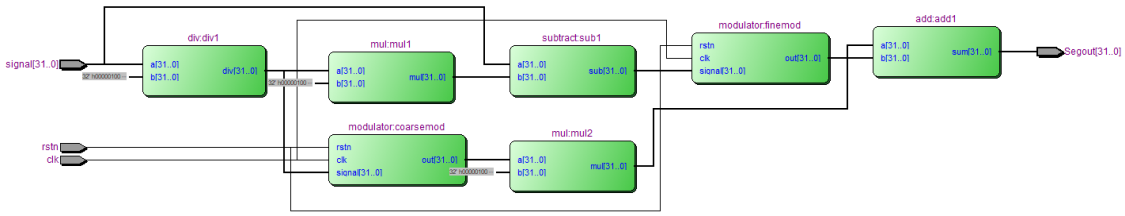


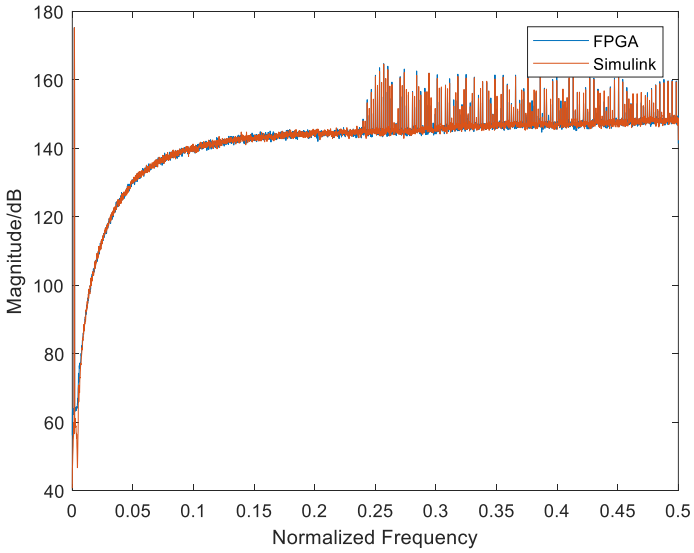
Fig. 7.6: Schematic view of the two-segment ΔΣ modulator provided by QuartusII

By using the third order inverse-Chebyshev filter given in Eqn. (7.3) as the coarse filter and the first order filter given in Eqn. (6.9) as the fine filter, the output PSD of the FPGA implementation and Simulink simulation are shown in Fig. 7.7.

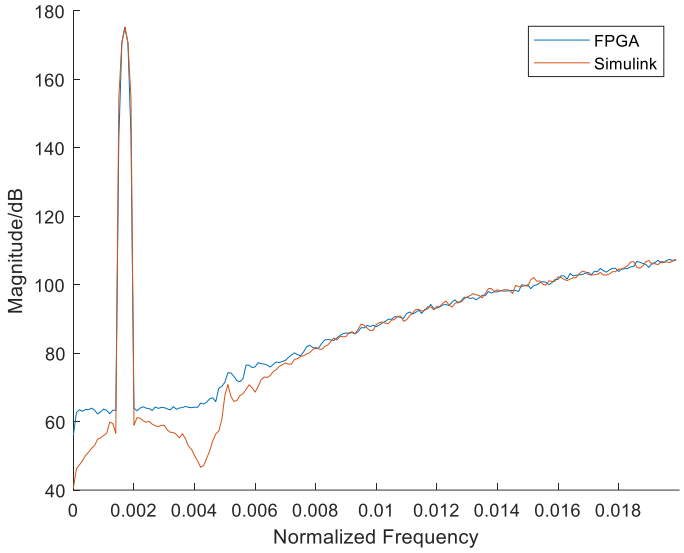
For this given input signal with an amplitude of $A = 2^8$ and a frequency of 0.0017 Hz, the SNR generated based on theory, Simulink, and FPGA are shown in Table. 7.1. We could see we meet the SNR requirements by using the 3rd order inverse Chebyshev filter as the noise transfer function despite a slight drop.

	SNR
Theoretical	96.0508 dB
Simulink	96.5575 dB
FPGA	90.0379 dB

Table. 7.1: Theory, Simulink and FPGA results of SNR of the two-segment ΔΣ modulator



(a)



(b)

Fig. 7.7: Output PSD of Simulink simulation and FPGA implementation of two-segment ΔΣ modulator (a): Nyquist frequency (b): in-band frequency

7.3 ΔΣ Oscillator Implementation

7.3.1 Single Path ΔΣ Oscillator

In this section, we move on to map the ΔΣ oscillator without segmentation into the FPGA. All the parameters in Table. 4.2 are reused.

Followed by the ΔΣ oscillator structure depicted in Fig. 4.3, we build it up in FPGA. The schematic view of the single path ΔΣ oscillator provided by QuartusII is shown in Fig. 7.8. We collect two outputs out of the system, one is the modulator output which is declared as *out*, the other is the sinewave the oscillator generates, declared as $x1$.

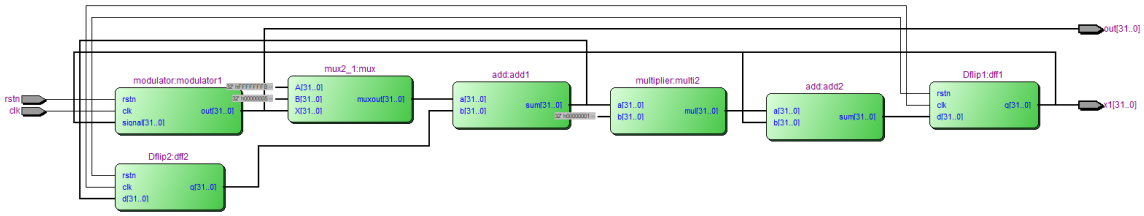
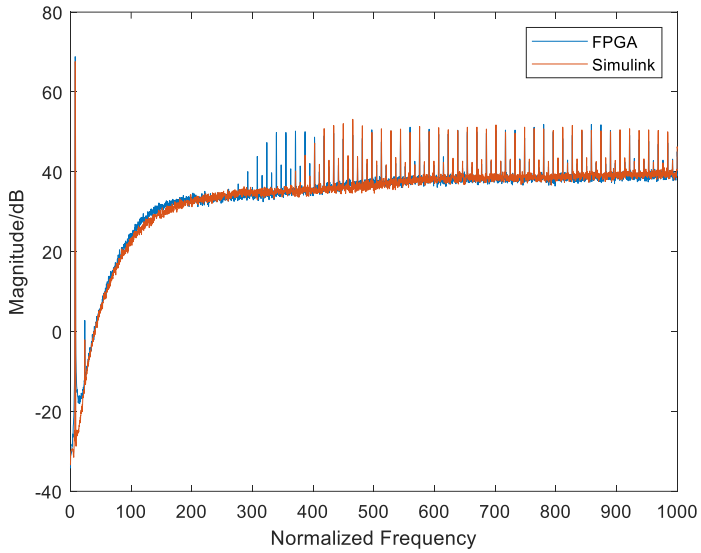
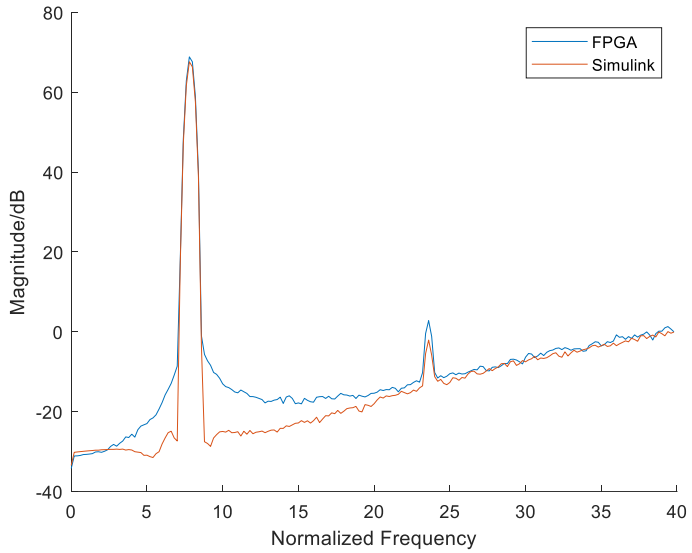


Fig. 7.8: Schematic view of the single path ΔΣ oscillator provided by QuartusII

By using the third order inverse-Chebyshev filter given in Eqn. (7.3) as the loop filter, the output PSD in Simulink simulation and the FPGA implementation is shown in Fig. 7.9.



(a)



(b)

Fig. 7.9: Output PSD of Simulink simulation and FPGA implementation of the single path ΔΣ oscillator (a): Nyquist frequency (b): in-band frequency

Note that $x1$ is the output of the first register. The Simulink simulation result and the testbench file result from FPGA implementation of $x1$ are plotted in Fig. 7.10. It is evident that the signal is a sinewave and the testbench file result from FPGA implementation closely aligns with that of MATLAB/Simulink simulation.

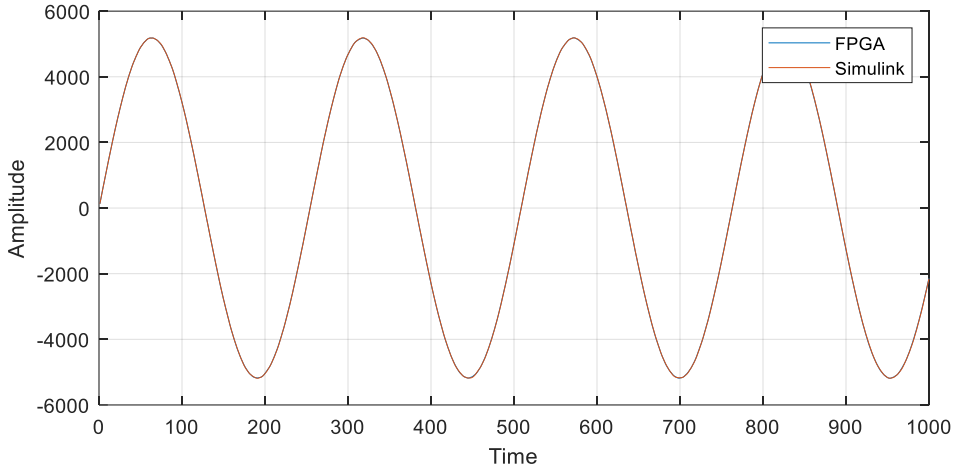


Fig. 7.10: Simulink and FPGA sinewave output of the single path ΔΣ oscillator

As seen from Fig. 7.9, the noise shape of the FPGA implementation closely matches with the MATLAB/Simulink simulation. Table 7.2 presents the amplitude, frequency, and system SNR results derived from theory, Simulink, and FPGA based on the modulator's output. We could see that the FPGA implementation demonstrates the generation of a sinewave with a frequency matching that of the Simulink simulation and theoretical expectations. The amplitude closely approximates the values observed in Simulink and theory, with only a minor decrease in SNR, which still meets our specified requirements.

	A	f_o	SNR
Theoretical	5181.5	7.8641 Hz	91.2114 dB
Simulink	5158.6	7.8641 Hz	86.9023 dB
FPGA	5259.3	7.8641 Hz	84.0895 dB

Table. 7.2: Theory, Simulink and FPGA results of oscillation frequency, amplitude and system SNR of the one-path ΔΣ Oscillator

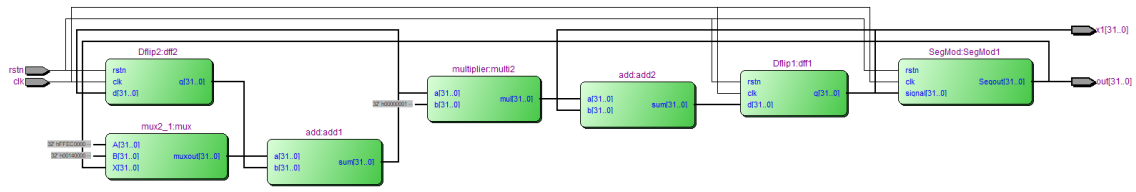
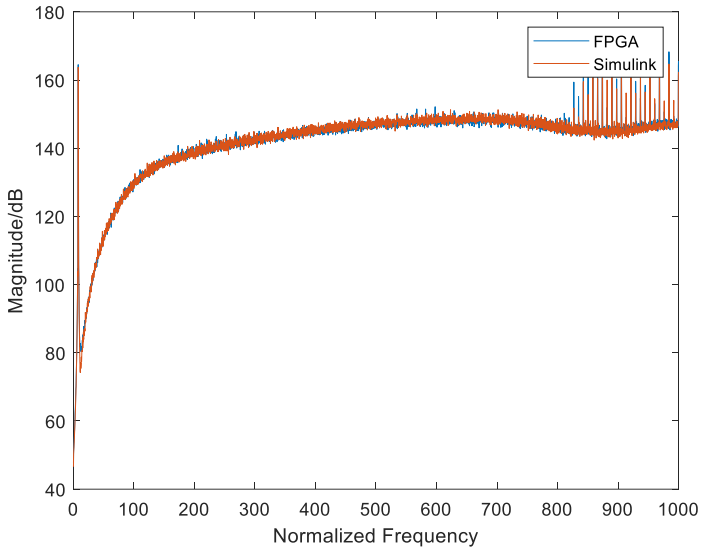


Fig. 7.11: Schematic view of the two-segment ΔΣ oscillator provided by QuartusII

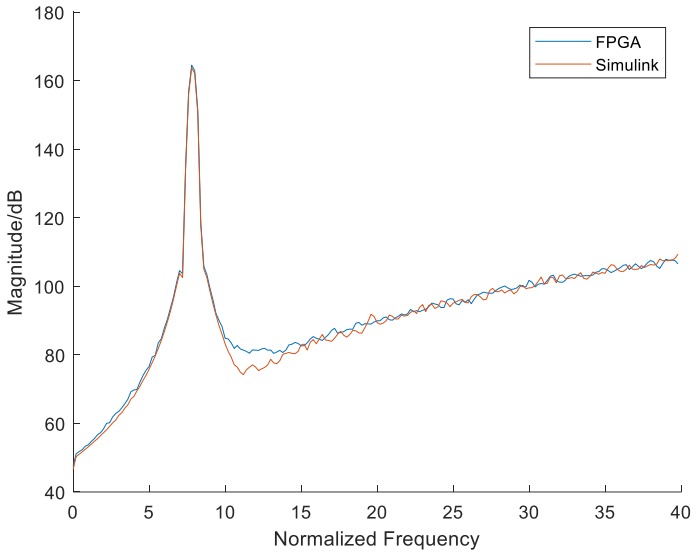
7.3.2 ΔΣ Oscillator with Segmentation

Proceeding with ΔΣ oscillator with segmentation, we built the system in FPGA according to Fig. 6.1 and Fig. 6.2. Following the simulations we performed in Section 6.2, all the parameters are reused and implemented in the FPGA. Fig. 7.11 presents the schematic view of the two-segment ΔΣ oscillator provided by QuartusII. Same as the previous subsection, the modulator output is declared as *out* and the sinewave the oscillator generates is declared as $x1$. Note that the SegMod module is the two-segment modulator we built up in Chapter 7.2.

By using the third order inverse-Chebyshev filter given in Eqn. (7.3) as the coarse filter and the first order filter given in Eqn. (6.9) as the fine filter, the output power spectral density from the Simulink simulation and the FPGA implementation of two-segment ΔΣ oscillator are shown in Fig. 7.12.



(a)



(b)

Fig. 7.12: Output PSD of Simulink simulation and FPGA implementation of the two-segment ΔΣ oscillator (a): Nyquist frequency (b): early frequency

Note that $x1$ represents the output of the initial register. The results of the Simulink simulation and the testbench file from the FPGA implementation of $x1$ are plotted in Fig. 7.13. It is clear that the signal demonstrates a sinusoidal waveform and the outcomes from the FPGA implementation's testbench file closely mirror those derived from the MATLAB/Simulink simulation.

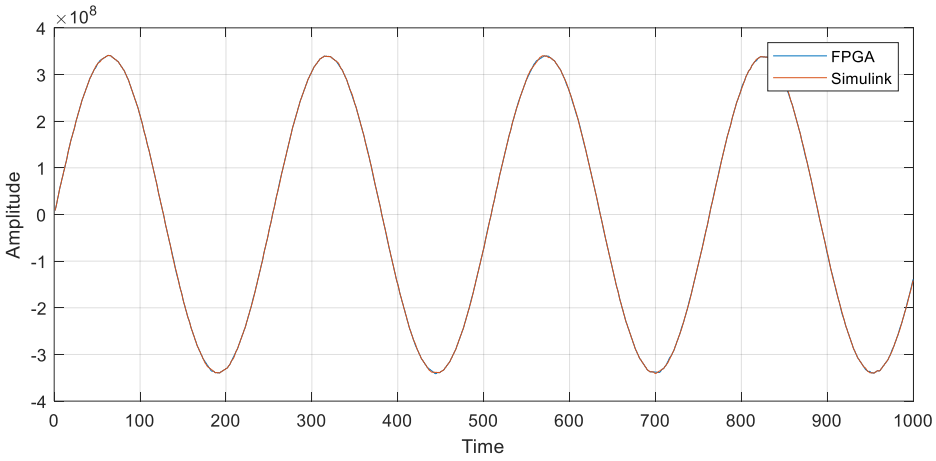


Fig. 7.13: Simulink and FPGA sinewave output of the two-segment ΔΣ oscillator

As evident in Fig. 7.12 and Fig. 7.13, the FPGA performance closely aligns with the Simulink results. Table. 7.3 shows the value of oscillation frequency, amplitude and system SNR derived from theory, Simulink, and FPGA, based on the modulator output.

	A	f_o	SNR
Theoretical	3.3957×10^8	7.8641 Hz	91.0075 dB
Simulink	3.5366×10^8	7.8641 Hz	81.6227 dB
FPGA	3.3592×10^8	7.8641 Hz	81.6089 dB

Table. 7.3: Theory, Simulink and FPGA results of oscillation frequency, amplitude and system SNR of the one-path ΔΣ oscillator

The results presented in Table. 7.3 further proves the validity of the two-segment ΔΣ oscillator implementation in FPGA. The oscillation frequency in the FPGA implementation precisely matches the values from theory and Simulink. There is some variance in amplitude and SNR when compared to Simulink, but it is small and falls within the specified requirements.

7.4 Hardware Costs

In this section, we conduct a comprehensive assessment of the hardware costs associated with single-path ΔΣ modulators and ΔΣ oscillators, as well as their two-segment counterparts.

For each FPGA implementation, Quartus II facilitates the generation of a resource utilization report. This report provides detailed information about the FPGA resources used in the design. This includes total logic elements (TLE), total registers, I/O pins and phase locked loops (PLL) used in the design.

The outcomes pertaining to hardware costs for both 32-bits single-path and two-segment ΔΣ modulators, as presented in Chapter 7.2, are shown in Table. 7.4. The results of hardware costs for single path and two-segment ΔΣ oscillator implemented in Chapter 7.3 are shown in Table. 7.5.

ΔΣ modulator	Total logic elements	Total registers	Total pins
Single path	2002	411	114
Two-segment	1093	273	114

Table. 7.4: Hardware costs for single path and two-segment ΔΣ modulator

ΔΣ oscillator	Total logic elements	Total registers	Total pins
Single path	3868	463	130
Two-segment	1320	401	130

Table. 7.5: Hardware costs for single path and two-segment ΔΣ oscillator

As can be seen from Table. 7.4 and 7.5, TLEs are significantly reduced due to segmentation. Specifically, for the 32-bit ΔΣ modulator, segmentation leads to a noteworthy 45.4% reduction in TLEs, while the 32-bit ΔΣ oscillator experiences an even more significant 65.9% reduction in TLEs due to segmentation. This effect extends to the number of registers utilized, which is also notably lower in comparison to single-path implementations.

In conclusion, the use of segmentation relaxes the hardware costs excessively for a relatively small decrease in SNR. This outcome underscores the practicality and resource-efficiency of segmentation in enhancing the performance of ΔΣ modulators and oscillators on hardware implementation.

7.5 Summary

This chapter delves into the hardware implementation of single-path and two-segment ΔΣ modulators and ΔΣ oscillators. It begins by introducing the FPGA setup and how number system is implemented within the FPGA. Subsequently, the FPGA implementation of ΔΣ modulators and ΔΣ oscillators is conducted. The outputs we obtained from FPGA are then analyzed and compared to the previous MATLAB/Simulink simulation. Result shows that the oscillation frequency and SNR of the ΔΣ oscillators are close to that of Simulink simulations. Finally, the hardware costs are calculated, revealing that utilizing segmentation can alleviate hardware resources by employing fewer TLEs.

Chapter 8: Conclusion

8.1 Discussion of Results

The pursuit of high-resolution, high-precision analog oscillators is a critical task in the field of mixed-signal testing and digital-to-analog conversion. The aim of this thesis is to explore the application of segmentation in ΔΣ oscillators, an approach that offers promising solutions for achieving the desired performance metrics while addressing hardware cost and resource utilization constraints.

The thesis begins with the introduction of analog signal generation and ΔΣ modulation. Then an extensive literature review of DAC performance metrics and structures is provided. In the first two chapters, use of segmentation in DACs and oscillators is introduced and will be the main topic we explore in the later chapters.

The third and fourth chapter provide the structures and designs of ΔΣ modulators and ΔΣ oscillators. Stability research is studied, and different transfer function mapping topologies are offered and compared. Then simulations in MATLAB and Simulink are conducted. Given a specific SNR, proper NTF designs of ΔΣ modulators and ΔΣ oscillators are provided to meet the targeted SNR.

The concept of segmentation is introduced in ΔΣ modulators in Chapter 5 and in ΔΣ oscillators in Chapter 6. The segmentation principle and how data is partitioned is introduced. Then simulations in Simulink are conducted. First and foremost, it became evident that segmentation does not affect the noise shaping if the coarse modulator remains the same. Simulation results show that the SNR

generated in two-segment ΔΣ modulators and ΔΣ oscillators are slightly lower than that of the single path, which strongly proves the theory.

Finally in Chapter 7, FPGA implementation is conducted. By comparing the outcomes of FPGA and implementation, we can conclude that segmentation has the potential to dramatically reduce hardware costs while maintaining the high SNR. By partitioning the oscillator into smaller, manageable segments, resource allocation is optimized, thereby relaxing the resource constraints on the FPGA. This, in turn, translates into cost savings, making the design and deployment of high-precision oscillators more economically viable.

8.2 Future Direction

As we conclude this exploration, it is evident that segmentation is a strategy worth pursuing for the design of high-resolution sigma-delta oscillators. However, there are some future works that can be conducted.

Future research in this field could delve deeper into the optimization of segmentation techniques, and the exploration of novel applications for high-precision oscillators in diverse domains. What's more, a tool like DSMOD built by Haurie [23] could be developed to generate Verilog coed for FPGA implementation of segmented ΔΣ oscillators according to different designs of loop filters.

In closing, this thesis has shed light on the potential of segmentation in ΔΣ oscillators. Hopefully the insights gained here will inspire further innovations and discoveries, leading to the realization of high-precision segmented ΔΣ oscillators meeting the demands of electronic world.

Appendix

Verilog Source Code

SegOsc

```
// Segmented Oscillator
// Author: Wenkang Zhou
// McGill University
// Monteal, QC

module SegOsc(rstn,clk,out,x1);

//Define inputs and outputs
input wire signed [31:0]x1;
output wire signed [31:0]out;
input clk;
input rstn;

wire signed [31:0]x2;
//Define x1(n+1) and x2(n+1) as well as wire after multiplication
wire signed [31:0]x11;
wire signed [31:0]x22;
wire signed [31:0]mul;

wire signed [31:0]muxout;


Dflip1 dff1(x11,rstn,clk,x1);
Dflip2 dff2(x22,rstn,clk,x2);

SegMod SegMod1(x1,rstn,clk,out);

mux2_1 mux(-32'd20,32'd20,out,muxout); //Parameter a2=20
add add1(x22,muxout,x2);

multiplier multi2(mul,x22,32'd1); // Parameter a1=1
add add2(x11,mul,x1);

endmodule

// Module of Register 1
```

```
module Dflip1 (d,rstn,clk,q);

input signed [31:0]d;
input rstn;
input clk;
output signed [31:0]q;
reg signed [31:0]q;

   always @ (posedge clk or negedge rstn)
     if (!rstn)
       q <= 64'd128;  // Parameter x1=128
     else
       q <= d;
endmodule

// Module of Register 2
module Dflip2 (d,rstn,clk,q);

input signed [31:0]d;
input rstn;
input clk;
output signed [31:0]q;
reg signed [31:0]q;

   always @ (posedge clk or negedge rstn)
     if (!rstn)
       q <= 64'd128;  // Parameter x2=128
     else
       q <= d;
endmodule

//Module of add
module add(sum,a,b);

output signed [31:0]sum;
input signed  [31:0]a;
input signed  [31:0]b;
reg signed [31:0] sum;

 always@(a or b)
   begin
         sum<=a+b;
      end
endmodule
```

```
// Segmented Modulator Module
module SegMod (signal,rstn,clk,Segout);

output signed [31:0]Segout;
input signed [31:0]signal;
input clk;
input rstn;
wire signed [31:0]Segout;
wire signed [31:0]signal;
wire signed [31:0]coarsein;
wire signed [31:0]coarseout;
wire signed [31:0]beforefinein;
wire signed [31:0]finein;
wire signed [31:0]fineout;
wire signed [31:0]aftercoarseout;

div div1(coarsein,signal,32'd256);
mul mul1(beforefinein,coarsein,32'd256);
subtract sub1(finein,signal,beforefinein);
modulator coarsemod(coarsein,coarseout,rstn,clk);
modulatorfine finemod(finein,fineout,rstn,clk);
mul mul2(aftercoarseout,coarseout,32'd256);
add add1(Segout,aftercoarseout,fineout);
endmodule

//Multiplier Module
module mul(
    input wire [15:0] integerPart1,
    input wire [15:0] fractionalPart1,
    input wire [15:0] integerPart2,
    input wire [15:0] fractionalPart2,
    output wire [31:0] result
);

wire [31:0] product1;
wire [31:0] product2;
wire [31:0] intermediateResult;

assign product1 = integerPart1 * integerPart2; // Integer multiplication
assign product2 = fractionalPart1 * fractionalPart2; // Fractional multiplication

// Combine integer and fractional products
assign intermediateResult = {product1[31:16], product2[31:16]}; // Combine the 16 MSBs
```

```
always @* begin
    if (product1 >= 2'b01_0000_0000_0000_0000) // Check for positive integer overflow
        result = 32'b01_1111_1111_1111_1111_1111_1111;
    else if (product1 <= -2'b01_0000_0000_0000_0000) // Check for negative integer overflow
        result = -32'b01_1111_1111_1111_1111_1111_1111;
    else
        result = intermediateResult;

    // The fractional part should be handled separately to maintain precision
    if (product2 >= 32'b10_0000_0000_0000_0000_0000_0000)
        result[15:0] = 16'b1111_1111_1111_1111; // Saturate to the maximum positive value
    else if (product2 <= -32'b10_0000_0000_0000_0000_0000_0000)
        result[15:0] = -16'b1111_1111_1111_1111; // Saturate to the maximum negative value
    else
        result[15:0] = product2[15:0]; // No fractional overflow, use the product2 value
end

endmodule


// Modulator Module
module modulator(signal,out,rstn,clk);
output signed [31:0]out;
input signed [31:0]signal;
input clk;
input rstn;
wire signed [31:0]out;
wire signed [31:0]signal;
wire signed [31:0]TFin;
wire signed [31:0]TFout;
wire signed [31:0]TFDout;
wire signed [31:0]relayin;

subtract sub1(TFin,signal,out);
TF TF1(TFin,rstn,clk,TFout);
add add2(relayin,signal,TFout);
relay relay1(relayin,out);

endmodule


//Fine Modulator Module
module modulatorfine(signal,out,rstn,clk);
```

```
output signed [31:0]out;
input signed [31:0]signal;
input clk;
input rstn;
wire signed [31:0]out;
wire signed [31:0]signal;
wire signed [31:0]TFin;
wire signed [31:0]TFout;
wire signed [31:0]relayin;

subtract sub11(TFin,signal,out);
TFfine TF11(TFin,rstn,clk,TFout);
add add22(relayin,signal,TFout);
relay relay11(relayin,out);

endmodule


// TFcoarse Module
module TF (TFin,rstn,clk,TFout);
output signed [31:0]TFout;
input signed  [31:0]TFin;
input clk;
input rstn;
wire signed [31:0] TFout;
wire signed [31:0] TFin;

wire signed [31:0] x1,y1,x2,y2,x3,y3; // shift registers for delay

parameter b0 = 54617; // 0.8334 in 32-bit fixed-point
parameter b1 = -88689; // -1.3533 in 32-bit fixed-point
parameter b2 = 37388; // 0.5705 in 32-bit fixed-point
parameter a0 = 65536; // 1 in 32-bit fixed-point
parameter a1 = -196562; // -2.9993 in 32-bit fixed-point
parameter a2 = 196562; // 2.9993 in 32-bit fixed-point
parameter a3 = -65536; // 1 in 32-bit fixed-point

Dflip d11 (TFin,rstn,clk,x1);//x(n-1)
Dflip d12 (x1,rstn,clk,x2);//x(n-2)
Dflip d13 (x2,rstn,clk,x3);//x(n-3)
Dflip d21 (TFout,rstn,clk,y1);//y(n-1)
Dflip d22 (y1,rstn,clk,y2);//y(n-2)
Dflip d23 (y2,rstn,clk,y3);//y(n-3)
```

```
assign TFout = (b0*x1+b1*x2+b2*x3-a1*y1-a2*y2-a3*y3)/a0;
endmodule


//TFfine Module
module TFfine (TFin,rstn,clk,TFout);
output signed [31:0]TFout;
input signed  [31:0]TFin;
input clk;
input rstn;
wire signed [31:0] TFout;
wire signed [31:0] TFin;

wire signed [31:0] x1,y1,x2,y2; // shift registers for delay

Dflip d111 (TFin,rstn,clk,x1);//x(n-1)

Dflip d211 (TFout,rstn,clk,y1);//y(n-1)


assign TFout = x1+y1;
endmodule


//Relay Module
module relay (relayin,relayout);
output signed [31:0]relayout;
input signed  [31:0]relayin;
reg signed [31:0] relayout;
// Threshold is defined to be 0

always @(relayin)
begin
    if(relayin>=0)
        relayout<=64'd128;
        else
        relayout<=-64'd128;

end

endmodule


//Multiplexer Module
```

```
module mux2_1(A,B,X,muxout);

output signed [31:0] muxout;
input signed [31:0] A;
input signed [31:0] B;
input signed [31:0] X;
reg signed [31:0] muxout;

always@(X,A,B)
begin
  if(X>=0)
    muxout<=A;
     else
          begin
          muxout<=B;
    end
end

endmodule


//Testbench file
// Verilog Test Bench template for design : SegOsc
//
// Simulation tool : ModelSim-Altera (Verilog)
//

`timescale 1 ps/ 1 ps
module SegOsc_vlg_tst();
// constants
// general purpose registers
reg eachvec;
// test vector input registers

reg clk;
reg rstn;
// wires
wire [31:0]  muxin;
wire [31:0]  x1;


//Generated sinewave data will be saved in a text file
integer filew1;
integer filew2;
```

```
// assign statements (if any)
SegOsc i1 (
// port map - connection between master ports and signals/registers
      .clk(clk),
      .muxin(muxin),
      .x1(x1),
      .rstn(rstn)
);
initial
begin
// code that executes only once
// insert code here --> begin
   clk=0;
       forever #50 clk = ~clk;
// --> end
$display("Running testbench");
end

initial
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
   filew2=$fopen("D:\\Matlab_files\\Segmented_Oscillator\\sigoscsineout.txt","w");
   filew1=$fopen("D:\\Matlab_files\\Segmented_Oscillator\\sigoscsinex1.txt","w");
       rstn=0;
       #10 rstn=1;

       #101000000  $stop;
@eachvec;
// --> end
end

   always@ (posedge(clk))
   begin
            $fwrite(filew2,"%d",$signed(muxin));
            $fwrite(filew1,"%d",$signed(x1));
   end

endmodule
```

Bibliography

[1] Mark Burns and Gordon W. Roberts, "An Introduction to Mixed-Signal IC Test and Measurement", 2001.

[2] L. Burst and M-S. Tsey, "Mixing Signals and Voltages on Chip", IEEE Spectrum, New York, NY, Aug. 1993, pp. 40-43.

[3] M. F. Toner and G.W. Roberts, “A BIST Scheme for an SNR, Gain Tracking, and Frequency Response Test of a Sigma-Delta ADC," IEEE Trans. on Circuits and Systems -II: Analog and Digital Signal Processing, Vol. 41, No. 12, pp. 1-15, Jan.1995.

[4] H. T. Nicholas and H. Samueli, “A 150MHz Direct Digital Frequency Synthesizer in 1.25 mm CMOS with 90dBc Spurious Performance”, IEEE Journal of Solid-State Circuits, vol. 26, pp. 1959-1969, Dec. 1991.

[5] A. K. Lu, G. W. Roberts, and D. Johns, “A high-quality analog oscillator using oversampling D/A conversion techniques," IEEE Trans. on Circuits and Systems - II: Analog and Digital Signal Processing, Vol. 41, No. 7, pp. 437-444, July 1994.

[6] M.L. Blostein, "Sensitivity Analysis of Parasitic Effects in Resistance-Terminated LC Two-Ports," IEEE Trans. Circuits and Systems., vol. 14, pp. 21-25, Mar.1967.

[7] Schreier, R., & Temes, G. C. (2005). Understanding Delta-Sigma Data Converters. New York: IEEE Press.

[8] Schindler, H. R. "Delta modulation." IEEE spectrum 7.10 (1970): 69-78.

[9] Abate, John Edward. "Linear and adaptive delta modulation." Proceedings of the IEEE 55.3 (1967): 298-308.

[10] Temes, Gabor C., Steven R. Norsworthy, and Richard Schreier, eds. Delta-Sigma Data Converters: Theory, Design, and Simulation. ieee Press, 1997.

[11] Romanov, Denis Evguenievitch. ΔΣ digital-to-analog converter with reduced hardware requirements using segmentation. McGill University (Canada), 2022.

[12] Emara, Ahmed S., et al. "An Area-Efficient High-Resolution Segmented ΣΔ-DAC for Built-In Self-Test Applications." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 29.11 (2021): 1861-1874.

[13] Lu, Albert K., Gordon W. Roberts, and David A. Johns. "A high-quality analog oscillator using oversampling D/A conversion techniques." IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 41.7 (1994): 437-444.

[14] A. S. Sedra and K. C. Smith, Microelectronics Circuits, 3rd ed., HRW-Saunders, Florida, 1991.

[15] G. Roberts and M. Burns, "An Introduction to Mixed-Signal IC Test and Measurement", Oxford University Press, 2005.

[16] M. Burns and G. Roberts, An Introduction to Mixed-Signal IC Test and Measurement, 2nd ed., Oxford University Press, 2005.

[17] J. Bryant and W. Kester, “Data converter architectures,” in The Data Conversion sHandbook, W. Kester, 3rd Edition. Oxford, U.K.: Elsevier/Newness, 2005, pp. 147-s174. September 2006

[18] Dennis Dempsey and Christopher Gorman, "Digital-to-Analog Converter," U.S. Patent 5,969,657, filed July 27, 1997, issued October 19, 1999. (describes an elegant solution for

segmented unbuffered string DACs).
[19] T. Wang et al., "High-resolution digital frequency synthesis for software-defined radio," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 61, no. 9, pp. 2719-2731, 2014.
[20] R. Han et al., "A 9 GHz CMOS Quadrature VCO With 29.4% Tuning Range," IEEE Journal of Solid-State Circuits, vol. 49, no. 10, pp. 2267-2278, 2014.
[21] J. Wang et al., "A CMOS high-Q LC oscillator with digital compensation technique," IEEE Journal of Solid-State Circuits, vol. 44, no. 9, pp. 2609-2616, 2009.L.E.
[22] R. Schreier, An empirical study of high-order single-bit delta-sigma modulators," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 40, no. 8, pp. 461–466, Aug. 1993.
[23] X. Haurie, "Signal Generation using High-Order Delta-Sigma Modulation," Master's Thesis, McGill University, November, 1996.
[24] M. R. Miller and C. S. Petrie, "A multibit sigma-delta ADC for multimode receivers," IEEE J. Solid-State Circuits, vol. 38, no. 3, pp. 475–482, Mar. 2003.
[25] Jien-Chung L. Modern digital designs with EDA, VHDL and FPGA[M]. Terasic, 2015.
[26] A. H. Nuttall, "Some Windows with Very Good Sidelobe Behavior", IEEE Trans. Acc. Speech Sig. Proc, Vol. ASSP-29, No. 1, February 1981.