University of Sarajevo
Faculty of Electrical Engineering
Department of Automation and Electronics

FPGA-based Sigma-Delta Class-D power DAC

Master's Thesis
- Second Cycle of Studies -

Author:
Ahmed Imamović

Supervisor:
Doc. dr. sc. Nedim Osmić, dipl. ing. el.

Sarajevo,
January 2025

Sažetak

U završnom radu je predstavljen dizajn i implementacija Sigma-Delta digitalno-analognog konvertora (DAC) zasnovanog na FPGA, integrisanog s pojačalom klase D za efikasnu i kvalitetnu reprodukciju audiosignala. Sigma-Delta modulator drugog reda i interpolacijski filter razvijeni su na nivou transfera registara (engl. *Register-Transfer Level* - RTL), iskorištavajući paralelizam svojstven FPGA arhitekturama za efikasnu digitalnu obradu signala. Izvorni interpolacijski filter dizajniran je u Matlabu, te su korespondirajući Verilog moduli generisani pomoću Matlabovog HDL Codera. Kako je generisana RTL implementacija filtera zahtijevala korištenje enormnog broja resursa to je izvršena optimizacija generisanog koda. Kao analogni dio dizajna iskorišteni su jednostavni niskopropusni RC filter i digitalno pojačalo klase D, te je izvršena usporedba ova dva rješenja. Dizajn je implementiran i testiran na GateMate FPGA evaluacijskoj ploči.

Ključne riječi: FPGA, DAC, Sigma-Delta, D klasa, RTL

Abstract

This thesis presents the design and implementation of an FPGA-based Sigma-Delta Digital-to-Analog Converter (DAC) integrated with a Class-D amplifier, aimed at achieving efficient and high-quality audio signal reproduction. A second-order Sigma-Delta modulator and an interpolating filter were developed at the Register-Transfer Level (RTL), leveraging the parallelism inherent in FPGA architectures to enable efficient digital signal processing. The original interpolating filter was designed in MATLAB, and the corresponding Verilog modules were generated using MATLAB's HDL Coder. These modules were resource-intensive and required manual optimization to improve efficiency and reduce hardware utilization. For the analog part of the design, both a simple RC low-pass filter and a Class-D digital amplifier were implemented and compared in terms of performance. The entire design was successfully implemented and tested on the GateMate FPGA evaluation board.

Keywords: FPGA, DAC, Sigma-Delta, Class-D, RTL

Elektrotehnički fakultet, Univerzitet u Sarajevu
Odsjek za automatiku i elektroniku
Doc. dr Nedim Osmić, dipl.el.ing.
Sarajevo, januar 2024.

Postavka zadatka završnog rada II ciklusa:

Sigma delta digitalno analogni konvertor baziran na FPGA i pojačalu D klase

Master teza bavi se istraživanjem dizajna i implementacije digitalno-analognog konvertera baziranog na Field-Programmable Gate Array (FPGA) platformi, Sigma-Delta modulaciji te pojačalu D klase, u svrhu reproduciranja audiosignala. Fokus teze je na istraživanju različitih načina kojima se proces konverzije može optimalno implementirati na nivou transfera registara (RTL) za potrebe FPGA uređaja sa ograničenim resursima. Digitalni dio predloženog dizajna sastoji se od implementacije interpolacijskog filtera i Sigma-Delta modulatora drugog reda u Verilogu. Analogni dio dizajna uključuje realizaciju pojačala D klase koristeći diskretne komponente. Teza u konačnici predstavlja rezultate i evaluaciju dizajna, iznoseći prednosti kao i prijedloge za poboljšanja.

Polazna literatura:

[1] S. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing", California Technical Pub., 1997.

[2] C. Schmidt, "Interleaving Concepts for Digital-to-Analog Converters", Springer Vieweg Wiesbaden, 2019.

[3] Lyons, Richard G. "Understanding digital signal processing", 3/E. Pearson Education India, 1997.

[4] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, "FPGA-based Implementation of Signal Processing Systems", Wiley, 2017,

Doc. dr Nedim Osmić, dipl. el. ing.

Faculty of Electrical Engineering, University of Sarajevo
Department of Automation and Electronics
Doc. Nedim Osmić, PhD, Electrical Engineer
Sarajevo, January 2024.

Master's Thesis Topic:

FPGA-based Sigma-Delta Class-D power DAC

The master's thesis explores the design and implementation of a Digital-to-Analog Converter (DAC) based on a Field-Programmable Gate Array (FPGA) platform, utilizing Sigma-Delta modulation and a Class-D amplifier for audio reproduction. The focus of the thesis is on investigating various approaches to optimally implement the conversion process at the Register-Transfer Level (RTL) for FPGA devices with limited resources. The digital part of the proposed design includes the implementation of an interpolation filter and a second-order Sigma-Delta modulator in Verilog. The analog part of the design involves the realization of a Class-D amplifier using discrete components. The thesis concludes by presenting the results and evaluation of the design, highlighting its advantages and offering suggestions for further improvement.

Primary Literature:

[1] S. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing", California Technical Pub., 1997.

[2] C. Schmidt, "Interleaving Concepts for Digital-to-Analog Converters", Springer Vieweg Wiesbaden, 2019.

[3] Lyons, Richard G. "Understanding digital signal processing", 3/E. Pearson Education India, 1997.

[4] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, "FPGA-based Implementation of Signal Processing Systems", Wiley, 2017,

---

Doc. dr Nedim Osmić, dipl. el. ing.

Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za automatiku i elektroniku

Izjava o autentičnosti radova
Završni rad
II ciklusa studija

Ime i prezime: Ahmed Imamović
Naslov rada: Sigma delta digitalno analogni konvertor baziran na FPGA i pojačalu D klase
Vrsta rada: Završni rad drugog ciklusa studija
Broj stranica: 60

Potvrđujem:

- da sam pročitao dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cijelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio prisustvo citiranog ili parafraziranog materijala i da sam se referirao na sve izvore;
- da sam dosljedno naveo korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio svaku pomoć koju sam dobio pored pomoći mentora i akademskog tutora/ica.

Sarajevo, 13. 01. 2025. god.

Potpis:
Ahmed Imamović

University of Sarajevo
Faculty of Electrical Engineering
Department of Automation and Electronics

Statement of Authorship
Master's Thesis
2nd Cycle Study Program

Name and Surname: Ahmed Imamović
Title of Thesis: Sigma-Delta Digital to Analog Convertor (DAC) based on FPGA and Class-D Amplifier
Type of Thesis: Master's Thesis of the Second Cycle of Study
Page Count: 60

I confirm:

- that I have read the documents related to plagiarism as defined by the Statute of the University of Sarajevo, the Ethical Code of the University of Sarajevo, and the study rules pertaining to the 1st, 2nd, and integrated study programs of the 1st and 2nd cycle, as well as the 3rd cycle study program at the University of Sarajevo, and that I am informed about plagiarism resources provided on the University's website;
- that I am aware of the university disciplinary rules regarding plagiarism;
- that the work I am submitting as a thesis, independent work, or any written work is my original work and has not been previously submitted;
- that my work has not been previously submitted, either in whole or in part, for obtaining a degree at the University of Sarajevo or any other higher education institution;
- that I have clearly indicated the presence of citations or paraphrasing of materials by referring to original sources;
- that I have properly used and cited all references and bibliographic resources recommended for citation styles, by providing complete bibliographic information for each citation and referenced source;
- that I have appropriately marked all assistance I received from other people, including mentoring and academic tutoring.

Sarajevo, 13. 01. 2025.

_____

Signature:
Ahmed Imamović

Contents

List of Figures **ix**

1 Introduction **1**
1.1 Research motivation and goal . . . 1
1.2 Literature review . . . 2
1.3 Main Objectives and Research Plan . . . 3
1.4 Research methodology . . . 3
1.5 Expected results . . . 3
1.6 Main contributions . . . 3
1.7 Structure of the Thesis . . . 4

2 Proposed Solution **5**
2.1 Realization of interpolating filter . . . 5
2.2 Realization of Sigma-Delta modulator . . . 6
2.3 Utilized hardware - GateMate FPGA Evaluation Board . . . 6
2.3.1 Toolchain Overview . . . 7

3 Digital to Analog Conversion **8**
3.1 DAC specifications . . . 8
3.2 The Need for Oversampling Converters . . . 10
3.3 Sigma-Delta Converters . . . 11
3.4 Digital Filters . . . 13
3.5 Noise Shaping . . . 13
3.6 Output Lowpass Filter . . . 14

4 Interpolating Filter **15**
4.1 Specifications of interpolating filters . . . 15
4.2 Realization of interpolating filters . . . 16
4.2.1 Interpolating filter partitioning . . . 17
4.2.2 Interchange of filtering with upsampling . . . 17
4.2.3 Polyphase structures for interpolation . . . 18
4.2.4 Half-Band Filters . . . 21
4.2.5 Cascaded Integrator-Comb Filters . . . 22
4.2.6 Compensation/Preconditioning FIR Filters . . . 22
4.2.7 Coefficient sharing . . . 24
4.2.8 Coefficient quantization . . . 25
4.3 Interpolating filter implementation . . . 27
4.3.1 Direct implementation . . . 27
4.3.2 MAC implementation . . . 27

5 Sigma-Delta Modulation **29**
5.1 Delta Modulation 29
5.2 Sigma-Delta Modulation 30
5.3 Analysis of a First-Order Sigma-Delta Modulator 31
5.4 Higher-Order Noise Shaping 33
5.5 One-Bit D/A Conversion with Second-Order Noise Shaping 35

6 Design and Implementation **38**
6.1 Design and implementation of interpolation filter 38
6.1.1 Specifications 38
6.1.2 Implementation 39
6.1.3 Design and generating Verilog code 41
6.1.4 RTL optimization 41
6.1.5 Optimization results 44
6.2 Design and implementation of Sigma-Delta modulator 45
6.3 Implementation of a Second-Order Sigma-Delta Modulator 45
6.4 DAC Top Module 46
6.5 Design of Class-D Amplifier 46

7 Experimental results **49**
7.1 DAC testing and evaluation 49
7.2 Class-D amplifier testing and evaluation 51
7.3 Summary 55

8 Conclusion **56**

Index of Terms **59**

List of Figures

2.1 Top-level schematic of the proposed solution. 5
2.2 GateMate FPGA Evaluation Board [16]. 6
2.3 Exemplary Toolchain Flow [17]. 7

3.1 A modern signal processing system contains a DSP core bracketed by data converters [1]. 11
3.2 Sigma-Delta DACs [19]. 12
3.3 Oversampling extends quantization error over a larger band, correspondingly reducing in-band error [2]. 13

4.1 Oversampling in the time and frequency domain [8]. 15
4.2 Interpolation with cascade of lower OSR filters [8]. 17
4.3 Multirate identity: two equivalent systems for upsampling. It is possible to interchange the filter with an expander if the filter is properly modified [12]. 18
4.4 (a) Realization of the direct form polyphase structure, and (b) the transposed form polyphase structure; both structures are shown for M = 3 [12]. 19
4.5 Interpolation system [12]. 20
4.6 Polyphase interpolation structure before (a) and after (b) the application of the multirate identity for upsampling [12]. 20
4.7 Polyphase interpolator with a commutator instead of delays [12]. 20
4.8 Impulse response and frequency response of a half-band FIR filter with order $M = 18$. Notice the odd symmetry of $H\left(e^{j\omega}\right)$ about $\omega = \pm\pi/2$ [12]. . 21
4.9 CIC filter applications[11]. 22
4.10 Compensation FIR filter magnitude responses: (a) with a first-order decimation CIC filter; (b) with third-order decimation CIC filter [11]. 23
4.11 Frequency magnitude response of a decimate-by-two compensation FIR filter [11]. 24
4.12 Symmetrical FIR filter with coefficient sharing [8]. 24
4.13 Frequency response, 16-bit coefficient quantization [8]. 26
4.14 Direct implementation of filter transfer function [8]. 27
4.15 MAC FIR-filter implementation [8]. 28

5.1 Delta modulation (DM) yields a signal that falls with frequency [2]. 30
5.2 Sigma-delta modulation (SDM) yields a noise floor that rises with frequency [2]. 31
5.3 A z-transform analysis of a Sigma-Delta modulator [2]. 32

5.4 With one-bit conversion, quantization noise is quite high. In-band noise is reduced with oversampling. With noise shaping, quantization noise is shifted away from the audio band, further reducing in-band noise [2]. . . . 32
5.5 Analysis of a first-order Sigma-Delta noise shaper [2]. 33
5.6 Higher orders of noise shaping result in more pronounced shifts in requantization noise [2]. 34
5.7 Operation of a second-order noise-shaping circuit [2]. 35
5.8 Analysis of a second-order noise shaper [2]. 36

6.1 Frequency response of the IF. 41
6.2 Passband and transition-band of the IF. 42
6.3 Block diagram of IF top module [6]. 42
6.4 Signal interface of the filter stages [6]. 43
6.5 FSM implementation of first two stages. 43
6.6 Comparison of the original and optimized filter response. 44
6.7 Signal interface of the Sigma-Delta modulator. 45
6.8 DAC Top Module. 46
6.9 Functional block diagram of class D amplifier [20]. 47
6.10 Audio to PWM signal [20]. 47
6.11 Circuit diagram of Class-D amplifier based solution. 48

7.1 DAC output through RC: 440Hz sinewave. 49
7.2 DAC output through RC: 440Hz sinewave amplitude spectrum. 50
7.3 440Hz sinewave SNR. 50
7.4 440Hz sinewave THD. 51
7.5 Lab bench setup: 1) GateMate's Evaluation Board, 2) Power supply, 3) Class-D amplifier circuit, 4) Speakers, and 5) PMOD output pins. 52
7.6 Class-D output: 2kHz sinewave. 52
7.7 Class-D output: 2kHz sinewave amplitude spectrum. 53
7.8 Class-D output: 2kHz sinewave SNR. 53
7.9 Class-D output: 2kHz sinewave THD. 54
7.10 Class-D output: power spectrum. 54
7.11 Post-filter for a 1-bit ΣΔ DAC and associated signals [1]. 55
7.12 Sallen-Key Low-Pass Filter [22]. 55

Chapter 1

Introduction

Since its inception in the early 1960s, sigma-delta modulation has evolved through a number of generations and now stands as one of the most popular methods for constructing high-performance analog-to-digital (ADC) and digital-to-analog converters (DAC). The concept of noise-shaping, which is central to sigma-delta modulation, continues to be refined and applied to new areas [1].

Sigma-Delta conversion methods are widely used in digital audio products. Delta modulation and Sigma-Delta modulation (also called Delta-Sigma) were developed in the 1940s and 1960s, respectively, for voice telephony applications. Limitations prohibited their use in high-quality music applications until the emergence of high-speed digital signal processing techniques in the 1980s, that improved audio performance [2].

In this thesis, an efficient implementation of a sigma-delta DAC on a Field-Programmable Gate Array (FPGA) platform is presented. This approach leverages the FPGA's inherent parallelism and configurability to achieve precise signal processing, ensure cost-effectiveness, and enable integration into modern systems.

1.1 Research motivation and goal

The motivation for using sigma-delta modulation to realize high-performance DACs lies in the fact that achieving linearity and accuracy better than approximately 14 bits is difficult, if not impossible, for DACs operating at the Nyquist rate. Sigma-delta modulation makes this task feasible [1].

The ones and zeros produced by sigma-delta modulator are very easy to transform back into an analog signal. All that is required is an analog low-pass filter, which might be as simple as a single RC network [3]. These advantages make them ideal for integration with modern digital embedded systems.

Leveraging FPGAs to implement a sigma-delta DAC not only allows for efficient and scalable designs but also facilitates rapid prototyping and customization. The availability of multiple distributed logic resources and dedicated registers makes FPGAs highly attractive for creating highly parallel, pipelined circuit architectures for digital signal processing (DSP) implementation [4].

In addition, the integration of a Class-D amplifier with the sigma-delta DAC enhances the system's utility in high-efficiency audio applications. Class-D gives the highest efficiency of any of the amplifier classes [5].

The goal of the thesis is to design and implement a sigma-delta DAC on an FPGA platform, interfaced with a Class-D amplifier, to provide a robust, efficient, and high-quality audio output solution. By combining the digital signal processing capabilities of the FPGA with the energy-efficient performance of the Class-D amplifier, this work aims to contribute to the development of compact and power-efficient digital audio systems suitable for a wide range of applications.

1.2 Literature review

The main paper this thesis relies on is the work found in [6], which details the design and simulation of a Sigma-Delta modulator in C code. The referenced work establishes a performance benchmark using the Compact Disc Digital Audio (CD-DA) format, with 16-bit samples at a 44.1 kHz sample rate, achieving a dynamic range (DR) greater than 96 dB for stable inputs below -2.5 dBFS. It presents an interpolation filter (IF) with an oversampling ratio (OSR) of 128, compatible with the CD-DA format implemented in RTL code. Additionally, a complete Sigma-Delta DAC intellectual property (IP) core was developed, featuring a control register and an AMBA AXI4-compatible data streaming interface, tailored for FPGA implementation. The final design was deployed on a Zynq-7000 SoC and its audio performance was evaluated using a spectrum analyzer. To provide a comprehensive understanding of its performance, the Sigma-Delta DAC solution was compared with a PWM DAC solution, highlighting trade-offs in power consumption, area usage, and audio quality.

Building upon this foundation, this thesis extends the work by implementing the Sigma-Delta modulator in RTL code and integrating it with a modified version of the proposed interpolation filter to create a complete Sigma-Delta DAC for FPGA deployment.

The application note from Analog Devices [7] provides foundational insights into the operation and application of Sigma-Delta converters. This document outlines the principles of Sigma-Delta modulation, including its reliance on oversampling and noise shaping to achieve high dynamic range and resolution with minimal analog circuitry. It also highlights practical design considerations and the trade-offs involved in implementing Sigma-Delta converters for various applications.

The paper [8] provides a detailed exploration of the design and implementation of interpolation filters tailored for Sigma-Delta digital-to-analog converters. It serves as an essential reference for understanding the role and optimization of interpolation filters in audio DACs, focusing on oversampling techniques and their impact on performance.

Detailed exploration of Sigma-Delta modulation and its practical implementation on FPGAs is given in the paper [9], with several key contributions. It covers both first-order and second-order modulators and highlights their efficient use in FPGA-based designs to achieve high resolution and dynamic range. The document introduces an error-feedback topology for second-order Sigma-Delta DACs, offering simplified yet effective digital designs tailored for FPGAs. Additionally, it addresses common non-idealities such as finite integrator gain, clock jitter, and comparator hysteresis, providing practical strategies for minimizing their impact. This Master thesis emphasizes the advantages of minimizing analog circuitry by leveraging FPGA resources, contributing valuable insights for implementing Sigma-Delta ADCs and DACs in digital systems.

This master's thesis also draws on books that have addressed this topic, such as [10].

Mentioned book provides a comprehensive overview of advanced DAC implementations. It also serves as a key reference for the fundamental theory of DACs and the performance metrics used to evaluate DACs.

Also in the books [11], [3] and [12] are used as foundational references for understanding multirate systems, including the theory and implementation of common finite impulse response (FIR) filter structures used for interpolation, such as cascaded integrator-comb (CIC) filters and half-band filters.

Lastly, the book [2] is used as a key reference for fundamental audio concepts and the application of Sigma-Delta modulation in audio systems.

1.3 Main Objectives and Research Plan

The primary objective of this thesis is to design and test an FPGA-based Sigma-Delta DAC, validating each of its core components through RTL simulation (IF and Sigma-Delta modulator) and in hardware (Class-D amplifier). Each of these components will be developed to operate as a standalone module, ensuring individual functionality before integrating them into a fully operational digital-to-analog conversion system.

The project begins with an in-depth theoretical study of each component, leaning on the literature already presented. After getting familiar with the basics, it is necessary to explore all possible optimization techniques that allow the design to be implemented using minimal hardware resources.

1.4 Research methodology

This research begins with a comprehensive understanding of the core concepts of digital signal processing using hardware, such as coefficient quantization, signed-bit addition, and multiplication. After that, the focus is on researching the best ways of designing, implementing, and verifying the individual components. Each module will then be simulated and tested to ensure that it meets theoretical expectations.

1.5 Expected results

The expected result is that the genereted sinewave at the input of the DAC is correctly reconstructed at the output of the analog lowpass filter, with minimal noise, and high signal-to-noise ratio making it possible to use in high-quality audio applications.

1.6 Main contributions

The contributions of the Master thesis are following:

- design of a modified version of the interpolation filter proposed in [6] with FSM based optimisation of the MATLAB generated RTL,
- RTL implementation of a second-order Sigma-Delta modulator,

- implementation of a S-D DAC IP for a Cologne Chip's CCGM1A1 FPGA on a GateMate FPGA evaluation board, helping improve the GateMate FPGA toolchain,
- realization of a Class-D audio amplifier using MOSFET transistors.

1.7 Structure of the Thesis

This thesis is divided into eight chapters, each focusing on a distinct aspect of the research.

Chapter 1 opens with an introduction to Sigma-Delta modulation, explaining its significance in digital-to-analog converters while detailing the research motivation, objectives, and primary contributions of this work.

Chapter 2 presents the proposed architecture of the Sigma-Delta DAC, detailing the integration of its components, including the interpolation filter, Sigma-Delta modulator, and analog interface using a Class-D amplifier.

A deeper understanding of digital-to-analog conversion is provided in Chapter 3, where DAC specifications, oversampling requirements, sigma-delta architectures, and related digital filters are examined.

Moving on to interpolation filters, Chapter 4 discusses their design and implementation, introducing efficient realization strategies, computational optimization, and practical considerations such as coefficient quantization.

The principles of Sigma-Delta modulation, both theoretical and practical, form the focus of Chapter 5, which includes a comprehensive analysis of first and second-order modulators.

The details of the hardware design of the Sigma-Delta DAC design are presented in Chapter 6, with an emphasis on RTL optimization, Verilog code generation, and Class-D amplifier integration.

Evaluation and testing results are showcased in Chapter 7, where the performance of the Sigma-Delta DAC and Class-D amplifier is analyzed.

In the final, Chapter 8 reflects on the achievements of the thesis, outlines its contributions to the field, and offers directions for future work.

Chapter 2

Proposed Solution

This thesis dives into the ways of implementing a Sigma-Delta DAC for audio applications utilizing FPGA hardware. A Sigma-Delta DAC consists of three main parts: an interpolating filter, Sigma-Delta modulator and analog circuitry that converts a digital signal to audible sound. In this thesis, a Class-D amplifier realized using discrete components is used to interface between and FPGA output and speakers. This means that most of the work is in digital domain and is implemented on an FPGA platform. Figure 2.1 displays a top level scheme of the developed solution. It can be seen that parts of FPGA fabric are also used for generating audio signals, mainly simple sinewaves.

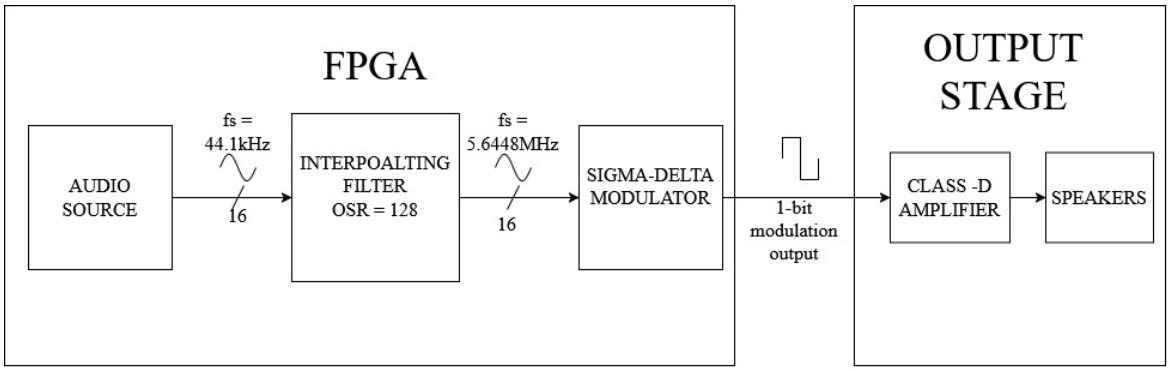


Figure 2.1: Top-level schematic of the proposed solution.

2.1 Realization of interpolating filter

Once the specifications are set, the filter needs to be designed and implemented in RTL code. Matlab offers convinient tools for both filter design and HDL implementation. The *filterBuilder* function provides a graphical interface to the object-oriented filter design paradigm and is intended to reduce development time during the filter design process [13]. This builder provides an easy way to fixed-point quantization. User can specify the word length and fraction length as needed for filter input and output, the filter numerator and denominator coefficients, the output of product operations, the output of arithmetic operations within the filter and the accumulators. It also offers a way to set how the filter treats values that overflow with several rounding or overflow modes. *Code Generation* functionality allows a user to generate VHDL or Verilog code from the designed filter.

2.2 Realization of Sigma-Delta modulator

In a later chapter Sigma-Delta modulator will be represented as a set of differential equations, making it very easy to implement on hardware. The RTL implementation requires a single module and uses both combinatorial and sequential logic.

The hardware description language (HDL) of choice, for both IF and modulator, is Verilog.

2.3 Utilized hardware - GateMate FPGA Evaluation Board

The GateMate FPGA family is the first one from EU based manufacturing . The architecture of the GateMate FPGA is based on a novel Cologne Programmable Element (CPE). The CPE architecture enables the efficient construction of arbitrarily sized logic [14].

The smallest chip from the family, CCGM1A1, incorporates a novel CPE architecture with 20,480 programmable elements for combinatorial and sequential logic, 40,960 latches/flip-flops within programmable elements, and a LUT-tree with 8 inputs for each CPE. Each CPE is configurable as a 2-bit full-adder or 2x2-bit multiplier. The FPGA is designed for low power consumption process and supports three operation modes: low power, economy, and speed. It features low start-up current requirements and requires only two supply voltages, which can be applied in any order. Additional features include four programmable PLLs, fast configuration with a quad SPI interface supporting up to 100 MHz and multi-chip configuration capabilities. It includes 1,280 Kbit dual-ported block RAM with variable data widths in 32 x 40 Kbit RAM cells, multipliers with arbitrary size implementable in the CPE array and multiple clocking schemas. All 162 GPIOs are configurable as single-ended or LVDS differential pairs and double data rate (DDR) support is provided in all GPIO cells. The FPGA also supports a 5 Gb/s SerDes. The package is compact, utilizing a 324-ball ball grid array (BGA) package (15x15 mm) and requiring only two signal layers on the printed circuit board (PCB)[15].

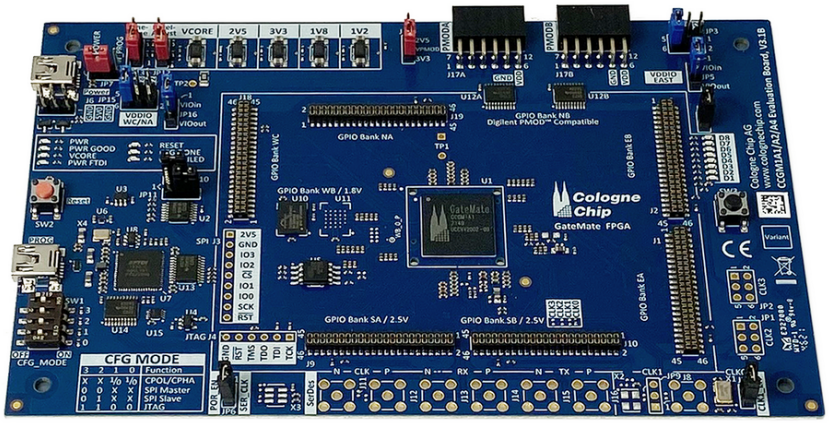


Figure 2.2: GateMate FPGA Evaluation Board [16].

The GateMate FPGA Evaluation Board is a feature-rich, ready-to-use development platform for the CCGM1A1 (see Figure 2.2). Interfaces include USB-JTAG, USB-SPI,

two Pmod-compatible ports and SerDes via SMA. Memory features include 64 Mbit Quad I/O SPI flash (80 MHz), 64 Mbit HyperRAM (166 MHz) and an optional footprint for additional HyperRAM or HyperFlash modules. The board is powered via USB with an adjustable V_{core} of 0.9–1.1V and has dimensions of 100x160 mm (Eurocard standard) [16].

2.3.1 Toolchain Overview

An exemplary toolchain flow from design entry to configuration is illustrated in Figure 2.3.

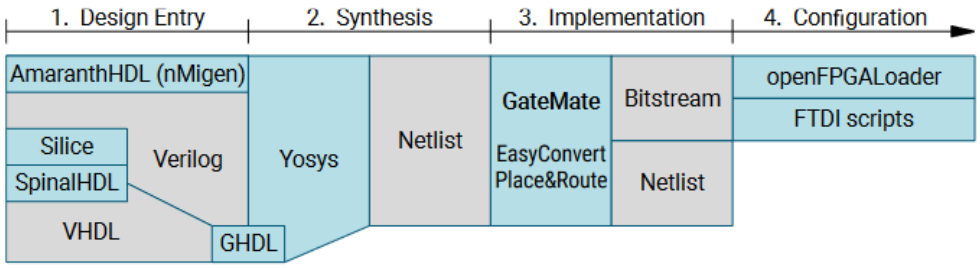


Figure 2.3: Exemplary Toolchain Flow [17].

The *Yosys Open Synthesis Suite* is used to perform RTL synthesis. It has extensive Verilog support. Synthesis generates a gate-level representation of the design entry in form of a Verilog netlist of architecture-specific primitives. It can be simulated or passed to the Place and Route tool for implementation and bitstream generation. Furthermore, simulation of the resulting netlist with back-annotated timing delays can be done using third-party simulators such as *Icarus Verilog* together with the *GTKWave* waveform viewer. Configuration bitstreams are loaded into the FPGA or an external flash memory via *openFPGALoader* [17].

* * *

In this chapter a proposed solution for Sigma-Delta DAC was presented in the form of a top-level schematic. It also specifies which tools are used in certain parts of the design. Lastly, it introduces the board that the design is to be implemented on, GateMate's FPGA Evaluation Board, with all of its features and utilized toolchain.

The next chapter will dive into the basics of the digital-to-analog conversion, highlighting the concepts of oversampling and noise-shaping.

Chapter 3

Digital to Analog Conversion

This chapter servers as an introduction to basics of digital-to-analog conversion. Additionally, the need for oversampling data converters will be discussed and their performance contrasted with that of Nyquist-rate converters.

3.1 DAC specifications

The digital-to-analog converter is one of the most critical elements in the reproduction system. Just as the analog-to-digital converter largely determines the overall quality of the encoded signal, the DAC determines how accurately the digitized signal will be restored to the analog domain [2]. A large set of specifications describe the performance of data converters. DACs are limited by both static and dynamic effects, as well as noise. The primary factors constraining high-speed DAC performance include image replica generation, bandwidth limitations, nonlinearity spurs and jitter [10].

The static performance of a DAC is characterized by multiple parameters, such as analog resolution, offset, gain error, differential nonlinearity (DNL), monotonicity, hysteresis, and integral nonlinearity (INL) [10].

In contrast, the dynamic performance of a DAC is evaluated using various metrics, including sampling rate, frequency response, analog bandwidth, eye diagram, step response, glitch impulse area, signal-to-noise ratio (SNR), signal-to-noise and distortion ratio (SINAD), effective number of bits (ENOB), dynamic range, total harmonic distortion (THD), spurious-free dynamic range (SFDR), intermodulation distortion (IMD) and figure of merit (FoM) [10].

The following text will provide a detailed discussion of some of these parameters and more [18]:

- **Resolution**: the number of bits that a DAC receives at its input to generate an analog output. The resolution, together with the reference voltage determines the minimum change in the output variable for a DAC. This is also known as the quantization step.

- **Dynamic range**: is the ratio between the largest signal level the converter can handle and the noise level, expressed in dB. The dynamic range determines the maximum SNR.

- **Differential nonlinearity**: the amplitude difference between the output signal levels of subsequent input codes is ideally equal. In reality, there are mismatches

resulting in nonlinear distortions. The deviation in amplitude between subsequent output codes is called differential nonlinearity and is usually measured in least significant bits.

- **Integral nonlinearity**: integral nonlinearity measures the deviation of the transfer function from the ideal characteristic. It is calculated by summing the DNL values. Usually, the INL is stated in least significant bits and the maximum INL is referred to as INL. The correlated part of the INL is the main source for nonlinearity, whereas the uncorrelated part can be treated as noise.

- **Monotonicity**: is the ADC feature that produces output codes that are consistently increasing with increasing input signal and consistently decreasing with decreasing input signal. Therefore, the output code will always either remain constant or change in the same direction as the input.

- **Hysteresis**: is the limit that denotes a dependence of the output code on the direction of the input signal (i.e., increasing or decreasing signal). If this happens hysteresis is the maximum of such differences.

- **Settling-time**: is the time at which the step response of a DAC enters and subsequently remains within a specified error band around its final value. The input is a step signal applied at time t = 0. The final value is defined to occur a long time after the beginning of the step.

- **Clock jitter**: is the standard deviation of the sampling time. It is also called aperture jitter or timing phase noise. It is normally assumed that clock jitter is like a noise with a white spectrum.

- **Digital-to-analog glitch impulse**: is the amount of signal injected from the digital inputs to the analog output when the inputs change state. The maximum normally occurs at half scale when the DAC switches around the MSB and many switches change state, i.e., from $01 \cdots 11$ to $10 \cdots 00$. The parameter is the integral of the glitch area and is measured in V-sec or A-sec.

- **Glitch power**: it can be due to a delay between bit controls or to timing mismatch in the analog sections. Normally its maximum occurs at half scale. Similarly to the previous specification it is the integral of the glitch area and is measured in V-sec or A-sec.

- **Signal-to-noise ratio (SNR)**: is the ratio between the power of the signal (normally a sinewave) and the total noise produced by quantization and the noise of the circuit. The SNR accounts for the noise in the entire Nyquist interval. The SNR can depend on the frequency of the input signal and it decreases proportional to the input amplitude.

- **Signal-to-noise-and-distortion ratio (SINAD or SNDR)**: is similar in definition to the SNR except that non linear distortion terms, generated by the input sine wave, are also accounted for. The SINAD is the ratio between the root-mean-square of the signal and the root-sum-square of the harmonic components plus noise (excluding dc). Since static and dynamic limitations cause a non-linear response the SINAD is dependent on both the amplitude and frequency of the input sine wave.

- **Effective number of bits (ENOB)**: measures the signal-to-noise and distortion ratio using bits. SINAD in dB and ENOB are linked by:

$$ENOB = \frac{SINAD_{dB} - 1.76}{6.02} \tag{3.1}$$

- **Total harmonic distortion (THD)**: is the ratio between the root-mean-square of the signal and the root-mean-square of harmonic components including aliased terms. Unless otherwise specified the HD accounts for the second through tenth harmonics: it is normally assumed that harmonic terms higher than the tenth have negligible effects.

- **Intermodulation distortion (IMD)**: accounts for spur tones caused by non-linearity when the input is a complex signal. Non-linearity not only causes distortion of a pure tone; but also, when the input is made of multiple sine waves the interaction between them produces intermodulation terms. This non-linearity of a data converter causes the mixing of the spectral components thus generating spurs at sum and difference requencies for all possible integer multiples of the input frequency tones. The IMD is quantified by parameters or diagrams.

- **Figure of merit (FoM)**: is a parameter used to measure the power effectiveness. It assumes that the total power is consumed mainly because of the bandwidth of the converted signal (BW) and the equivalent number of bits (ENB). Publications or data-sheets use different definitions of the FoM.

3.2 The Need for Oversampling Converters

Computational and signal processing tasks are now performed predominantly by digital means, since digital circuits are robust and can be realized by extremely small and simple structures which can in turn be combined to obtain very complex, accurate and fast systems. Since the physical world nevertheless remains stubbornly analog, data converters are needed to interface with the digital signal processing (DSP) core. As the speed and capability of DSP cores increases, so too must the speed and accuracy of the converters associated with them [1].

Figure 3.1 illustrates the block diagram of a signal processing system with analog input and output signals, plus a central digital engine. As shown, the analog input signal (usually after some amplification and filtering) enters an ADC which transforms it into a digital data stream. This stream is processed by the DSP core, and the resulting digital output signal is reconverted into analog form by a DAC. The DAC output is usually also filtered and amplified to obtain the final analog output signal [1].

Data converters (both ADCs and DACs) can be classified into two main categories: Nyquist-rate and oversampled converters. In the former category, there exists a one-to-one correspondence between the input and output samples. Each input sample is separately processed, regardless of the earlier input samples; the converter has no memory. As the name implies, the sampling rate f_s of Nyquist-rate converters can be as low as Nyquist's criterion requires, i.e., twice the bandwidth f_B of the input signal.

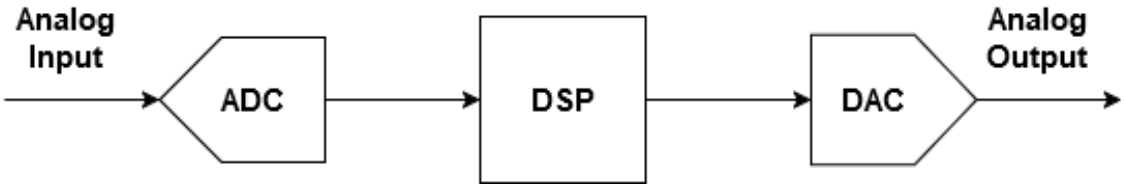


Figure 3.1: A modern signal processing system contains a DSP core bracketed by data converters [1].

In most cases, the linearity and accuracy of Nyquist-rate converters is determined by the matching accuracy of the analog components (resistors, current sources or capacitors) used in the implementation. In many applications (such as digital audio), higher resolution and linearity is required, perhaps as much as 18 or even 20 bits. The only Nyquist-rate converters capable of such accuracy are the integrating or counting ones. These, however, require at least 2N clock periods to convert a single sample, and hence are too slow for most signal-processing applications [1].

Oversampling data converters are able to achieve over 20 ENOB resolution at reasonably high conversion speeds by relying on a trade off. They use sampling rates much higher than the Nyquist rate, typically higher by a factor between 8 and 512, and generate each output utilizing all preceding input values. Thus, the converter incorporates memory elements in its structure. This property destroys the one-to-one relation between input and output samples. Now only a comparison of the complete input and output waveforms can be used to evaluate the converter's accuracy, either in the time or in the frequency domain.

The implementation of oversampling converters requires a considerable amount of digital circuitry, in addition to some analog stages. Both need to be operated faster than the Nyquist rate. However, the accuracy requirements on the analog components are relaxed compared to those associated with Nyquist-rate converters. The cost paid for high accuracy thus includes faster operation and added digital circuitry; both of these are getting cheaper as digital IC technology advances. Hence, oversampling converters are gradually taking over in many applications previously dominated by Nyquist-rate ones [1].

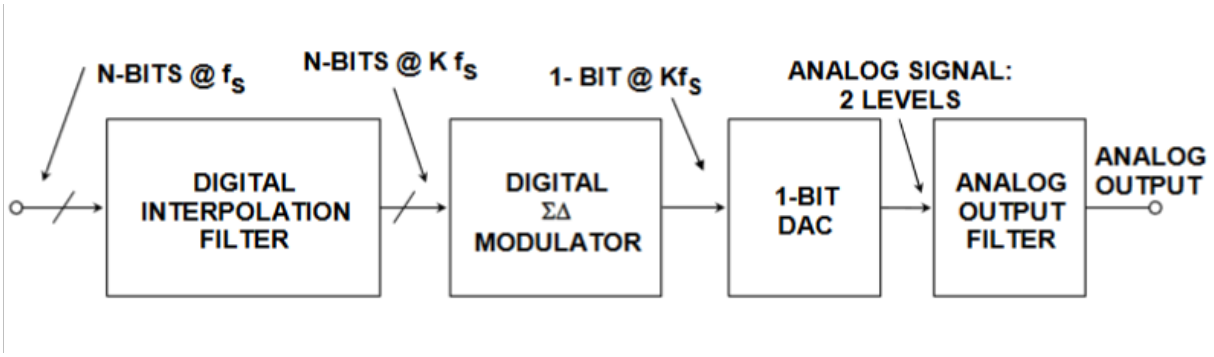
3.3 Sigma-Delta Converters

Design limitations and the relative cost of pulse-code modulation (PCM)[1] converter architectures encouraged the development of Sigma-Delta converters. These converters are characterized by short word lengths, very high oversampling rates, and noise shaping. They demonstrate that conversion can be performed either with a high-resolution quantizer at a low sampling rate (as in traditional converters), or with a low-resolution quantizer at a high sampling rate (as in Sigma-Delta converters). Sigma-Delta ADCs and DACs both use conversion methods such as Sigma-Delta modulation with noise shaping, and process high sampling-rate signals with oversampling and decimation filters. These

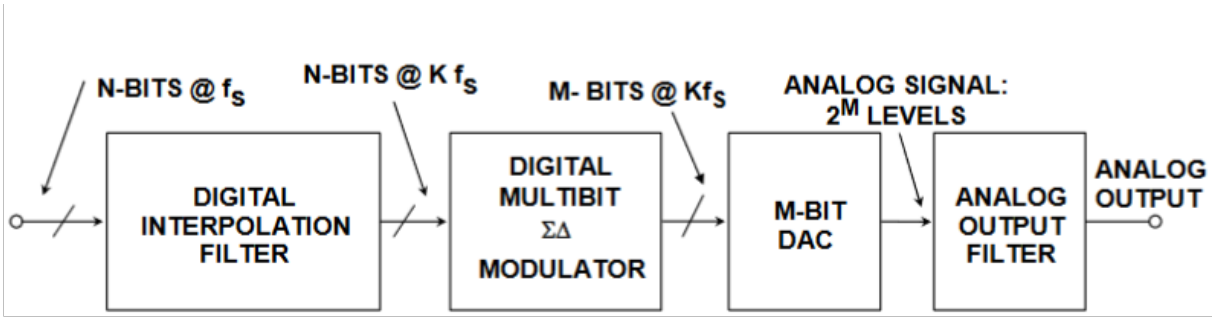
[1]The most commonly used modulation method is pulse-code modulation (PCM). In PCM, the input signal undergoes sampling, quantization, and coding. By representing the measured analog amplitude of samples with a pulse code, binary numbers can be used to represent amplitude. At the receiver, the pulse code is used to reconstruct an analog waveform.

converters share the goal of translating nonideal converter errors into uncorrelated, benign noise. Sigma-Delta converters are sometimes referred to as one-bit or multi-bit converters, depending on the specific architecture employed [2].

A Sigma-Delta DAC, unlike the Sigma-Delta ADC, is mostly digital (see Figure 3.2). It consists of an *interpolation filter* (a digital circuit which accepts data at a low rate, inserts zeros at a high rate, and then applies a digital filter algorithm and outputs data at a high rate), a Sigma-Delta modulator (which effectively acts as a low pass filter to the signal but as a high pass filter to the quantization noise, and converts the resulting data to a high-speed bit stream), and a 1-bit DAC (see Figure 3.2a) whose output switches between equal positive and negative reference voltages. The output is filtered in an external analog lowpass filter (LPF). Because of the high oversampling frequency, the complexity of the low-pass filter (LPF) is much less than the case of traditional Nyquist operation. It is possible to use more than one bit in the Sigma-Delta DAC, and this leads to the multi-bit architecture shown in Figure 3.2b [19].



(a) Single bit Sigma-Delta DAC.



(b) Multibit Sigma-Delta DAC.

Figure 3.2: Sigma-Delta DACs [19].

To summarize, oversampling used in conjunction with digital filtering is a powerful tool in modern sampled data systems. A primary advantage is the relaxation of the requirements on the antialiasing/anti-imaging filter. Another advantage is the increase in SNR which occurs because of the process gain. The Sigma-Delta ADC and DAC architecture is the ultimate extension of the oversampling concept and is the architecture of choice for most voiceband and audio signal processing data converter applications [19].

3.4 Digital Filters

Because of the phase shift and distortion they introduce, analog brick-wall output filters have been abandoned by manufacturers, in favor of digital filters. A digital filter is a circuit (or algorithm) that accepts, in this case, audio samples and outputs audio samples. The values of the throughput audio samples are altered to produce filtering. When used on the output of a digital audio system, the digital filter simulates the process of ideal low-pass filtering and thus provides waveform reconstruction. Rather than suppress high-frequency images after the signal has been converted to analog form, digital filters perform the same function in the digital domain, prior to digital-to-analog (D/A) conversion. Following D/A conversion, a gentle, low-order analog filter removes remaining images that are present at very high frequencies. In most cases, finite impulse response (FIR) digital filters are used. Oversampling techniques allow a less complex digital filter design. With oversampling, additional sample values are computed by interpolating between original samples. Because additional samples are generated (perhaps two, four, or eight times as many), the sampling frequency of the output signal is greater than the input signal [2]. This type of digital filter referred to as an interpolation filter, will be discussed in a later chapter.

3.5 Noise Shaping

Another important benefit of oversampling is a decrease in audio band quantization noise. This is because the total noise power is spread over a larger (oversampled) frequency range. In particular, each doubling of the OSR lowers the quantization noise floor by 3 dB (10 log(R) = 10 log(2) = 3.01 dB). For example, in a four-times oversampling filter, data leaves the filter at a four-times frequency, and the quantization noise power is spread over a band that is four times larger, reducing its power density in the audio band to one-fourth, as shown in Figure 3.3. In this example, this yields a 6-dB reduction in noise (3 dB each time the sampling rate is doubled), which is equivalent to adding one bit of word length. Higher oversampling ratios yield corresponding lower noise (for example, eight-times yields an additional 6 dB) [2].

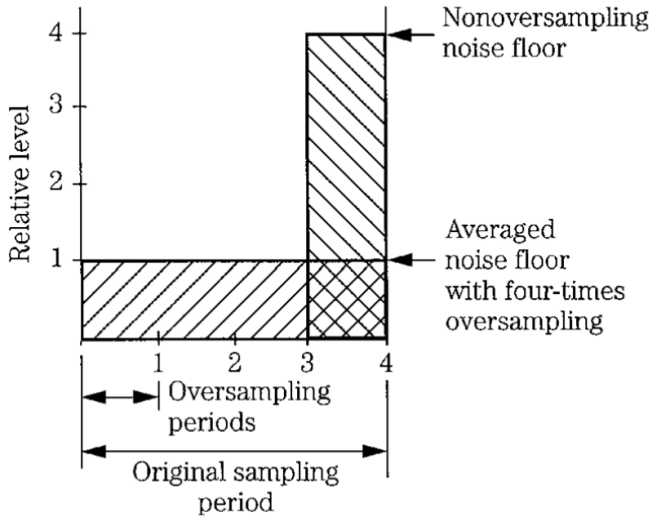


Figure 3.3: Oversampling extends quantization error over a larger band, correspondingly reducing in-band error [2].

Noise shaping, also called spectral shaping, can significantly reduce the in-band noise floor by changing the frequency response of quantization error. The technique is widely used in both analog-to-digital and digital-to-analog conversion. For example, noise shaping can be accomplished with Sigma-Delta modulation. When quantization errors are independent, the noise spectrum is white. By selecting the nature of the dependence of the errors, the noise spectrum can be shaped to any desired frequency response, while keeping the frequency response of the audio signal unchanged. Noise shaping is performed by an error-feedback algorithm that produces the desired statistically dependent errors [2].

3.6 Output Lowpass Filter

The first and last circuits in an audio digitization system are low-pass filters, known as anti-aliasing and anti-imaging filters, respectively. Although their analog designs can be almost identical, their functions are very different. In lieu of classic analog anti-imaging filters, digital filters using oversampling techniques have replaced brick-wall analog filters. However, even digital filters employ a low-order analog low-pass filter on the final output of the system. Given the criteria of the Nyquist sampling theorem, the function of the input low-pass filter is clear: it removes all frequency content above the half-sampling frequency to prevent aliasing. Similarly, a low-pass filter at the output of the digitization system removes frequency content above the half-sampling frequency. However, its function is different. This filter converts the D/A converter's output pulse-amplitude modulation (PAM) staircase to a smoothly continuous waveform, thus reconstructing the band-limited input signal. The output filter is sometimes called a smoothing filter [2].

* * *

This chapter introduced the basics of digital-to-analog conversion including DAC specifications and comparison of Nyquist rate converters with oversampling ones. It also gave a brief introduction to sigma-delta converters, highlighting their advantages. Concepts of digital filtering, noise shaping and reconstruction filtering were also discussed.

The next chapter dives into the first and most complex part of the design, the interpolating filter.

Chapter 4

Interpolating Filter

An interpolating filter (IF) plays a crucial role in the performance and power efficiency of a Sigma-Delta DAC. The following sections will discuss the basic theory and common strategies for designing and implementing IFs.

4.1 Specifications of interpolating filters

In principle, oversampling involves two processes: increasing the input sampling rate f_{si}, by inserting zeros between the existing samples (a process known as *zero-stuffing*) and filtering the signal using a digital low-pass filter to limit its bandwidth. These steps are combined into an interpolating filter. An example of $2x$ oversampling (shown for both the time and frequency domains) is illustrated in Figure 4.1.

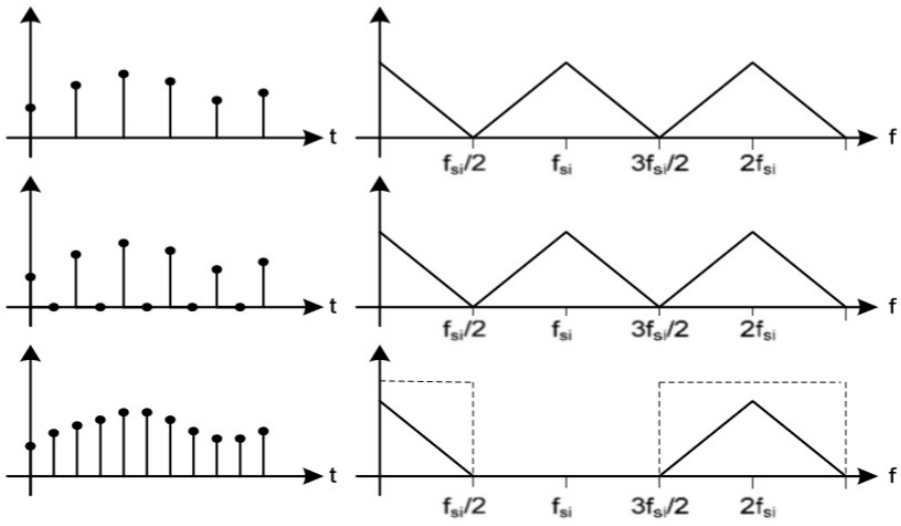


Figure 4.1: Oversampling in the time and frequency domain [8].

The ideal interpolation filter is characterized by:

$$H_L(f) = \begin{cases} 1, & 0 < f < \frac{f_{si}}{2}, \\ 0, & \frac{f_{si}}{2} < f < \frac{L f_{si}}{2}. \end{cases} \tag{4.1}$$

The impulse response is given by the inverse Fourier transform of the frequency characteristic and is hence:

$$h_L[n] = \text{sinc}\left(n\frac{1}{L}\right), \quad n \in (-\infty, \infty). \tag{4.2}$$

Real filters are designed based on the tolerated deviation from this ideal characteristic. Specified deviations include passband ripple, stopband attenuation, and transition-band length (i.e., slope of the filter) [8].

In typical DAC applications, the signal is not resampled after the interpolator, so the replicated spectra from zero-stuffing do not fold into the baseband. While this suggests that a low-pass filter could be omitted, doing so would negatively affect performance. The high-frequency energy could saturate the Sigma-Delta modulator (SDM), increase jitter sensitivity at the discrete-to-continuous time interface, and cause slewing in the analog circuitry. Unlike in an ADC, the need for low-pass filtering in an oversampled DAC is less clear-cut and depends the design of the SDM. The SDM itself generates significant high-frequency noise at its output. As long as the images are attenuated to a level well below this noise floor, they will have no impact on the discrete-to-continuous time interface or the subsequent analog filtering [8].

Sample-rate in typical high-end audio applications is often 44.1 kHz (CD-Audio). The passband is defined up to the presumably audible limit of 20 kHz and the stopband usually specified from 24.1 kHz. This gives a 4.1 kHz transition band that is symmetric around half the input sample rate. However, the transition-band is sometimes increased for higher sampling-rates, allowing lower-order filters to be used. It should also be noticed that there is usually not a uniform stopband attenuation requirement all the way to infinity. At high frequencies, the delta-sigma modulator has a high noise-floor and the damping from the analog output filter will also be significant. This reduces the requirement for image suppression at frequencies much higher than the passband and can be exploited to make more efficient filters, which will be shown later [8].

As previously mentioned, the requirements for the filter in a DAC are relaxed. The passband-deviation should be well within audible limits, while the stopband attenuation should be high enough for the high-frequency content not to affect the performance of the rest of the converter. A survey on existing high-end converters showed typical passband deviation to be in the range 0.0001dB to 0.001dB, while stopband-attenuation is found to be ranging from 75dB to 120dB [8].

As far as the IIR vs. FIR discussion goes, linear-phase symmetrical FIR filters are almost exclusively used in audio interpolators due to their ability to maintain a constant group delay [8].

4.2 Realization of interpolating filters

The next few subsections will discuss how the computational complexity of the interpolation can be reduced.

4.2.1 Interpolating filter partitioning

From the previous section it is established that the purpose of the IF is to take advantage of the increased clock frequency, and to suppress all unnecessary replicas of the signal spectrum occurring between the baseband and $OSR \cdot f_s$. It was also mentioned that the unwanted sidebands need not be totally erased, since truncation noise will be introduced in their place anyway in the noise-shaping loop. In principle, it is possible to raise the sampling frequency immediately to $OSR \cdot f_s$ and then to carry out all filtering at this elevated clock rate. However, this would require all digital circuitry to function at high speed, and it would hence dissipate an unnecessarily large amount of power. It would also generate more digital activity, and thus more digital noise, than necessary. Therefore, it is preferable to perfom the increasing of the clock frequency and the filtering in parallel steps, with most of the signal processing performed at a low clock rate [1].

The way to reduce the number of computations is to use several filters in cascade. This is illustrated in Figure 4.2 with $8x$ OSR low-pass filtering being performed using three $2x$ OSR filters in cascade. As can be seen, the transition-band can be increased for each subsequent filter, as the aliases from the previous filtering are spaced further and further apart [8].

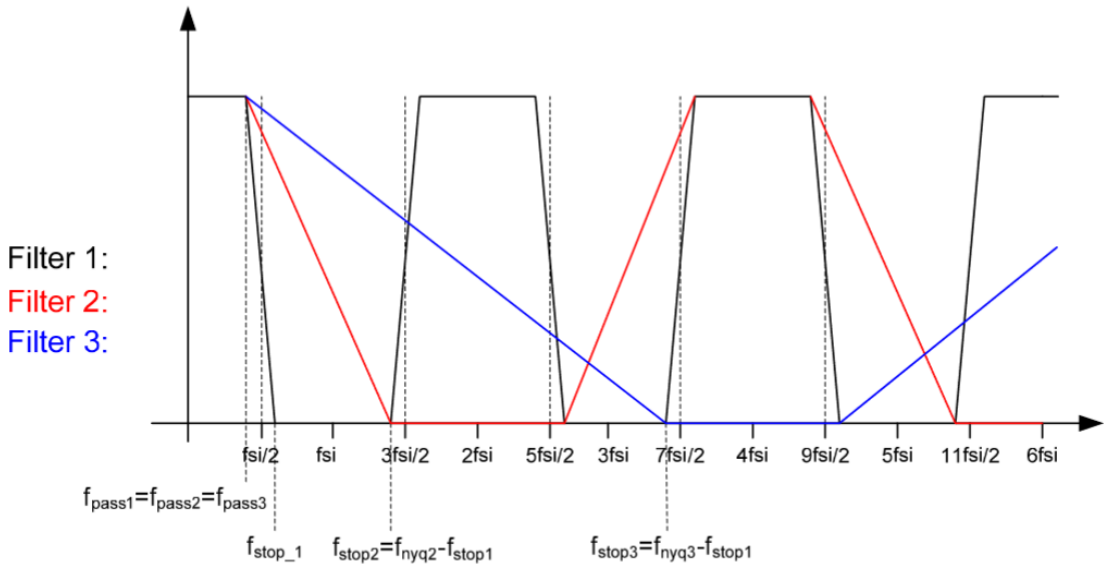


Figure 4.2: Interpolation with cascade of lower OSR filters [8].

4.2.2 Interchange of filtering with upsampling

Before diving into polyphase filter structures, an important identity is to be introduced. This *multirate identity* states that the order of filtering and upsampling can be interchanged if the impulse response of the filter is upsampled [12].

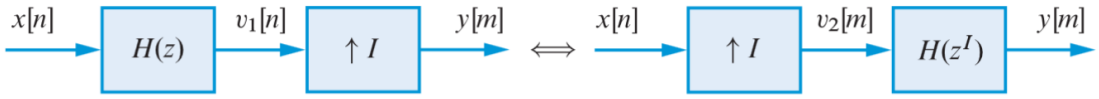


Figure 4.3: Multirate identity: two equivalent systems for upsampling. It is possible to interchange the filter with an expander if the filter is properly modified [12].

The output of the system on the left in Figure 4.3 is given by

$$Y(z) = V_1\left(z^I\right) = H\left(z^I\right) X\left(z^I\right). \tag{4.3}$$

The output of the system on the right in Figure 4.3 is

$$Y(z) = H\left(z^I\right) V_2(z) = H\left(z^I\right) X\left(z^I\right). \tag{4.4}$$

Comparison of 4.3 and 4.4 shows that the two structures are equivalent.

4.2.3 Polyphase structures for interpolation

Polyphase filter structures are widely used to simplify the implementation of decimators and interpolators. For an FIR filter with length $N = ML$, the polyphase decomposition breaks the impulse response into M subsequences of length L. The basic idea is best illustrated with a simple example. For $N = 6$ and $M = 2$, even and odd terms are grouped as follows [12]:

$$H(z) = h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[4]z^{-4} + h[5]z^{-5} \tag{4.5}$$

$$= \left(h[0] + h[2]z^{-2} + h[4]z^{-4}\right) + z^{-1}\left(h[1] + h[3]z^{-2} + h[5]z^{-4}\right), \tag{4.6}$$

If the following sub-filters are defined:

$$P_0(z) \triangleq h[0] + h[2]z^{-1} + h[4]z^{-2}, \tag{4.7}$$

$$P_1(z) \triangleq h[1] + h[3]z^{-1} + h[5]z^{-2}, \tag{4.8}$$

$H(z)$ can be expressed as follows:

$$H(z) = P_0\left(z^2\right) + z^{-1}P_1\left(z^2\right). \tag{4.9}$$

For $M = 3$ the following decomposition is obtained:

$$H(z) = P_0\left(z^3\right) + z^{-1}P_1\left(z^3\right) + z^{-2}P_2\left(z^3\right), \tag{4.10}$$

where the polyphase components are given by:

$$P_0(z) \triangleq h[0] + h[3]z^{-1}, \tag{4.11}$$

$$P_1(z) \triangleq h[1] + h[4]z^{-1}, \tag{4.12}$$

$$P_2(z) \triangleq h[2] + h[5]z^{-1}. \tag{4.13}$$

It is noted that when N, the length of the impulse response, is a composite number, multiple polyphase decompositions exist. To avoid potential confusion, M should be included in the notation. In general, the impulse response of the k-th sub-filter is obtained by downsampling the shifted sequence $h[n+k]$ by a factor M, that is:

$$p_k[n] \triangleq h[nM+k], \quad k = 0, 1, \ldots, M-1. \tag{4.14}$$

Therefore, $H(z)$ can be expressed as:

$$H(z) = \sum_{k=0}^{M-1} z^{-k} P_k\left(z^M\right), \tag{4.15}$$

where

$$P_k(z) = \sum_{n=0}^{L-1} p_k[n] z^{-n}. \tag{4.16}$$

Using (4.10) for $M = 3$, we note that the z-transform of the output sequence $y[n]$ can be expressed into two equivalent forms as follows [12]:

$$\begin{aligned} Y(z) &= H(z)X(z) \\ &= P_0(z^3)X(z) + z^{-1}P_1(z^3)X(z) + z^{-2}P_2(z^3)X(z), \qquad (4.17) \\ &= P_0(z^3)X(z) + z^{-1}\left\{P_1(z^3)X(z) + z^{-1}[P_2(z^3)X(z)]\right\}. \qquad (4.18) \end{aligned}$$

The first expression (4.17) leads to the direct form polyphase structure shown in Figure 4.4 (a); the second expression (4.18) leads to the transposed form polyphase structure shown in Figure 4.4 (b).

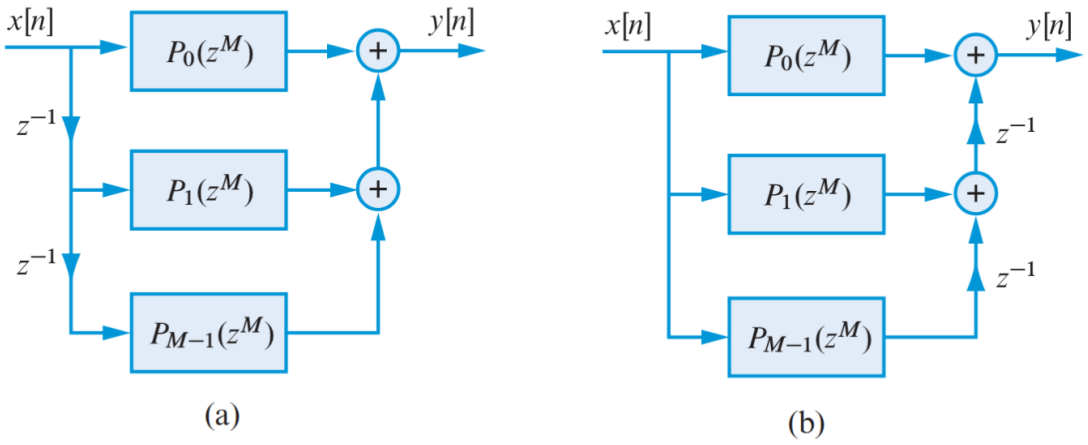


Figure 4.4: (a) Realization of the direct form polyphase structure, and (b) the transposed form polyphase structure; both structures are shown for M = 3 [12].

The most important application of polyphase structures is in implementation of computationally efficient decimators and interpolators. A polyphase structure can be obtained for the interpolation system shown in Figure 4.5. Replacement of the filter $H(z)$ by the polyphase structure in Figure 4.4 (b) for $M = I$, yields the polyphase structure in Figure 4.6 (a). If the multirate identity for upsampling is applied next, the structure in Figure 4.6 (a) takes the form shown in Figure 4.6 (b). The output of each polyphase filter is given by [12]:

$$p_k[n]h[nI+k], \quad k=0,1,\ldots,I-1, \tag{4.19}$$
$$y_k[n]p_k[n] * x[n], \tag{4.20}$$
$$y[m]y_k[n], \quad m=nI+k. \tag{4.21}$$

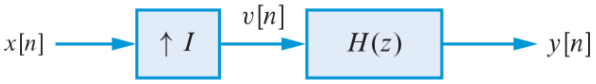


Figure 4.5: Interpolation system [12].

The upsampling and delay operations in 4.6 (b) can be replaced by the commutator structure shown in Figure 4.7. It is noted that while the filtering operation takes place at the lower input sampling rate, the commutator picks up sequential samples from the filtered sequences $y_0[n], y_1[n], \ldots, y_{L-1}[n]$, at the higher output sampling rate. The commutator starts with $y_0[n]$ and continues counterclockwise; for each input sample it performs one full rotation to collect L interpolation samples from the output of the polyphase bank.

The advantage of the polyphase interpolation structure is that the filter operates at the lower input sampling rate; the number of computations per input sample is the same as for the implementation of the direct form [12].

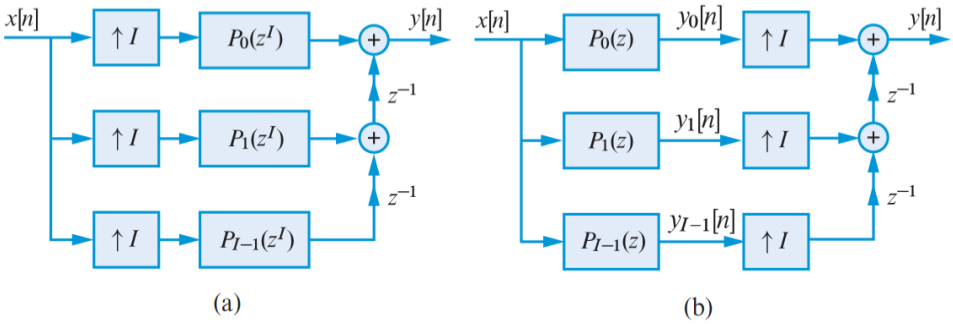


Figure 4.6: Polyphase interpolation structure before (a) and after (b) the application of the multirate identity for upsampling [12].

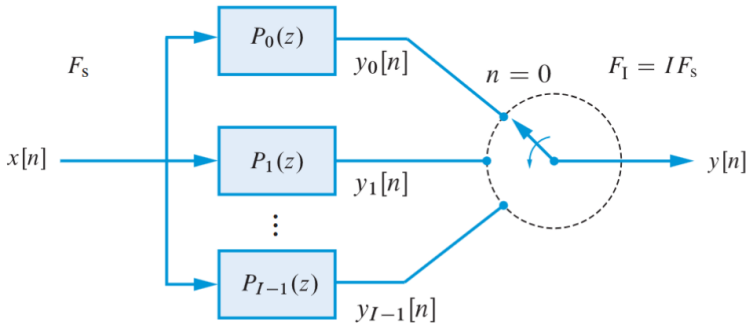


Figure 4.7: Polyphase interpolator with a commutator instead of delays [12].

4.2.4 Half-Band Filters

Decimation and interpolation by a factor of two require low-pass filters with cutoff frequency $\omega_c = \pi/2$, that is, ideal half-band filters [12]. The impulse response of this ideal filter is:

$$h[n] = \begin{cases} \frac{\omega_c}{\pi} \frac{\sin \omega_c(n-\alpha)}{\omega_c(n-\alpha)}, & n - \alpha = \pm 2, \pm 4, \ldots, \\ \frac{1}{2}, & n = \alpha, \\ 0, & \text{otherwise.} \end{cases} \tag{4.22}$$

Assuming the simplest case, that is, $\alpha = 0$, noncausal zero-phase filters with a real frequency response function $H\left(e^{j\omega}\right)$ are considered. The following expression applies:

$$h[0] = \frac{1}{2}, \quad h[2n] = 0, \quad n = \pm 1, \pm 2, \ldots. \tag{4.23}$$

Thus, the polyphase representation of (4.15) is given by:

$$H(z) = P_0\left(z^2\right) + z^{-1} P_1\left(z^2\right) = \frac{1}{2} + z^{-1} P_1\left(z^2\right), \tag{4.24}$$

where $P_0(z) = \sum_n h[2n] z^{-n}$ and $P_1(z) = \sum_n h[2n+1] z^{-n}$. In general, any filter with $h[-n] = h[n]$ or $H\left(e^{-j\omega}\right) = H\left(e^{j\omega}\right)$ that satisfies (4.23) is called a half-band filter. Figure 4.8 shows the impulse response and frequency response of an M th-order FIR half-band filter.

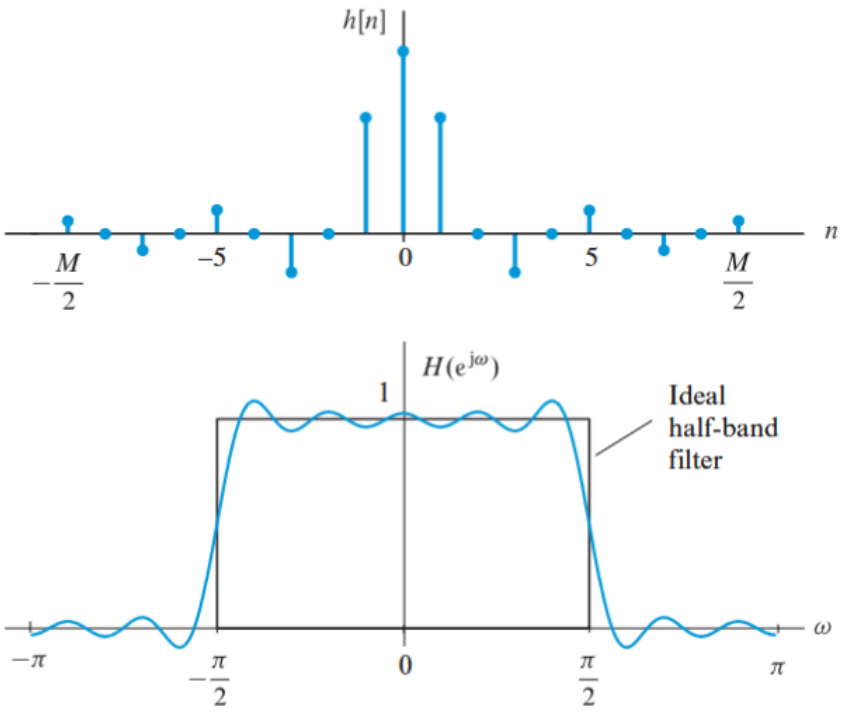


Figure 4.8: Impulse response and frequency response of a half-band FIR filter with order $M = 18$. Notice the odd symmetry of $H\left(e^{j\omega}\right)$ about $\omega = \pm\pi/2$ [12].

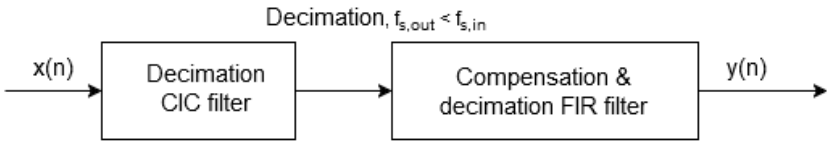
First, it is noted that the $M/2$ zeros in the impulse response reduce the amount of computation by almost one-half. Second, it is noted that $H\left(e^{j\omega}\right)$ has an odd symmetry to $\omega = \pm\pi/2$. The implications of this symmetry for a practical low-pass filter are: i)

the peak errors are equal, and ii) the band-edges are symmetric with respect to $\pi/2$ [12]. Lastly, since the coefficients of the second polyphase are symmetric, memory is saved by only storing half of them.

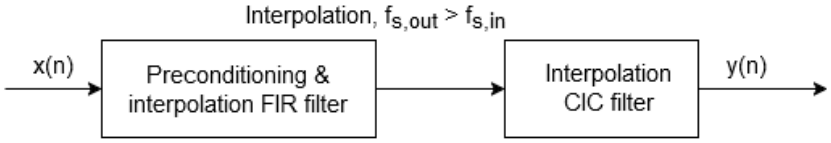
4.2.5 Cascaded Integrator-Comb Filters

Cascaded integrator-comb (CIC) filters are computationally efficient implementations of narrowband low-pass filters and, as such, are used in conjunction with hardware implementations of decimation and interpolation in modern communication systems [11].

CIC filters are well-suited for anti-aliasing filtering prior to decimation (sample rate reduction), as shown in Figure 4.9a; and for anti-imaging filtering for interpolated signals (sample rate increase), as in Figure 4.9b. Both applications are associated with very high data rate filtering, such as hardware quadrature modulation and demodulation in modern wireless systems, and Sigma-Delta ADCs and DACs [11].



(a) decimation



(b) interpolation

Figure 4.9: CIC filter applications[11].

Because their frequency magnitude response envelopes are $\sin(x)/x$-like, CIC filters are typically followed, or preceded, by higher-performance linear-phase low-pass tapped-delay line FIR filters whose tasks are to compensate for the CIC filter's non-flat passband. That cascaded-filter architecture has valuable benefits. For example, with decimation, narrowband low-pass filtering can be attained at a much reduced computational complexity compared to a single low-pass FIR filter due to the initial CIC filtering. In addition, the follow-on FIR filter operates at reduced clock rates, minimizing power consumption in high-speed hardware applications. An added bonus of using CIC filters is that they do not require multiplications [11].

4.2.6 Compensation/Preconditioning FIR Filters

In typical decimation and interpolation filtering applications, reasonably flat passband performance and a narrow transition region are desired. These properties are not inherently provided by CIC filters, which exhibit drooping passband gains and wide transition regions. To address this issue, particularly in decimation, a compensation nonrecursive FIR filter is applied after the CIC filter, as shown in Figure 4.9 (a), to narrow the output bandwidth and flatten the passband gain [11].

The compensation FIR filter's frequency magnitude response is ideally an inverted version of the CIC filter passband response similar to that shown by the dashed curve in Figure 4.10 (a) for a simple three-tap FIR filter. With the dotted curve representing the uncompensated passband droop of a first-order $R = 8$ CIC filter, the solid curve represents the compensated response of the cascaded filters. If either the passband bandwidth or CIC filter order increases, the correction becomes more robust, requiring more compensation FIR filter taps. An example of this situation is shown in Figure 4.10 (b), where the dotted curve represents the passband droop of a third-order $R = 8$ CIC filter and the dashed curve, taking the form of $[x/\sin(x)]^3$, is the response of a 15-tap compensation FIR filter [11].

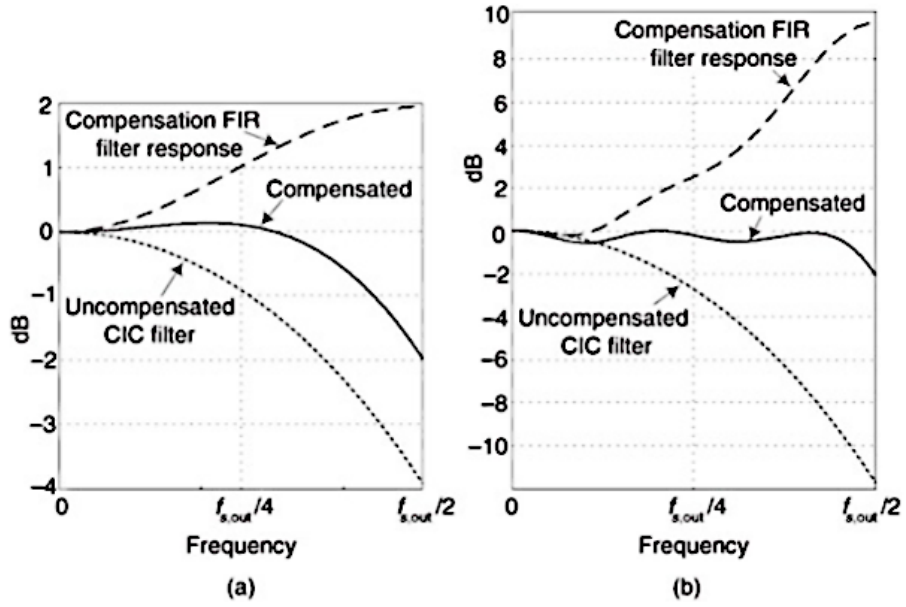


Figure 4.10: Compensation FIR filter magnitude responses: (a) with a first-order decimation CIC filter; (b) with third-order decimation CIC filter [11].

A wideband correction also means signals near $f_{s,\text{out}}/2$ are attenuated with the CIC filter and then must be amplified in the correction filter, adding noise. As such, practitioners often limit the passband width of the compensation FIR filter to roughly one-fourth the frequency of the first null in the CIC filter response [11].

The dashed curves in Figure 4.10 represent the frequency magnitude responses of compensating FIR filters within which no sample rate change takes place. If a compensating FIR filter were designed to provide an additional decimation by two, its frequency magnitude response would look similar to that in Figure 4.11, where $f_{s,\text{in}}$ is the compensation filter's input sample rate [11].

A decimating CIC filter is merely a very efficient recursive implementation of a moving average filter with NR taps, whose output is decimated by R. Likewise, the interpolating CIC filter is insertion of $R - 1$ zero samples between each input sample followed by an NR-tap moving average filter running at the output sample rate $f_{s,\text{out}}$. The cascade implementations in Figure 4.9 result in total computational workloads far less than using a single FIR filter alone for high sample rate change decimation and interpolation. CIC filter structures are designed to maximize the amount of low sample rate processing to minimize power consumption in high-speed hardware applications. Again, CIC filters require no multiplications; their arithmetic is strictly additions and subtractions giving

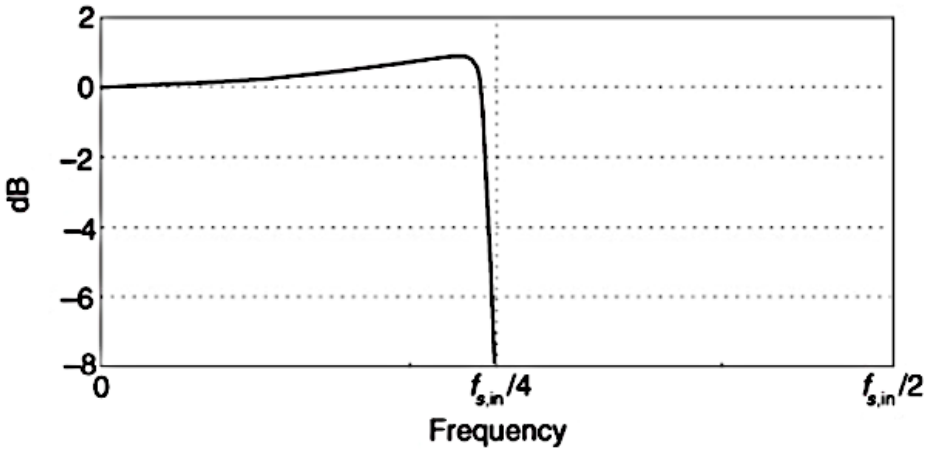


Figure 4.11: Frequency magnitude response of a decimate-by-two compensation FIR filter [11].

them high performances [11].

4.2.7 Coefficient sharing

The linear-phase FIR filter are characterized by symmetry, allowing the number of stored coefficients to be reduced by half. Since the coefficients on each side of the centre tap are equal, the number of required multiplications is cut in half as illustrated in Figure 4.12. While this optimization reduces the number of required multiplications, it is important to note that the number of additions and delay elements remains unchanged [8].

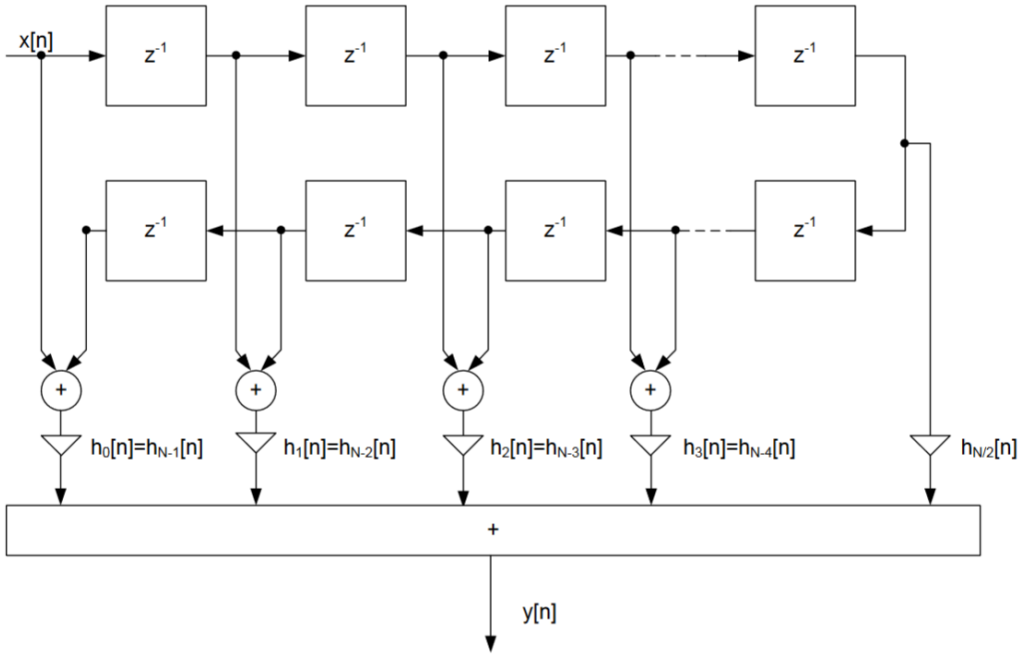


Figure 4.12: Symmetrical FIR filter with coefficient sharing [8].

4.2.8 Coefficient quantization

Efficient partitioning and realization of interpolators must also account for the precision of filter coefficients in practical implementations. Infinite-precision filters with infinite word lengths are not feasible, so coefficients must be quantized, balancing accuracy with available chip area or DSP resources. The chosen word length significantly impacts resource usage, even for a fixed structural complexity. To prevent uncontrolled word growth within the filter, internal variables may also require quantization. Filter behavior under quantization depends heavily on the filter structure. Typically, coefficients are truncated to a fixed number of bits, and their word length is further influenced by the bit budget for internal data paths during multiplication [8].

Fixed-point vs. floating-point arithmetic

The choice between fixed-point and floating-point arithmetic depends on the application's precision requirements and available computational resources. While floating-point provides higher precision and dynamic range, it demands significantly more resources and complexity, making it less suitable for hardware-constrained environments. Fixed-point arithmetic, on the other hand, is more resource-efficient and is commonly implemented using either truncation or rounding. Truncation involves discarding the least significant bits (LSBs) that exceed the specified word length, whereas rounding adjusts the value to the nearest representable number. Although truncation is simpler to implement, it can introduce bias, while rounding generally offers better accuracy at the cost of slightly higher complexity [8].

Consequences of coefficient quantization

For FIR filters, general approximations of quantized coefficient effects can easily be analyzed. An Nth-order ideal FIR transfer function is given by:

$$H(z) = \sum_{n=0}^{N} h[n] z^{-n}. \tag{4.25}$$

For a filter with quantized coefficients, the transfer function becomes:

$$\hat{H}(z) = \sum_{n=0}^{N} \hat{h}[n] z^{-n} = \sum_{n=0}^{N} (h[n] + e[n]) z^{-n} = H(z) + E(z). \tag{4.26}$$

This means that the FIR filter with quantized coefficients can be modeled as two parallel filters: the ideal filter and a filter with coefficients consisting of the quantization errors, respectively [8]. From taking a look at the frequency response:

$$\hat{H}\left(e^{j\omega}\right) = H\left(e^{j\omega}\right) + E\left(e^{j\omega}\right), \tag{4.27}$$

the error must be bounded by:

$$\left|E\left(e^{j\omega}\right)\right| \leq \sum_{n=0}^{N} |e[n]|. \tag{4.28}$$

For a uniformly distributed error, where $-\Delta/2 \leq e[n] < \Delta/2$, the normal method of analyzing quantization errors shows that the error variance is bounded by [8]:

$$\sigma_E^2(\omega) \leq \frac{(2N+1)\Delta^2}{12}. \tag{4.29}$$

Figure 4.13 shows a FIR filter without coefficient quantization and with 16-bit fixed-point quantization. It can be observed that the stopband damping is reduced for about 20dB.

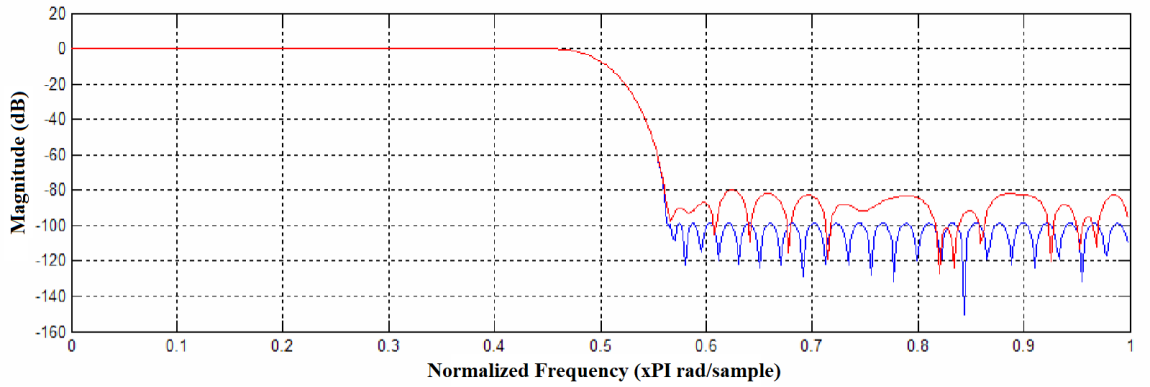


Figure 4.13: Frequency response, 16-bit coefficient quantization [8].

4.3 Interpolating filter implementation

The filter implementation depends on the target hardware, available resources and operating frequency. The following text discusses two types of implementations of digital filters.

4.3.1 Direct implementation

The most straightforward approach to implementing an interpolating filter is to realize it in direct form, as illustrated in Figure 4.14. This method is advantageous for its ability to operate at high speeds. However, it comes with the drawback of high resource usage due to the large number of delay elements, multipliers, and adders required. In hardware implementations, resource usage can be significantly reduced with a multiplier-free realization. By quantizing the coefficients, multipliers can be replaced with shift-and-add operations, which are more resource-efficient. However, the high number of registers and adders needed for the design can still consume substantial hardware area [8].

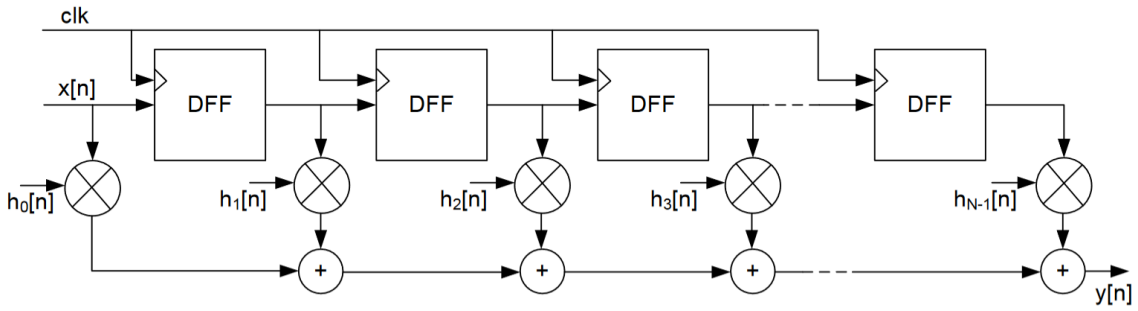


Figure 4.14: Direct implementation of filter transfer function [8].

4.3.2 MAC implementation

A popular approach to realizing high-order FIR filters is through the multiplier-accumulator (MAC) implementation. In this method, the filter coefficients are stored in a read-only memory (ROM) and sequentially multiplied with input samples. The products are accumulated, and once all summed, the convolution result is sent to the output. Input data is stored in a rotational register, where new samples are shifted in for each complete convolution cycle, as shown in Figure 4.15.

The MAC implementation requires two storage registers of length N for an N-th order filter. However, the circuit's complexity is greatly reduced due to only using a single multiplier and a single adder. These components, along with the accumulator, must operate at N times the input sample rate to complete a full convolution for each input sample. Consequently, the MAC implementation is less suitable for high-speed applications but offers an area-efficient solution for low sample rate applications. This approach is particularly popular in audio interpolation and filtering [8].

* * *

This chapter discussed in detail the implementation strategies for the interpolating filter, making sure the design meets the set specifications and resource constraints.

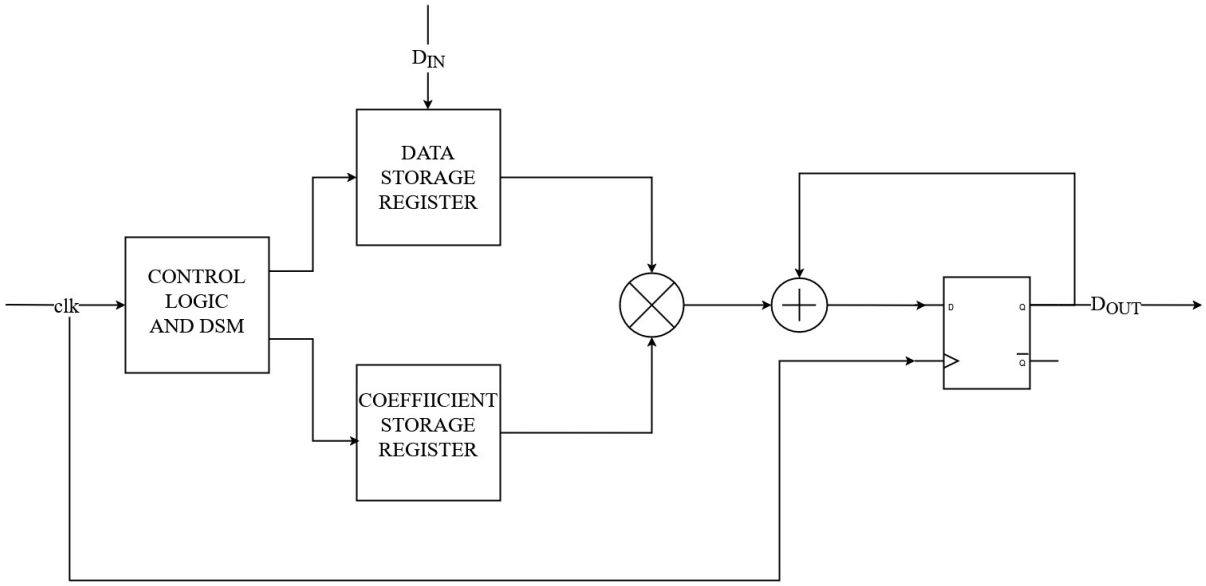


Figure 4.15: MAC FIR-filter implementation [8].

The next chapter dives into the second integral part of the design, the Sigma-Delta modulator.

Chapter 5

Sigma-Delta Modulation

Sigma-Delta conversion methods are widely used in digital audio products. These techniques are extremely competitive in both A/D and D/A applications because they obviate the need for analog brick-wall filters. Although sigma-delta methods use familiar techniques such as oversampling, highly sophisticated processing is required to implement noise shaping and thus decrease the high in-band noise levels otherwise present in sigma-delta conversion. A variety of sigma-delta A/D and D/A architectures have been devised, with different algorithms and orders of noise shaping.

5.1 Delta Modulation

Delta modulation and Sigma-Delta modulation (also called Delta-Sigma) were developed in the 1940s and 1960s, respectively, for voice telephony applications [2]. Limitations prohibited their use in high-quality music applications until the emergence of high-speed digital signal processing techniques in the 1980s [2], that improved audio performance. Differential pulse-code modulation (DPCM) is a technique in which the derivative of the signal is quantized [2]. When signal changes between sample periods are small, the quantizer's word length can be reduced. With very high oversampling rates, the changes between sample periods are made very small, thus the quantizer can be reduced to one bit. A one-bit DPCM coder is known as a delta modulator (DM). In other words, DM codes the differences in the signal amplitude, its slope, instead of the signal amplitude itself. In contrast, a Sigma-DPCM technique places an integrator at the input to the quantizer. Instead of coding the slope of the signal, Sigma-DPCM, like PCM, codes its amplitude. However, with Sigma-DPCM, the signal can be quantizing to only one or a few bits; both implementations are generally known as Sigma-Delta modulation. As with Delta modulation, Sigma-Delta modulation requires high oversampling rates [2].

A Delta-modulation encoder and decoder are shown in Figure 5.1. This is known as a single integration modulator. The analog input signal is compared to the integrated output pulses and the delta (difference) signal is applied to the quantizer. The quantizer generates a positive pulse when the difference signal is negative, and a negative pulse when the difference signal is positive. This difference error signal moves the integrator step by step closer to the present value input, tracking the derivative of the analog input signal. The integrator's output is a past approximation to the input, thus the coder operates similarly to other integrating feedback loops such as phase-locked loops.

A Delta-modulation decoder consists of an integrator and a low-pass filter. When the one-bit pulses are integrated using a time constant that is long compared to the sample

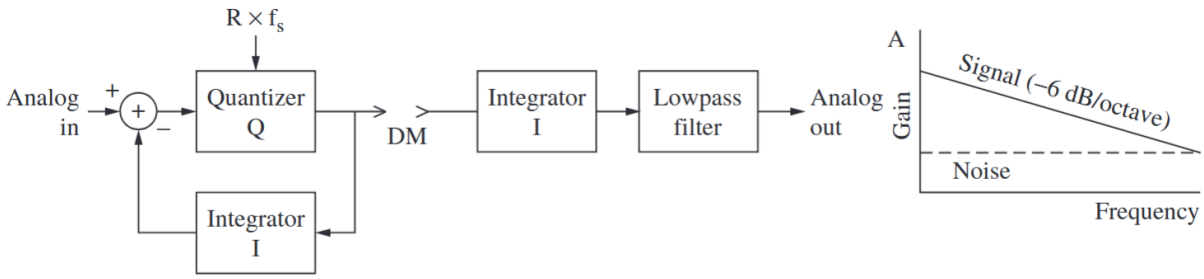


Figure 5.1: Delta modulation (DM) yields a signal that falls with frequency [2].

period, a step waveform is produced. An output analog waveform is produced by low-pass filtering this step waveform. Requantization error at the output of the integrator is white. As in PCM, oversampling decreases the error level by 3 dB for every factor of two oversampling. The dynamic range can be improved, that is, the requantization error made smaller, by making the delta (difference or step size) value smaller. The limit to which the delta can be reduced is given by the maximum derivative of the signal. The coded signal amplitude decreases at 6 dB/octave; thus, $Signal/Noise$ decreases as the signal frequency increases.

The maximum derivative occurs at the highest signal frequency and amplitude, with exceeding this limit resulting in slope overload distortion. The success of DM is reliant on assumptions about the encoded signal's nature, with its dynamic range depending on the signal spectrum. DM performs well only when the signal exhibits low-pass characteristics and correlated patterns at low levels, restricting single-integration DM applications. Bit reduction at a high sampling frequency is required to output a one-bit signal from a high-bit source which greatly degrades the signal's dynamic range [2].

5.2 Sigma-Delta Modulation

Sigma-Delta modulation (SDM) was developed to overcome the limitations of Delta modulation. Sigma-Delta systems quantize the delta (difference) between the current signal and the sigma (sum) of the previous difference. A first-order (single integration) Sigma-Delta modulation encoder and decoder are shown in Figure 5.2. An integrator is placed at the input to the quantizer; signal amplitude is constant with increasing frequency. The input to the midrise quantizer is the integral of the difference between the input and the quantized output as applied via negative feedback. If the input signal in one sample period is greater than the value accumulated in the feedback loop over previous samples, the converter outputs a "1." Otherwise, if it is lower, the output is a "0." In other words, the difference between the input signal and the accumulated error is quantized. When the error is sufficiently large, the quantizer changes state to reduce the error. Thus, the difference between the input signal and the output signal approaches zero; the average value of the output approximates the input. There is little *dc* error in the output signal. However, the frequency spectrum of the quantizing error rises with increasing frequency (6 dB/octave) [2].

The integrator forms a low-pass filter on the difference signal thus providing low-frequency feedback around the quantizer. This feedback results in a reduction of quantization noise at low (in-band) frequencies. Unlike PCM and DM, the noise floor is not

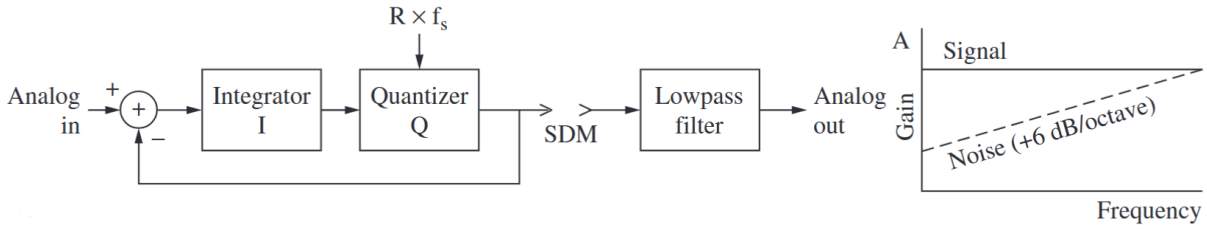


Figure 5.2: Sigma-delta modulation (SDM) yields a noise floor that rises with frequency [2].

flat, but shaped by a first-order high-pass characteristic. Like PCM, Sigma-Delta modulation quantizes the signal amplitude directly, and not its derivative as in DM. Thus the maximum quantizer range is determined by the maximum signal amplitude and is not dependent on signal spectrum. As with delta modulation, to achieve high resolution, high oversampling rates are required. For example, with an audio band of 24 kHz and 64 times oversampling, the internal sampling frequency rises to 3.072 MHz, thus quantization noise is spread from dc to 1.536 MHz. However, because the quantizer has only two states, the quantization error and the resulting noise level are high. To overcome this, Sigma-Delta modulators add noise shaping to move the noise power to higher, out-of-band frequencies. The noise is shaped by the inverse of the loop transfer function. When a low-pass filter is placed in the loop, the noise spectrum increases with frequency. In many designs, a multi-bit quantizer is used within the loop and a low-bit PCM signal is coded. The dynamic range increases with 6 dB for every quantization bit. As in one-bit designs, the noise floor increases by 6 dB/octave in multi-bit designs, and noise shaping is required[2].

A one-bit Sigma-Delta modulation decoder theoretically requires only a low-pass filter to decode the signal, to remove high-frequency (out-of-band) components. In other words, it averages the output signal to produce an analog waveform. An integrator is not needed in the decoder (as in DM) because the signal's amplitude is coded, not its slope. In multi-bit designs, a D/A converter is needed in the decoder to decode the low-bit PCM signal[2].

5.3 Analysis of a First-Order Sigma-Delta Modulator

The signal and noise transfer functions for a first-order Sigma-Delta modulator are analyzed in Figure 5.3.

X(z) is the z-transform of the input sequence and Y(z) represents the output sequence. The quantization error is represented as white noise N(z). For a zero noise source, the transfer function shows a low-pass characteristic. For a zero signal, the transfer function shows a high-pass characteristic. In other words, as the loop integrates the difference between the input signal and the sampled signal, it low-pass filters the signal and high-pass filters the noise. If the system is designed so the signal's frequency content is less than the filter's cutoff frequency, the signal will not be affected. Given a first-order noise-shaping loop, with a maximum amplitude sinusoidal input, the maximum signal-to-error ratio will be:

$$S/E = 6.02(N + 1.5) - 3.41dB \tag{5.1}$$

where N is the number of quantization bits and L is the number of octaves of oversampling [2].

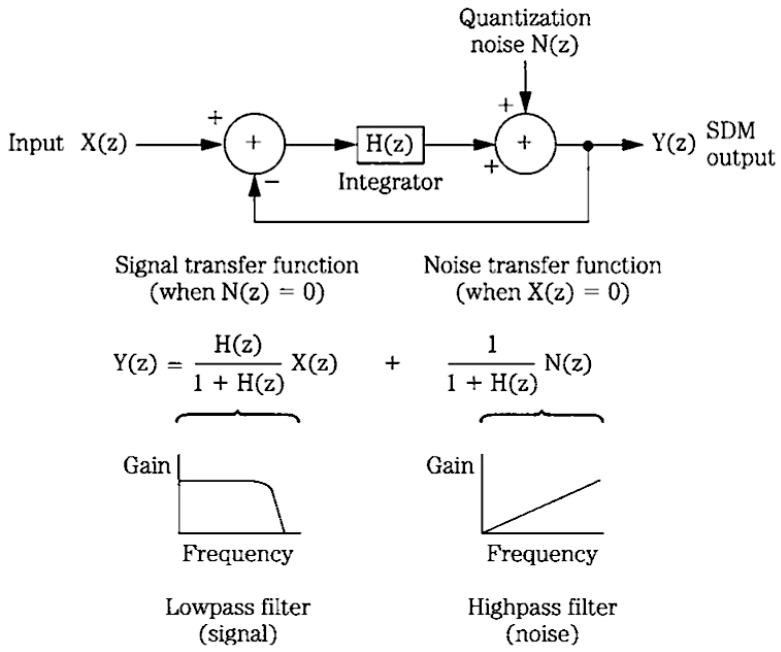


Figure 5.3: A z-transform analysis of a Sigma-Delta modulator [2].

The performance of a Sigma-Delta converter relies on both oversampling and its noise-shaping characteristic. The quantization noise floor falls from 0 to $f_s/2$ Hz and is quite high, as shown in Figure 5.4. Quantization noise is reduced with an R oversampling ratio because noise is spread over a 0-Hz to $f_a/2$-Hz spectrum where $f_a = R \times f_s$ Hz. With Sigma-Delta noise shaping, in-band noise (0 to $f_s/2$) is further decreased, and out-of-band noise is increased [2].

Figure 5.5 summarizes the mathematical basis of first-order Sigma-Delta noise shaping. Quantization noise is assumed to be random, and the quantizer is modeled as an additive noise source. Note that the $(1 - z^{-1})$ factor doubles quantized noise power. However, the same factor also shifts the noise to higher frequencies. This Sigma-Delta modulator forms the basis for many A/D and D/A one-bit and multi-bit converters [2].

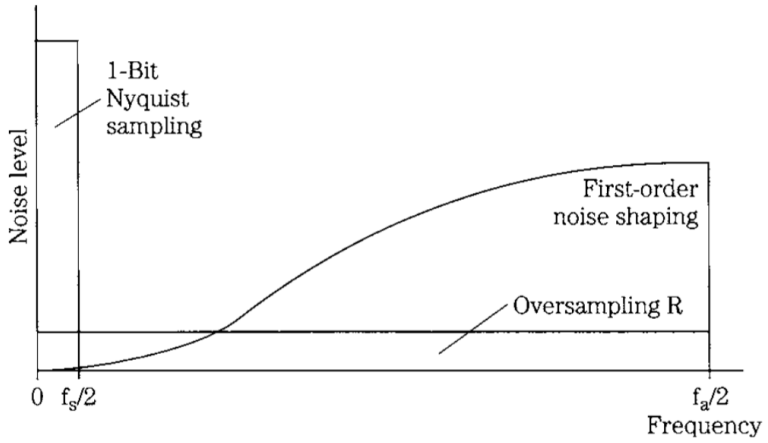


Figure 5.4: With one-bit conversion, quantization noise is quite high. In-band noise is reduced with oversampling. With noise shaping, quantization noise is shifted away from the audio band, further reducing in-band noise [2].

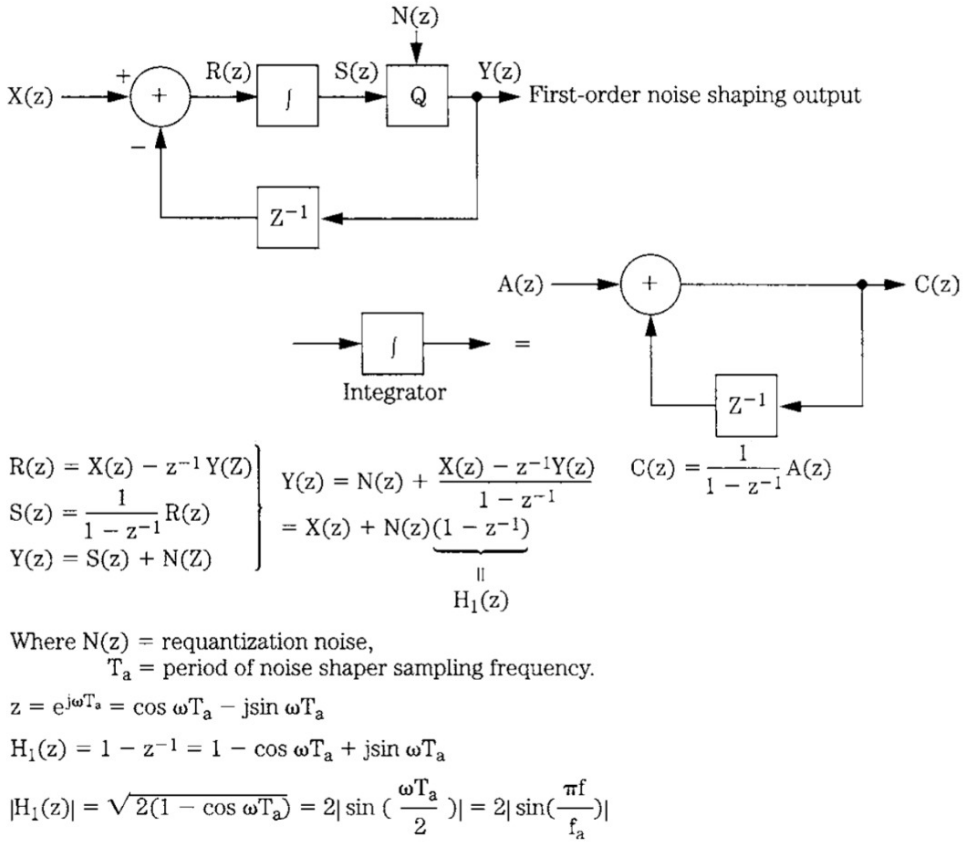


Figure 5.5: Analysis of a first-order Sigma-Delta noise shaper [2].

5.4 Higher-Order Noise Shaping

As noted, first-order (single integration) Sigma-Delta modulation is not satisfactory for high-fidelity audio performance. Higher-order loops further decrease in-band quantization noise, with the penalty of increased total noise power. For example, a second-order loop would yield:

$$S/E = 6.02(N + 2.5L) - 11.14\,\text{dB} \tag{5.2}$$

This provides a benefit of 2.5 bits/octave, with a fixed-noise penalty approximately equal to two equivalent bits.

The input/output characteristic of a basic Sigma-Delta noise shaper of nth order is [2]:

$$Y(z) = X(z) + (1 - z^{-1})^n N(z) \tag{5.3}$$

where

$Y(z)$ – the noise-shaped output,
$X(z)$ – the input signal,
n – the order of the differentiation,
$N(z)$ – the quantization noise (assumed to be white).

This characteristic can be theoretically composed of n cascaded digital differentiators. As n increases, the slope in frequency of the shaping function increases, making it more effective in suppressing low-frequency noise. However, the out-of-band noise could overly burden subsequent analog filters. A successful noise-shaping circuit thus seeks to balance a high oversampling rate with noise-shaping order to reduce in-band noise and shift it away from the audible range. Higher-order noise-shaping loops can remove even more in-band noise overall, but relatively more noise is present near the Nyquist frequency. Hence, these algorithms are more effective at high oversampling rates so there is more spectral space between the highest audio frequency and the Nyquist frequency. This allows the use of very simple analog low-pass filters [2].

Using first-order and higher-order noise-shaping algorithms, a series of noise-shaping curves can be generated, as shown in Figure 5.6. As higher orders of noise shaping are used, the in-band noise level is decreased. The frequency response curves described by Sigma-Delta noise-shaping equal a unity value at $f_a/6$ Hz where f_a is the noise-shaping oversampling frequency. Noise is reduced only for $f < f_a/6$ Hz, and increased for $f_a/6 < f < f_a/2$ Hz. The noise level reaches a maximum at $f_a/2$ Hz. As the oversampling rate is increased, the portion of the noise curve in the audio band is relatively reduced. Although the shape of the noise curve remains the same, high oversampling rates relatively decrease in-band noise [2].

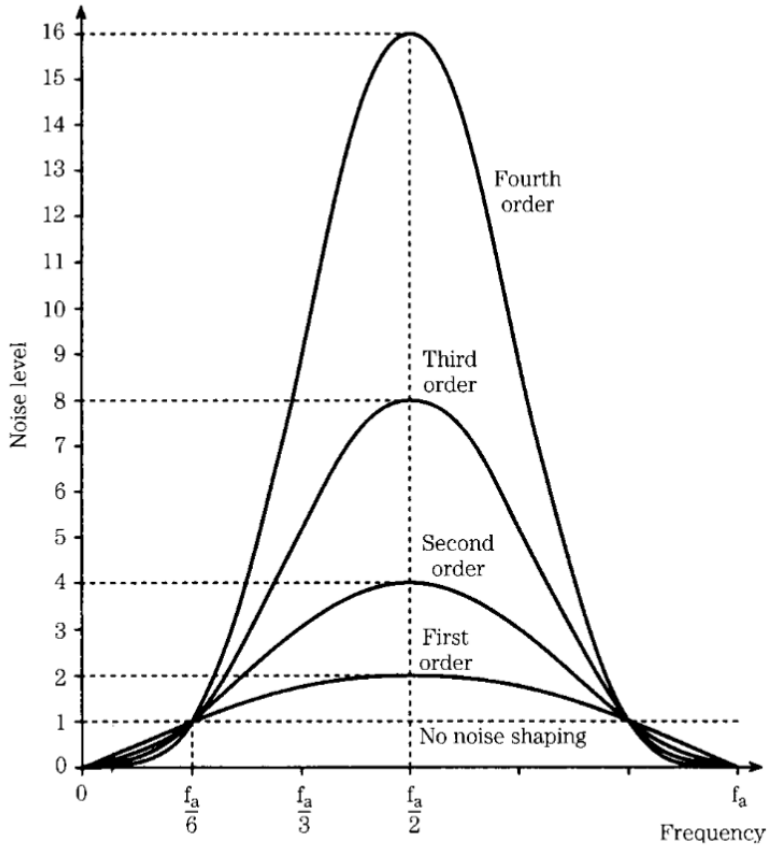
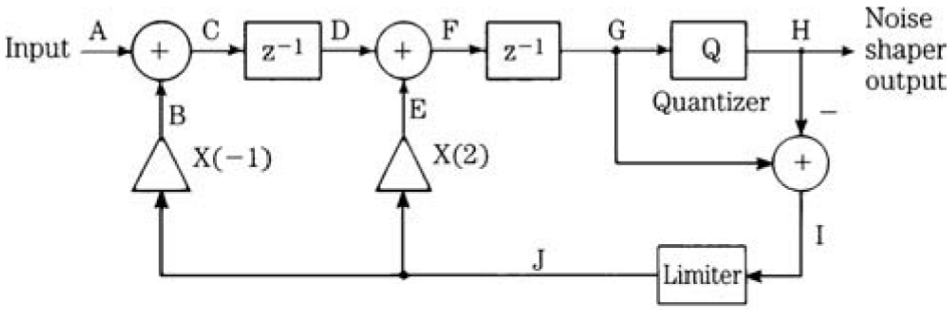


Figure 5.6: Higher orders of noise shaping result in more pronounced shifts in requantization noise [2].

5.5 One-Bit D/A Conversion with Second-Order Noise Shaping

Higher-order Sigma-Delta modulation loops offer wider dynamic range and overall better performance, but loops greater than second-order can be unstable. The quantizer is nonlinear because its effective "gain" inversely varies with the input level. A large input signal can overload the loop, reducing the gain of the quantizer, thus causing instability that will persist even after the signal is withdrawn. For example, a powering-up transient could cause a converter to oscillate. Converters thus sense instability by counting the number of consecutive ones or zeroes in the bitstream, and reset when necessary. Virtually all Sigma-Delta converters use at least a second-order modulator loop[2].



	A	B	C	D	E	F	G	H	I	J
t_0	0.5	−0.4	0.1	0.5	0.8	1.3	−0.6	−1	0.4	0.4
t_1	0.5	−0.3	0.2	0.1	0.6	0.7	1.3	+1	0.3	0.3
t_2	0.5	0.3	0.8	0.2	−0.6	−0.4	0.7	+1	−0.3	−0.3
t_3	0.5	−0.6	−0.1	0.8	1.2	2.0	−0.4	−1	0.6	0.6

Figure 5.7: Operation of a second-order noise-shaping circuit [2].

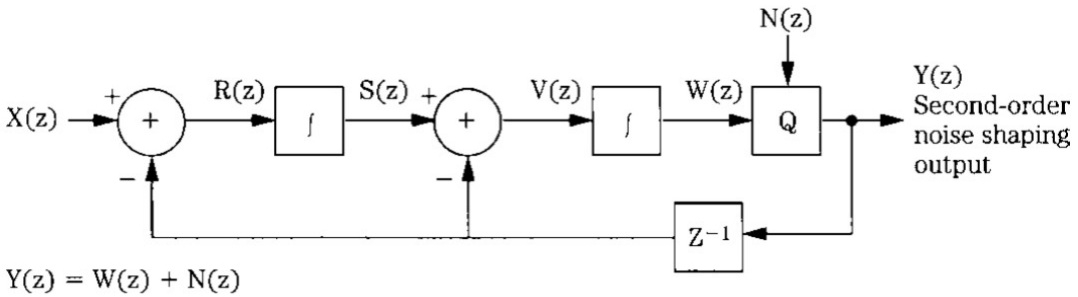
In this one-bit converter, the quantization noise introduced by the word length reduction is spectrally shaped by a low-pass feedback loop around the quantizer (see Figure 5.7). Second-order noise shaping is performed:

$$Y(z) = X(z) + (1 - z^{-1})^2 N(z) \tag{5.4}$$

where:

$Y(z)$ – the noise-shaped output
$X(z)$ – the input signal
$N(z)$ – the quantization noise

The requantization noise of the output signal is corrected by feedback, and noise shaping attenuates its in-band level. As a result, the spectrum of the noise in the output signal is shifted away from the audio band. Figure 5.8 summarizes the mathematical basis of second-order noise shaping.



$$Y(z) = W(z) + N(z)$$

$$W(z) = \frac{1}{1 - z^{-1}} V(z)$$

$$V(z) = S(z) - z^{-1} Y(z)$$

$$S(z) = \frac{1}{1 - z^{-1}} R(z)$$

$$R(z) = X(z) - z^{-1} Y(z)$$

$$(1 - z^{-1})^2 Y(z) = X(z) - z^{-1} Y(z) - z^{-1} Y(z) + z^{-2} Y(z) + N(z)(1 - z^{-1})^2$$

$$Y(z) = X(z) + N(z)\underbrace{(1 - z^{-1})^2}_{\| \\ H_2(z)}$$

$$|H_2(z)| = (|H_1(z)|)^2 = 4\left|\sin^2\left(\frac{\pi f}{f_a}\right)\right|$$

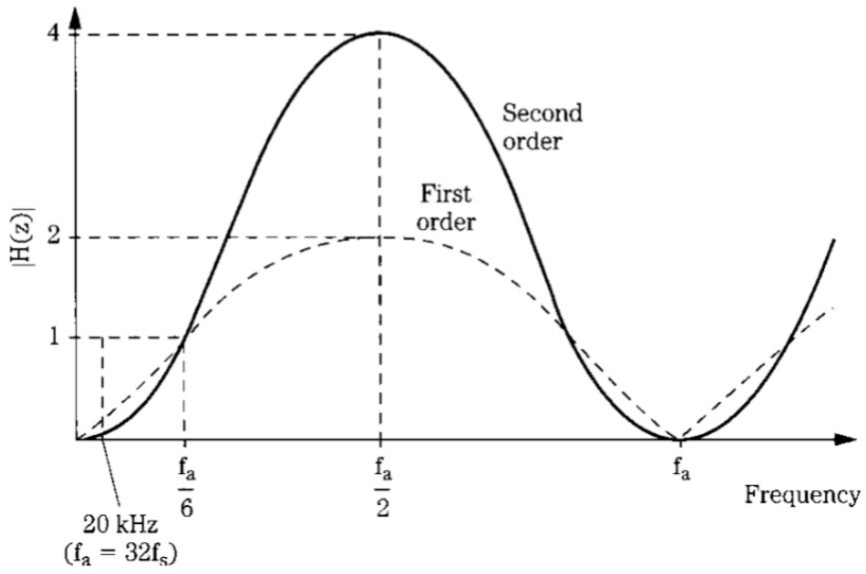


Figure 5.8: Analysis of a second-order noise shaper [2].

* * *

This chapter further discussed the concept of oversampling, the effects of noise shaping, and Sigma-Delta modulation in general.

The next chapter dives into the design and implementation of the IF and the Sigma-Delta modulator, as well as the analog output stage.

Chapter 6

Design and Implementation

This chapter describes in detail how the previously discussed IF and Sigma-Delta modulator are implemented on an FPGA platform using Verilog HDL.

6.1 Design and implementation of interpolation filter

The first task is to determine specifications for the IF and design it accordingly. This Master thesis follows a design proposed in [6] with some key modifications. MATLAB's *filterBuilder* is used, as it offers complete control of the design parameters and conversion to Verilog code.

6.1.1 Specifications

The sample rate and bit depth on the input of the IF is given by the CD-DA format. This gives a sample rate of 44.1kHz, and a bit depth of 16 bits on the input. The OSR of the filter is 128 and the output wordlength, that is also 16 bits, matches the input of the Sigma-Delta modulator.

The specification for the maximum allowed passband ripple is set at 0.001dB. The specifications for the phase response of the IF is set to a linear phase response, in accordance with standard used for audio filters. The finite wordlength arithmetic in the filter will add quantization noise to the signal, and this noise should be audible. The design goal for maximum added distortion in the baseband is set to 1dB [6]. The spectral replicas should be attenuated under the noise floor of the Sigma-Delta modulator, but for the spectral replicas close to the passband where the Sigma-Delta modulators noise floor is very low, a maximum attenuation of 100dB is set. This is in the middle range of the attenuation used by a typical high-end DAC, as discussed in section 4.1. The summary of specifications is listed below:

- OSR: 128,
- Input sample rate: 44.1kHz,
- Input/Ouput wordlength: 16 bits,
- Passband ripple: 0.001dB,
- Distortion: 1dB.

6.1.2 Implementation

The IF is designed using Matlab's DSP toolbox, and Verilog code for the IF is generated through Matlab's code generation.

6.1.2.1 Partitioning

The IF has an interpolation factor of 128. Instead of performing this upsampling in a single stage, it is divided into seven stages, with each stage performing 2x upsampling. The input sample rate, passband, transition-band and stopband for each filter stage is determined, and is summarized in Table 6.1.

Table 6.1: Sample rates and filter bands for the interpolation stages [6].

IF #	**Input sample rate**	**Passband**	**Transition-band**	**Stopband**
1	44.1 kHz	0-20 kHz	20 kHz-24.1 kHz	24.1 kHz-44.1 kHz
2	88.2 kHz	0-22.05 kHz	22.05 kHz-66.1 kHz	66.1 kHz-88.2 kHz
3	176.4 kHz	0-22.05 kHz	22.05 kHz-154.4 kHz	154.4 kHz-176.4 kHz
4	352.8 kHz	0-22.05 kHz	22.05 kHz-330.8 kHz	330.8 kHz-352.8 kHz
5	705.6 kHz	0-22.05 kHz	22.05 kHz-683.6 kHz	683.6 kHz-705.6 kHz
6	1.411 MHz	0-22.05 kHz	22.05 kHz-1.389 MHz	1.389 MHz-1.411 MHz
7	2.822 MHz	0-22.05 kHz	22.05 kHz-2.800 MHz	2.800 MHz-2.822 MHz

The passband, stopband and transition-band of the first filter are set to the typical values for the CD-DA format, and is symmetrical around the middle of the spectrum. The passband, stopband and transition-band for the rest of the filters are also symmetrical around the middle of the spectrum, but have a slightly higher stopband and (passband) to ensure that the whole frequency spectrum of the replicas are suppressed. A symmetrical frequency response allows for the use of half-band filters [6].

The required stopband attenuation for each of the filter stages is in accordance with the specifications of the IF. The complete IF filter must suppress the spectral replicas under the noise floor of the Sigma-Delta modulator. Since the modulator itself has a low-pass filter characteristic, it helps suppress the spectral replicas at high frequencies. This allows for filters in stages 2-6 to have their requirements relaxed [6]. Table 6.2 summarizes the minimum attenuation requirements for each filter stage. The first filter stage is set to 100 dB in accordance with the specifications.

Table 6.2: Stopband attenuation summary [6].

Filter #	1	2	3	4	5	6	7
Attenuation	100 dB	100 dB	78 dB	60 dB	40 dB	23 dB	14 dB

6.1.2.2 Filter classes

The IF uses seven FIR filter stages with varying specifications and types are to be determined accordingly. FIR filters were chosen for their ability to precisely implement linear phase filters. The early stages, with demanding requirements like narrow transition bands

and high stopband attenuation, use half-band filters. Later stages, with relaxed requirements like wide transition bands and low stopband attenuation, use CIC filters, which require no multiplications. However, the CIC filters' passband droop is corrected using an inverse Sinc FIR filter to meet passband ripple specifications. Thus, the filter chain consists of half-band → inverse Sinc → CIC subclasses for optimal efficiency.

6.1.2.3 Matlab design

The *filterBuilder* from Matlab DSP system toolbox is used to generate the three FIR subclasses. All filters have the interpolation factor of $2x$ and linear phase. The final configuration of the IF is summarized in table 6.3.

Table 6.3: Summary of the final IF configuration [6].

Filter Stage	1	2	3	4	5	6	7
FIR Subclass	Half-band	Half-band	Inverse sinc	CIC	CIC	CIC	CIC
Filter Structure	D-F FIR Polyphase	D-F FIR Polyphase	D-F FIR Polyphase	CIC	CIC	CIC	CIC
Polyphase Length	66	12	6	-	-	-	-
Filter Length	131	23	12	-	-	-	-
Differential Delay	-	-	-	1	1	1	1
Number of Sections	-	-	-	3	2	1	1

The fixed-point arithmetic is chosen for the filter, because of the low resource usage in comparison to the floating-point arithmetic. Since CIC filters don't have coefficients, only the first three filter stages needed quantization. The number of bits used for each of the filter stages is summarized in table 6.4.

Table 6.4: Quantized coefficients wordlenght [6].

Filter Stage	1	2	3
Bits	22	20	16

The frequency response of the of the final IF is shown in Figure 6.1. The zoomed in part of the passband can be seen in Figure 6.2.

To avoid the bloating of the wordlength through the cascaded filters, and to scale the final wordlength to the 16 bit input of the Sigma-Delta modulator, quantization must be implemented for each stage. To reduce the quantization noise, intermediate values such as accumulators and products use full precision, leaving only outputs to be quantized. Table 6.5 summarizes the output wordlengths of each stage. The accumulators within the first three filters are utilizing saturation logic to deal with overflows.

Table 6.5: Output wordlength of each filter stage.

Filter Stage:	1	2	3	4	5	6	7
Output wordlength:	21	20	18	19	18	17	16

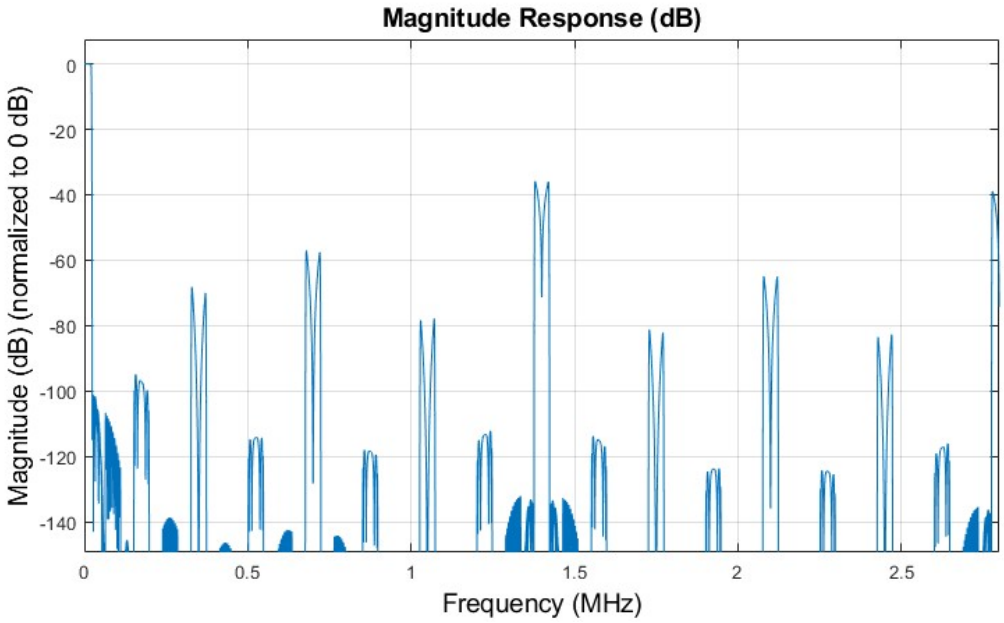


Figure 6.1: Frequency response of the IF.

6.1.3 Design and generating Verilog code

As stated previously, Matlab offers an HDL coder, which can generate complete Verilog code from a filter object. Additionaly, the HDL coder can also generate testbenches for the generated HDL filters. The HDL coder implements the filters in a polyphase structure. The filter stages are effectively running on their lowest possible clock rates. All filters are driven by a global clock, and the operating frequency of each filter is divided from the latest stage backwards. This is done by propagating a clock enable signal through the cascaded filter structure. This implementation is shown in the block diagram of the IF's top module in Figure 6.3. Each of the filter stages in Figure 6.3 is generated with the same interface, which propagates the samples from the lower to higher frequency stage, and clock enable signals in the opposite direction through the IF. This interface is shown in Figure 6.4.

6.1.4 RTL optimization

Although the HDL coder does the job of implementing filters in RTL, it does not provide the optimal solution, nor does it take advantage of all the properties discussed in Section 4.2. The implementation strategies for the filters is by default a direct/fully parallel implementation, as the MAC/serial implementation is not available. The HDL coder's implementation of the last 4 CIC filters was a standard fully parallel CIC implementation with delays and accumulators. This default solution utilizes a large number of resources, making it only possible to implement on a FPGA device that has a large number of LUTs availible. To have a design that with lesser utilization cost, RTL code must be manually optimized.

The optimization strategy is to remove the multiplications with zeros in the half-band filters, take advantage of coefficient sharing, and implement the half-band filters in a MAC implementation.

Every even-indexed half-band filter coefficient is zero except for the middle one, which

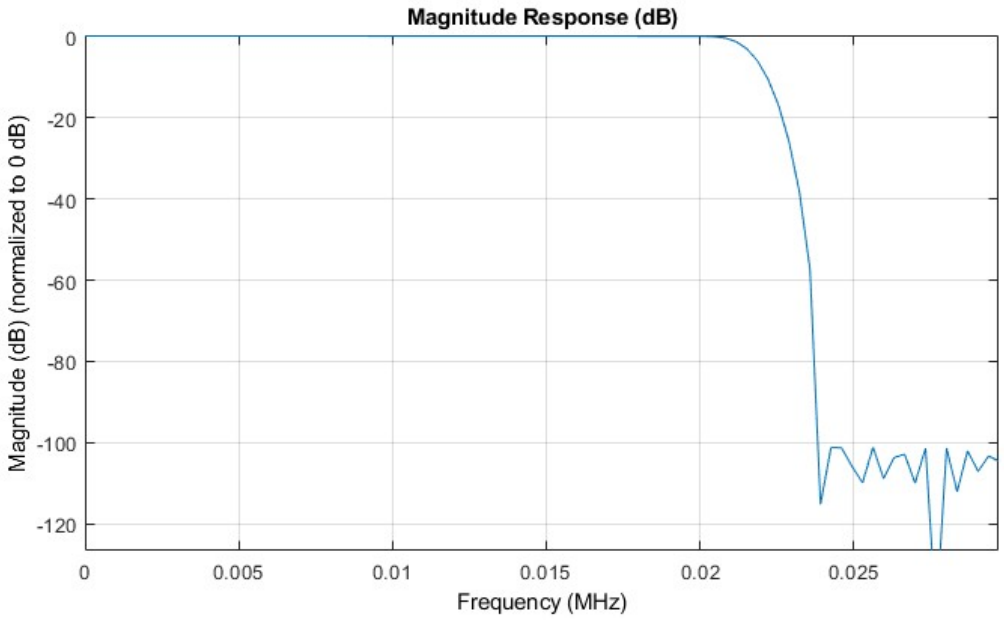


Figure 6.2: Passband and transition-band of the IF.

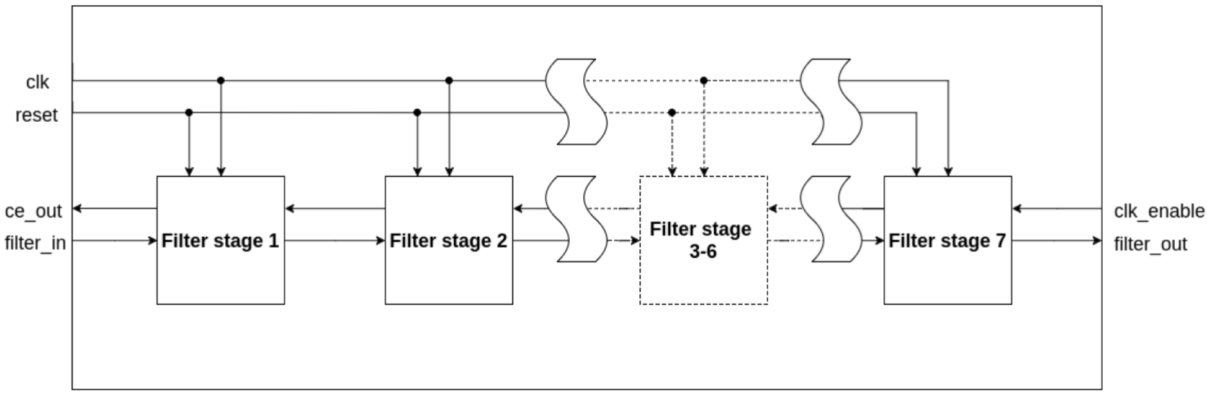


Figure 6.3: Block diagram of IF top module [6].

is equal to one. The half-band filters are implemented in the polyphase structure, where one polyphase sub-filter uses even and the other odd coefficients. This means that in the first sub-filter, only one product is non-zero and requires calculation. This allows for halving a number of multiplications by changing the RTL code so that instead of performing any multiplication in the first sub-filter, the correct delayed sample is set to the output.

The second polyphase sub-filter has symmetric coefficients which allows for usign a coefficient sharing shown in Figure 4.12. The second sub-filter can also be implemented in a MAC structure. The filter in the first stage needs 33 clock cycles to compute the MAC result, and the filter in the second stage needs 6 clock cycles. The filter outputs are clocked 64 and 32 times slower than the global operating clock, respectively. This means that there is enough time to calculate the MAC result for both stages. The RTL is changed so that the second sub-filter is implemented with coefficient sharing in a MAC structure. The complete filter can now be represented as a finite state machine (FSM) solution using only three states (see Figure 6.5).

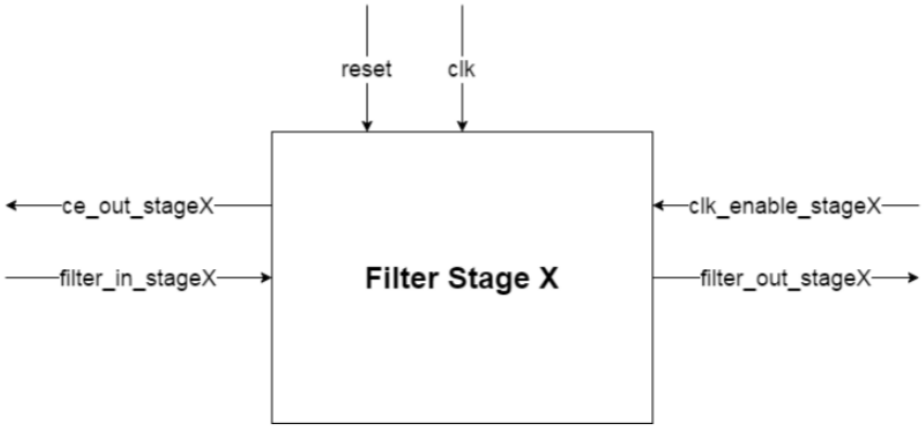


Figure 6.4: Signal interface of the filter stages [6].

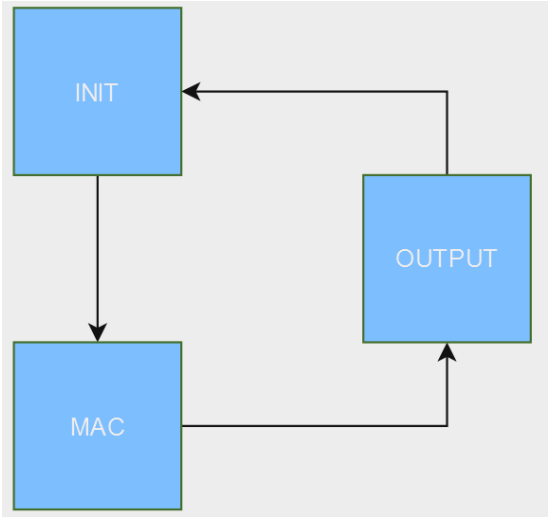


Figure 6.5: FSM implementation of first two stages.

The first state, *INIT*, clears the previous MAC result, resets the internal control counter and takes up one clock period. The second state computes the MAC result for the second sub-filter and takes up 33 periods for the first stage and 6 for the second. The third stage is the last one. It uses up the rest of the clock periods before the next input sample. It also sets the output sample twice: once for the previously calculated MAC result and once for the delayed sample that represents the output of the other sub-filter.

Unlike half-band filters, all the coefficients of the inverse Sinc filter are non-zero. Although the inverse Sinc filter has symmetric coefficients, its two polyphase subfilters do not. This means that implementing coefficient sharing within each sub-filter is not possible. However, the sub-filters can share coefficients with each other, reducing the total number of stored coefficients by half. The sub-filters can also be implemented in a MAC structure. The RTL is now modified the same way as it is for half-band filters, with MAC state now calculating values for two registers.

The generated RTL code of the CIC filters is deemed to be good, and no apparent optimization is found for these filters [6].

6.1.5 Optimization results

The optimized RTL code was synthesized using Yosys with GateMate's CCGM1A1 as the target, in order to evaluate the resource usage of the IF after optimization. As shown in Table 6.6, the optimization resulted in a significant reduction in DSP resources, including adders and multipliers, as well as a noticeable decrease in the overall number of wires. This came at the cost of an increased number of LUTs, reflecting a good trade-off given that there are much less DSP resources available on the device.

Table 6.6: Comparison of the original and optimized IF.

Resource	Original Filter	Optimized Filter	Improvement (%)
Num. of wires	1154	978	15.25
Num. of wire bits	34743	10650	69.35
Num. of cells	10420	4718	54.72
CC_ADDF	3667	1265	**65.5**
CC_BUFG	1	1	0
CC_DFF	5423	2141	**60.52**
CC_IBUF	19	19	0
CC_LUT1	19	132	-594.74
CC_LUT2	110	324	-194.55
CC_LUT3	828	196	76.33
CC_LUT4	144	598	-315.28
CC_MULT	84	3	**96.43**
CC_MX4	-	15	N/A
CC_OBUF	17	17	0

To test and compare the original filter generated by Matlab and the manually optimized one, a testbench was developed. The stimulus is also generated by Matlab and the goal is for the two outputs of the filters to match. Figure 6.6 illustrates that the optimized filter has an identical output as the original one. The only difference now is that the latency is lower, since the MAC implementation uses less pipeline registers.

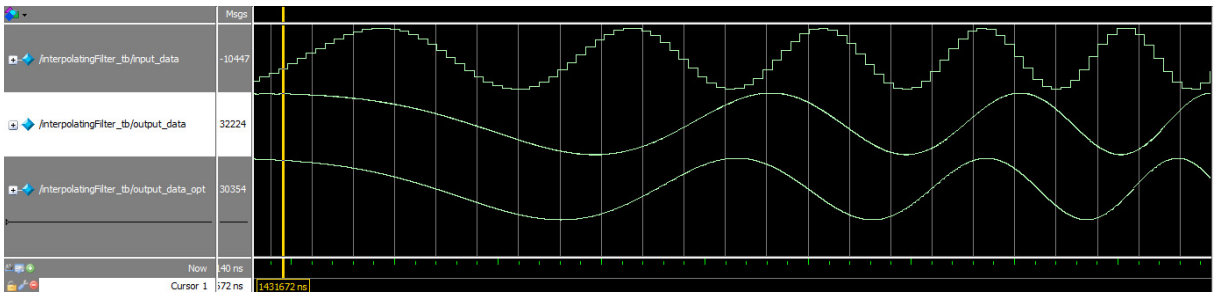


Figure 6.6: Comparison of the original and optimized filter response.

With this, the discussion about implementing an interpolating filter on an FPGA is concluded. The full RTL code can be found in the GitHub repository.

6.2 Design and implementation of Sigma-Delta modulator

The general interface for Sigma-Delta modulator Verilog module is shown in Figure 6.7. It takes 16 bit signed input from the IF and outputs a one-bit PCM signal. In Chapter 5, it was noted that all high-performance DACs incorporate at least a second-order Sigma-Delta modulator, and its implementation is discussed below.

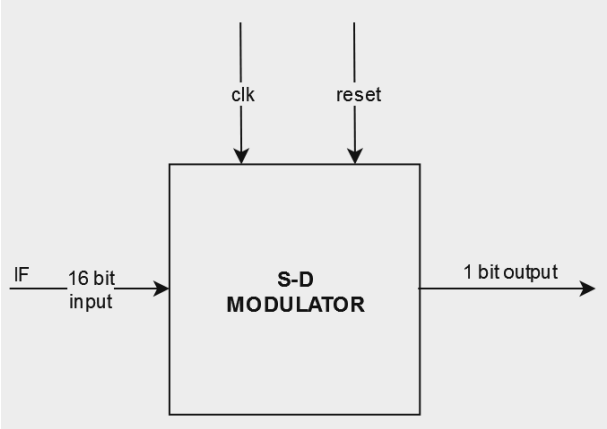


Figure 6.7: Signal interface of the Sigma-Delta modulator.

6.3 Implementation of a Second-Order Sigma-Delta Modulator

From the Figure 5.8 that shows analysis of a second-order Sigma-Delta modulator, a set of differential equations can be derived. In order to avoid overflows, all values are stored in signed 21 bit registers, including the input that is sign-extended by 5 bits at the beginning. The modulation can be summarized by the following algorithm:

Algorithm 1 Second-Order Sigma-Delta Modulator

1: **Input:** *sd* (signed 16-bit),
2: **Output:** *out* (1-bit)
3: **Parameters:** FULLSC $\leftarrow$ 150% of full input scale
4: Compute the first residual: $r_k \leftarrow in_k - y_{k-1}$
5: First accumulator: $s_k \leftarrow r_k + s_{k-1}$
6: Compute the second residual: $v_k \leftarrow s_k - y_{k-1}$
7: Second accumulator: $w_k \leftarrow v_k + w_{k-1}$
8: Quantize: $y_k \leftarrow MSB(w_k) \quad ? - FULLSC : FULLSC$
9: Set output bit: $out \leftarrow\sim MSB(w_k)$

The sequential logic is used to set the output register and to update the values to be passed to the next iteration. The combinatorial logic computes the values of all the registers. The feedback signal is quantized to ±150% of a full input scale to account for values growing throughout the registers [9].

6.4 DAC Top Module

After the design of the IF and Sigma-Delta modulator is finished, they are connected in a top module. Both modules are driven by a global clock at the 128 times the input sample frequency of 44.1kHz (see Figure 6.8).

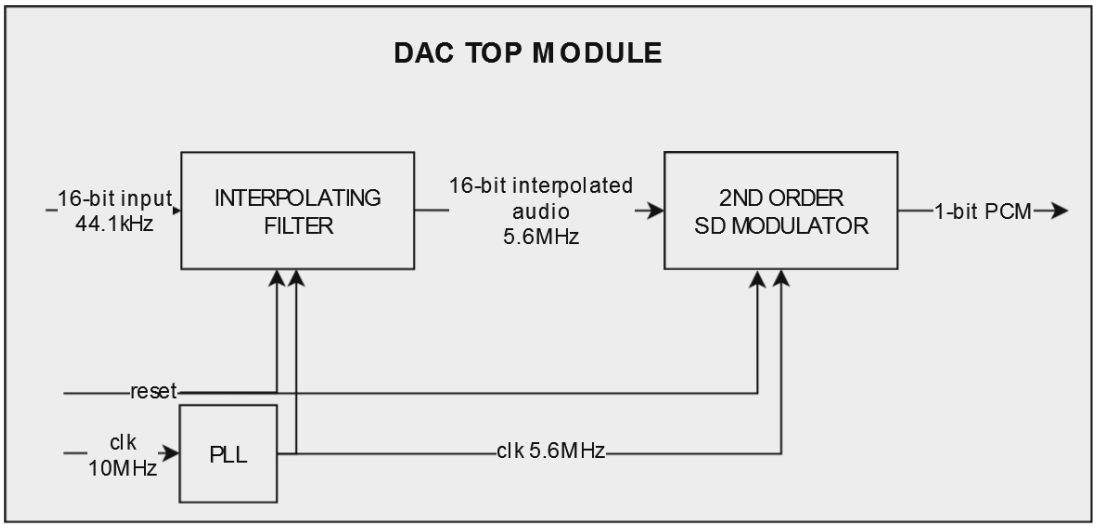


Figure 6.8: DAC Top Module.

The phase-locked loop (PLL) is used to generate a global operating clock frequency of 5.6MHz. It is at this rate, that the modulator outputs the PCM signal. The clock frequency is divided backwards through the IF module so that the input sample rate is 44.1kHz.

From the post-implementation utilization report, showed in Table 6.7, it can be seen that the complete design uses only around 17% of the available logic elements for a 20K LUT device. This leaves plenty of room for the main application to be implemented.

Table 6.7: Utilization Report.

Resource	Used	Available	Percentage
CPEs	3516	20480	17.2%
CPE Registers	2248	40960	5.5%
Flip-flops	2248	—	—
Latches	0	—	—

6.5 Design of Class-D Amplifier

Once the PCM signal that carries the audio is present at the output of FPGA pins, it requires analog circuitry to reproduce the sound. The solution presented here uses a digital Class D amplifier.

A class-D audio amplifier works by using pulse width modulation (PWM) to switch the power transistors on and off at a high frequency. This high-frequency switching creates an average output that is proportional to the input signal. The input signal is

first converted into a PWM signal by a class-D amplifier driver IC or using high speed comparator op-amp. This PWM signal is then used to control the power transistors, which are typically N-channel MOSFETs, to switch on and off at a high frequency. The output of the transistors is filtered by an LC filter to remove the high-frequency switching noise, leaving only the amplified audio signal.

To understand it better the functional block diagram of class D amplifier is presented in Figure 6.9.

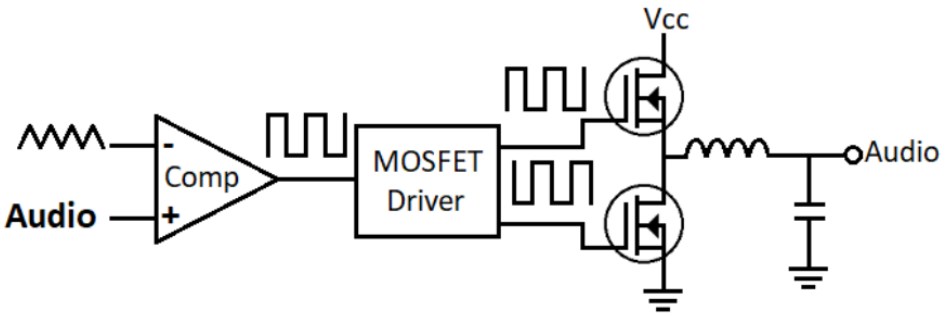


Figure 6.9: Functional block diagram of class D amplifier [20].

Firstly, the PWM equivalent of the audio signal is generated by comparing it to a triangular signal using a comparator (see Figure 6.10).

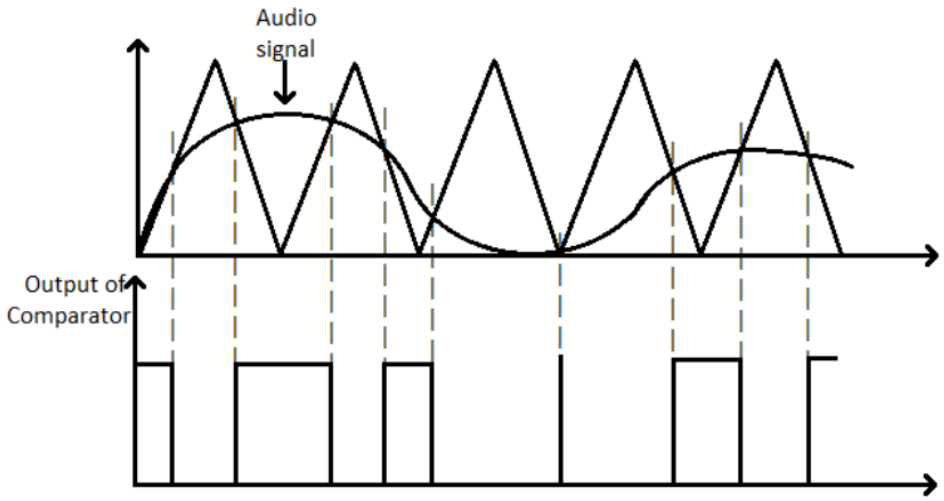


Figure 6.10: Audio to PWM signal [20].

To recreate an audio signal accurately, the frequency of the triangular signal should be at least 10 times higher than that of the audio signal. In a class D amplifier, the triangular signal is connected to the inverting terminal, while the audio signal is connected to the non-inverting terminal of a comparator. When the audio signal's voltage is lower than the triangular signal's voltage, the comparator output is low. On the other hand, when the audio signal's voltage is higher than the triangular signal's voltage, the comparator output is high. This process generates a high-frequency PWM signal that corresponds to the audio signal. The PWM signal is then inverted using an inverter. Two MOSFETs, acting as switches, are driven by MOSFET drivers. Since MOSFETs operate in their

linear (ohmic) region, the voltage drop across their drain and source is minimal, resulting in very low power loss, making this amplifier a highly efficient one. After amplification, the high-frequency components of the PWM signal need to be removed. This is achieved using an LC low-pass filter, which is preferred over an RC filter to minimize real power loss [20].

For the solution this Master thesis presents, the PWM/PCM signal is already generated on FPGA via Sigma-Delta modulation, so the op-amp part is omitted. This also allows for simply using two FPGA pins in place of a MOSFET driver, leaving only MOSFETs and LC low-pass filter to connect to the FPGA. The values for filter are $L = 30\mu H$ and $C = 2.2\mu F$ which give a cut-off frequnecy of 19.6kHz. The full circuit diagram of the proposed solution is illustrated in Figure 6.11.

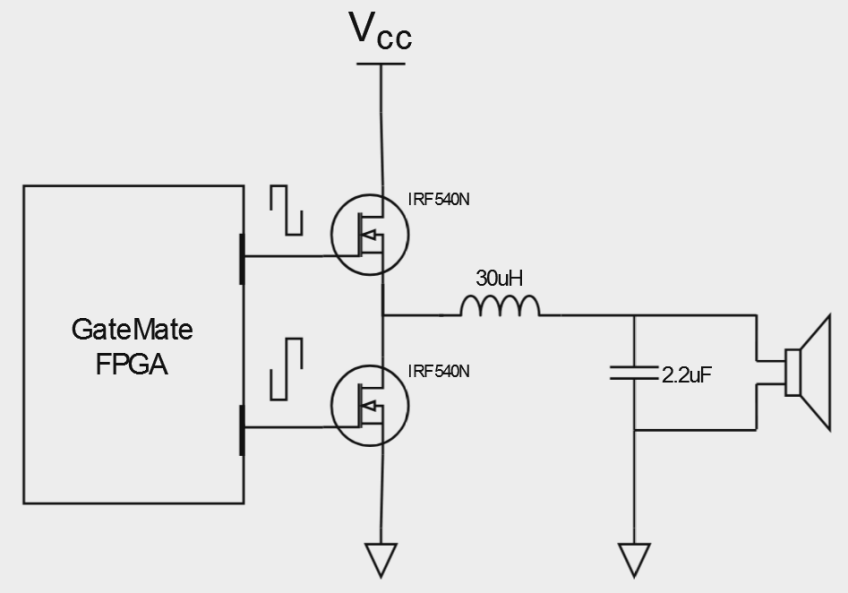


Figure 6.11: Circuit diagram of Class-D amplifier based solution.

* * *

This chapter provided a detailed explanation of the implementation of the IF and Sigma-Delta modulator in HDL, emphasizing the impact of optimization on the IF design. Additionally, the design of the output stage, utilizing a digital Class-D amplifier filter, was presented.

The next chapter will focus on testing and evaluating the proposed design.

Chapter 7

Experimental results

The following chapter presents the results and evaluation metrics of the implemented solution. Two experiments were conducted: one for the evaluation of the DAC and another for the Class-D amplifier. A LUT-based sinewave generator [21] was implemented within the FPGA. However, the details of this implementation are beyond the scope of this thesis and will not be elaborated upon here.

7.1 DAC testing and evaluation

To test and evaluate the Sigma-Delta DAC itself, a PCM at the output of a digital pin is low-passed by a simple RC filter ($R = 33k\Omega$, $C = 0.22nF$, $f_c = 18.55kHz$). PCM code carries a sinewave signal of 440Hz, which is displayed on an oscilloscope and saved for analysis in Matlab.

Figure 7.1 shows the 440Hz sinewave displayed on an oscilloscope. The generated input frequency is correctly reconstructed, however, with noticeable noise.

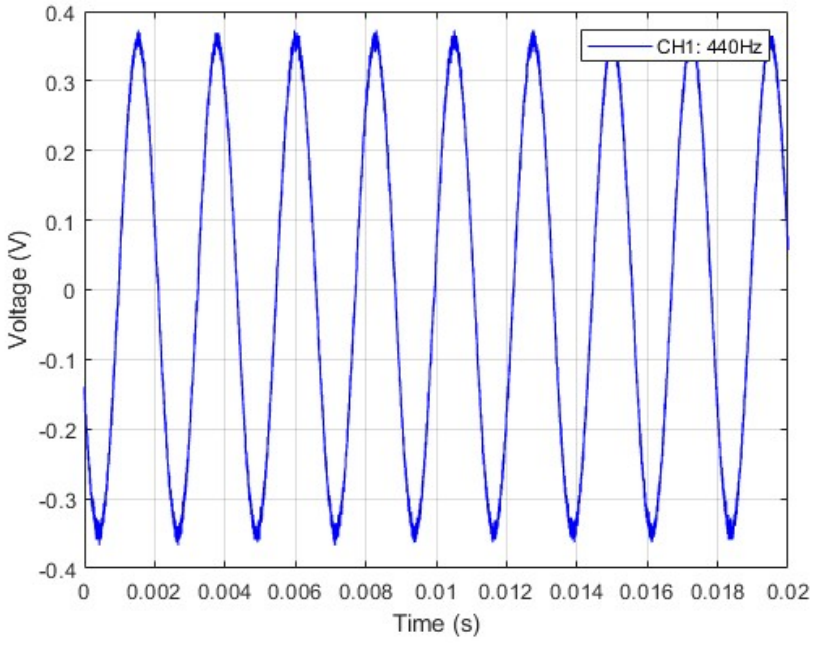


Figure 7.1: DAC output through RC: 440Hz sinewave.

Figure 7.2 shows the amplitude spectrum of the sinewave.

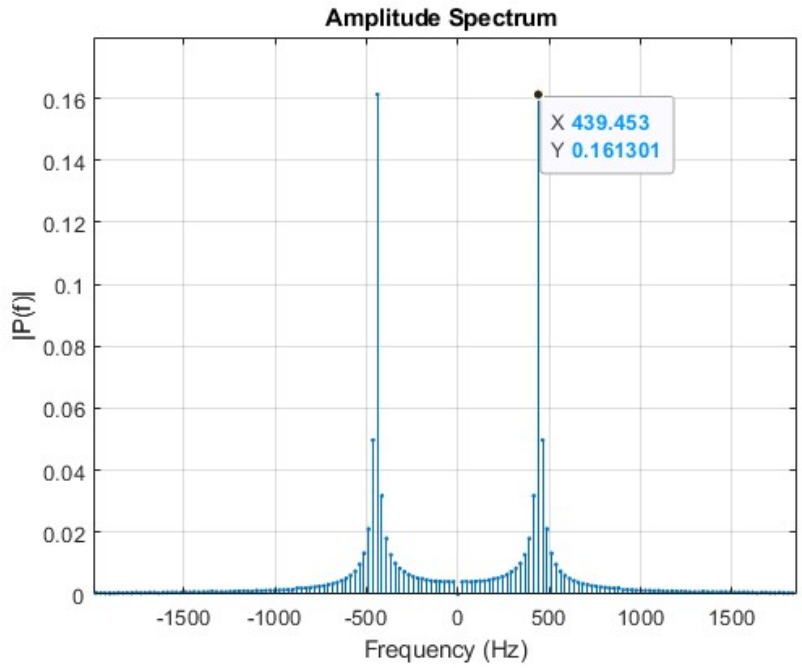


Figure 7.2: DAC output through RC: 440Hz sinewave amplitude spectrum.

To calculate the SNR and THD performance, built-in Matlab functions were used. Figures 7.3 and 7.4 illustrate the calculated values.

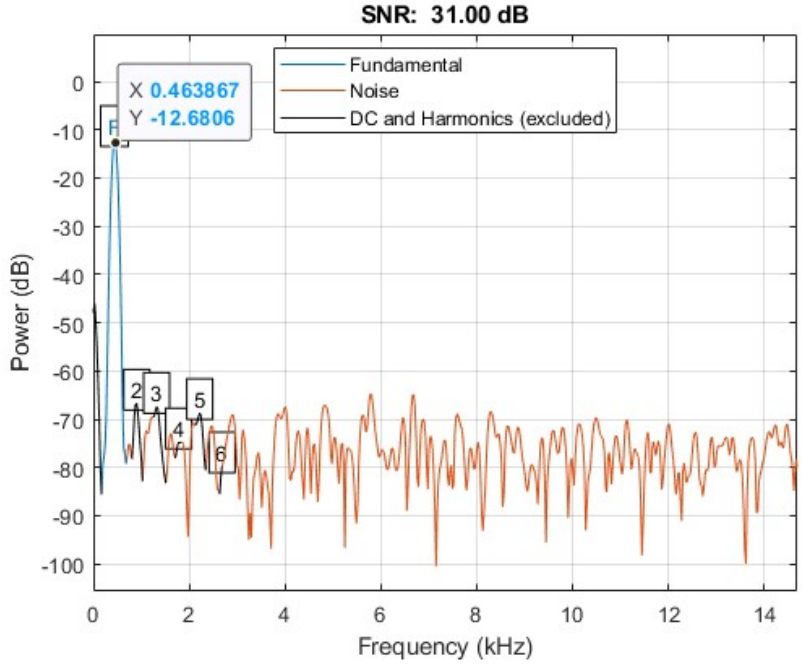


Figure 7.3: 440Hz sinewave SNR.

For basic sine wave reconstruction, an SNR of 31 dB is acceptable. However, for high-fidelity audio a higher SNR (e.g., >60 dB) would be desirable.

THD of -49 dB indicates that the harmonic components are approximately 0.316% of the fundamental signal. While a THD of 0.316% is sufficient for many mid-range

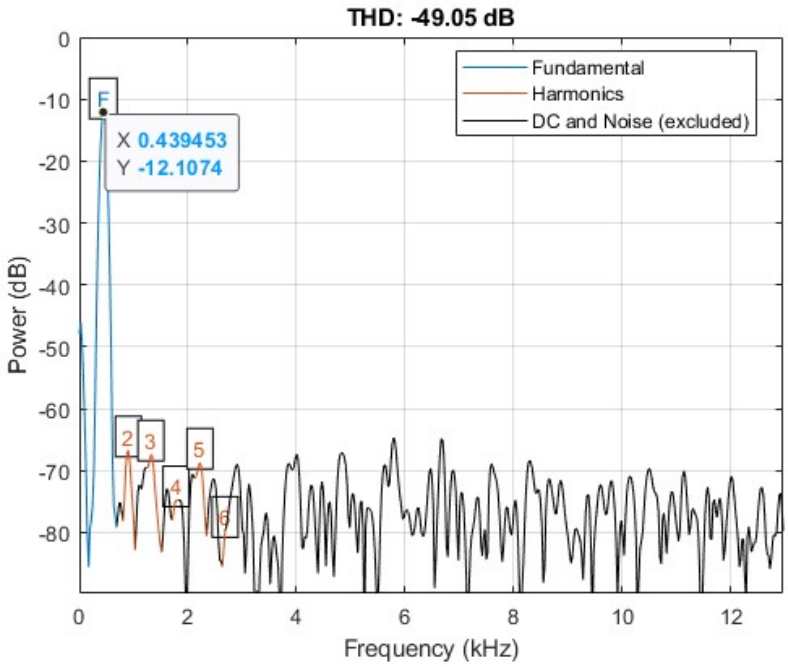


Figure 7.4: 440Hz sinewave THD.

audio and signal reconstruction applications, it may not meet the requirements for high-fidelity audio. Reducing THD further typically requires higher-order filters and improved reconstruction methods. A simple RC filter does not accomplish target values because it has limited roll-off and cannot effectively suppress high-frequency quantization noise introduced by the Sigma-Delta modulator. As a result, residual noise and harmonics degrade the SNR and increase total harmonic distortion.

7.2 Class-D amplifier testing and evaluation

The schematic shown in Figure 6.11 is realized using discrete components and breadboard (see Figure 7.5). The pair of inverted PCM signals now carries the modulated sinewave of 2kHz.

The resulting reconstructed sinewave is shown in Figure 7.6 and the amplitude spectrum is shown in Figure 7.7. From these two figures it can be seen that the reconstruction of the desired frequency is done correctly, but this time with even more noise. Class-D amplification allows for efficient reproduction of the sound along with the control of its volume. However, in this form, it does not provide enough high-frequency filtering.

As expected, the values of SNR and THD don't meet the target values. From the Figures 7.8 and 7.9 it can be seen that the SNR and THD are now twice as bad as in the first setting.

To view the effect of high-frequency noise, the full power spectrum is shown in Figure 7.10. It can be seen that the noise level increases slightly with higher frequencies.

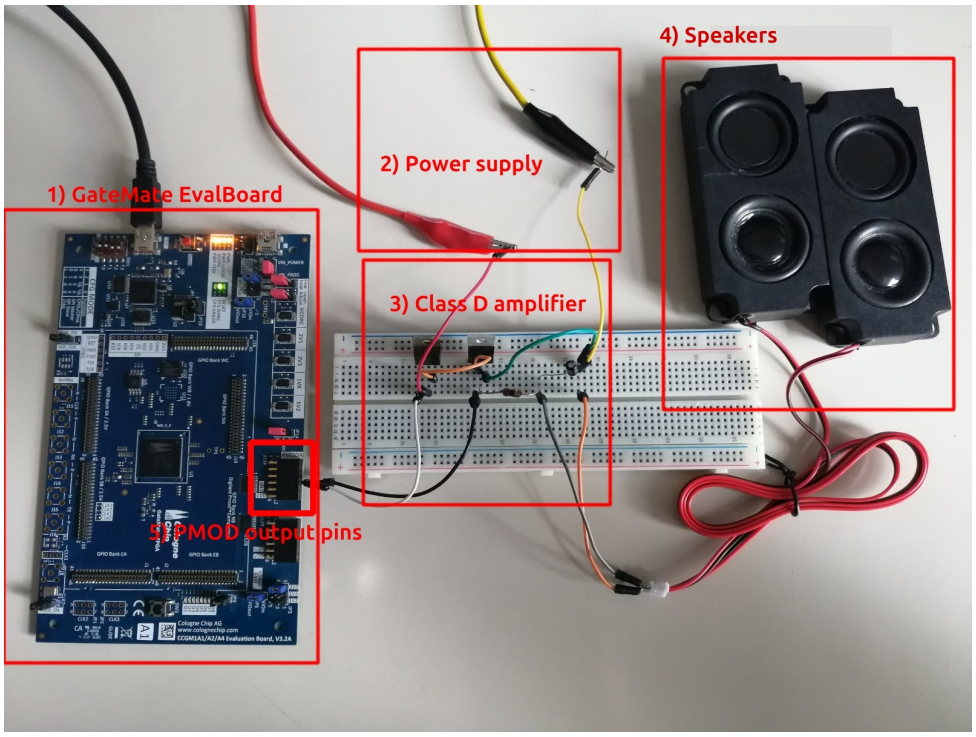


Figure 7.5: Lab bench setup: 1) GateMate's Evaluation Board, 2) Power supply, 3) Class-D amplifier circuit, 4) Speakers, and 5) PMOD output pins.

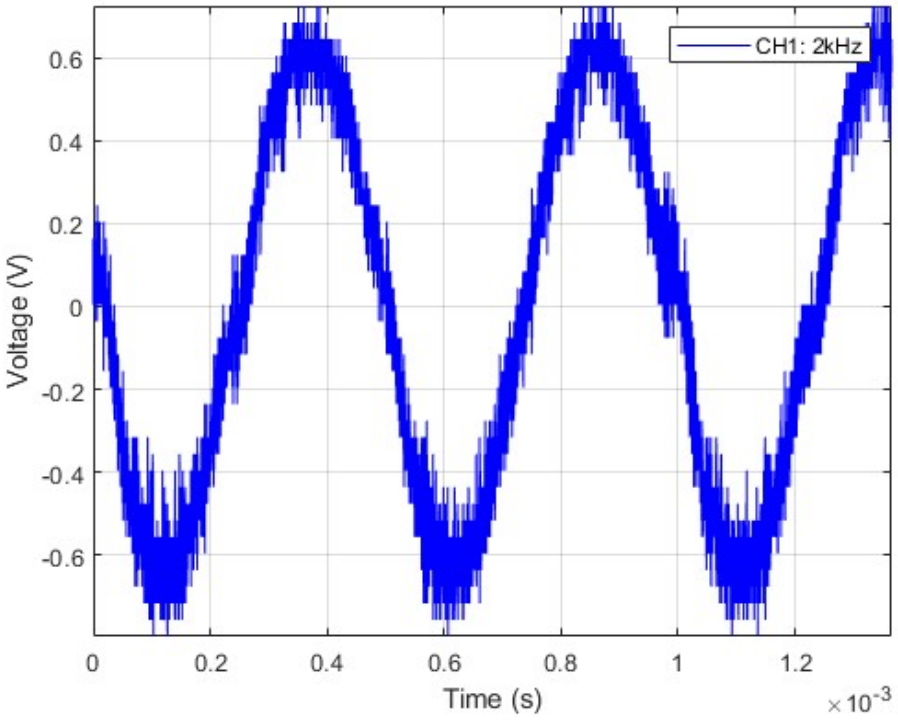


Figure 7.6: Class-D output: 2kHz sinewave.

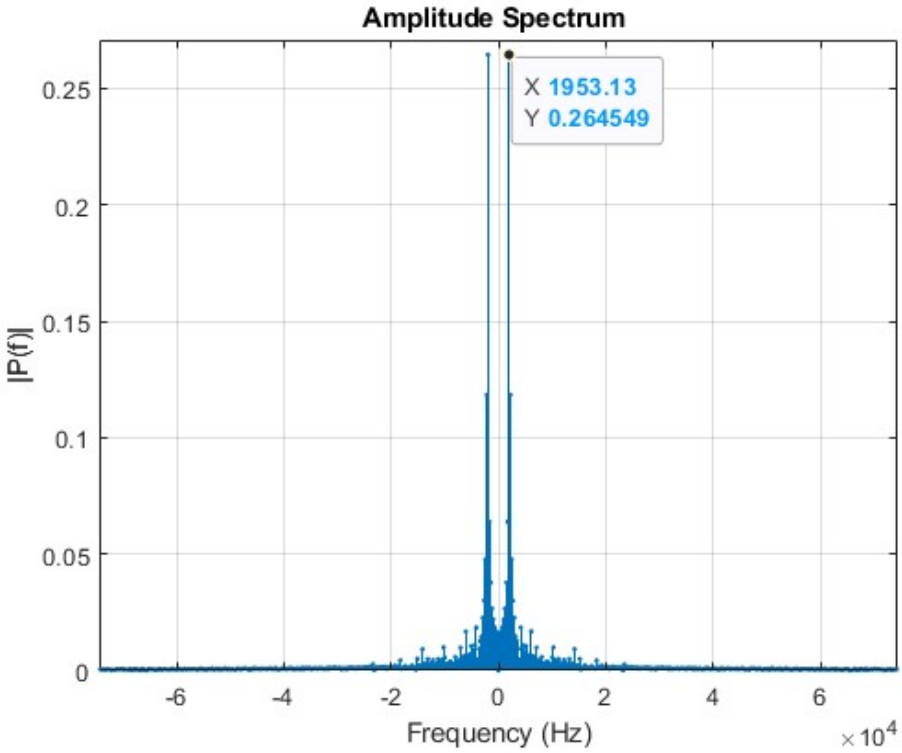


Figure 7.7: Class-D output: 2kHz sinewave amplitude spectrum.

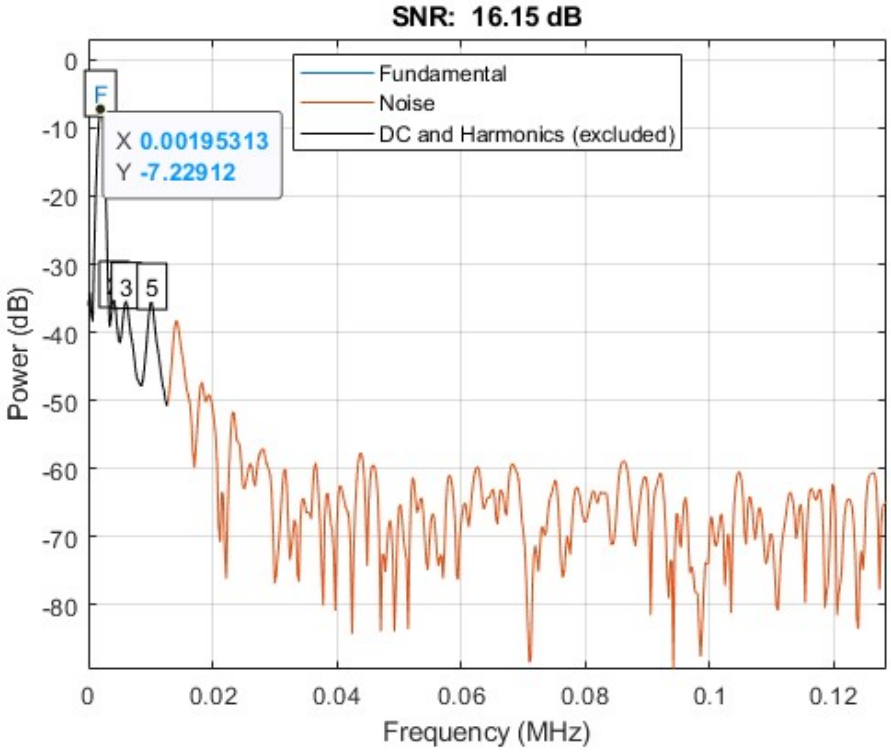


Figure 7.8: Class-D output: 2kHz sinewave SNR.

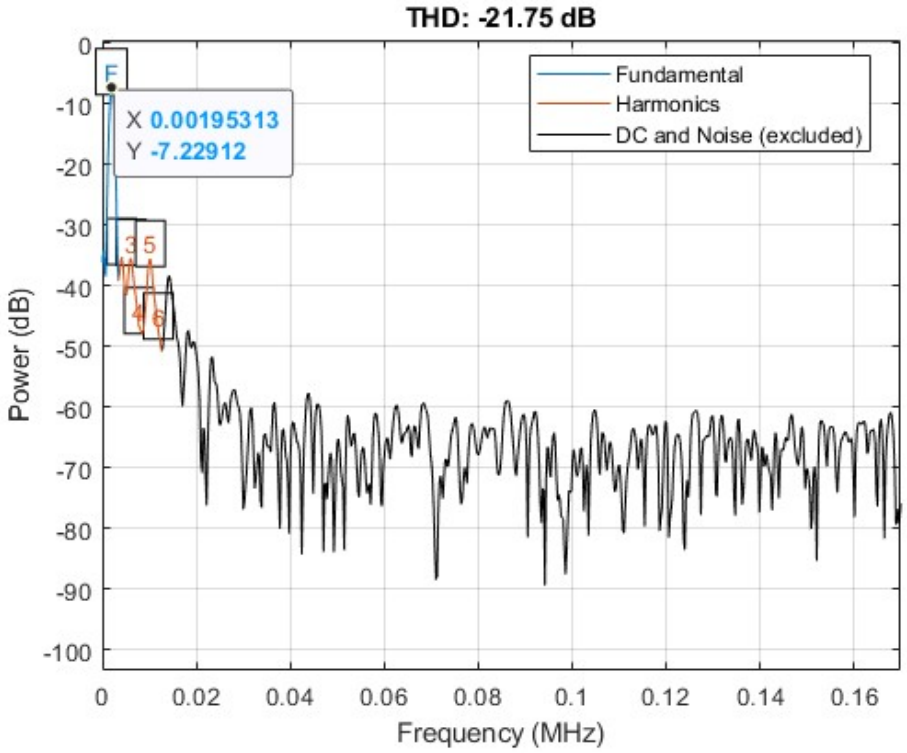


Figure 7.9: Class-D output: 2kHz sinewave THD.

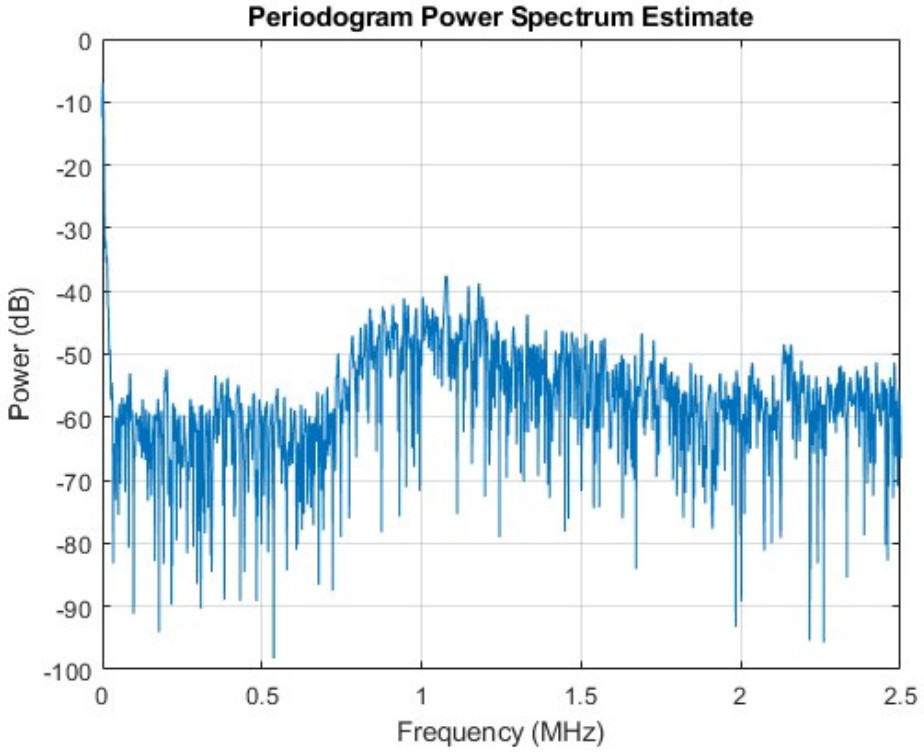


Figure 7.10: Class-D output: power spectrum.

7.3 Summary

Overall, the resulting signals reproduced by the developed Sigma-Delta DAC did not meet the high-quality performance expectations. This is mainly due to the inadequate analog circuitry needed to reconstruct PCM signals to analog form. The analog filter needs to remove all out-of-band portions of the output signal of the internal DAC, and must not introduce appreciable nonlinear distortion into the signal while doing so. This task is particularly difficult for single-bit DACs where a large two-level analog signal enters the post- filter [1]. A simple RC circuit or single-stage filtering is not sufficient to meet these requirements. This highlights the need for significant improvements in this aspect of the design presented in the thesis.

The block diagram of a proposed improvement, that is a typical post-filter for a single-bit DAC, is shown in Figure 7.11.

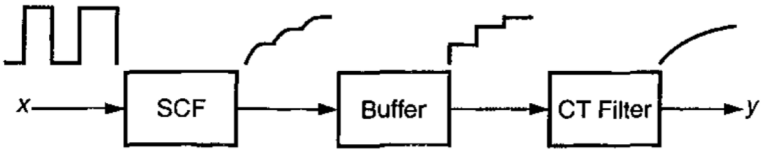


Figure 7.11: Post-filter for a 1-bit ΣΔ DAC and associated signals [1].

It is customary to use switched-capacitor filter (SCF) stages as the input stages of the post-filter. A SCF with sampled-data input and output needs only the samples $x(nT)$ of $x(t)$ as its input signal, and it can remove most of the high-frequency power from the signal (and thus reduce its step size). Once the step size of the waveform is small enough, such that the slew rate required for its linear continuous-time (CT) processing is acceptably low, it can then be filtered by a CT active filter. Since the signal/noise ratio of the switched-capacitor (SC) filter in Sigma-Delta DAC must often be extremely high, its susceptibility to internal noise sources is an important design factor. This may result in the choice of unconventional architectures for the SCF [1].

After sufficient filtering, the step size of the SCF waveforms can be greatly reduced. However, the waveforms will still exhibit nonlinear distortion. Hence, it is necessary to use a buffer stage which is driven by the samples $y(nT)$ of the SCF output, and which provides a waveform free of such transients. This can be achieved by using a direct-charge-transfer (DCT) stage [1].

The output of the buffer stage can now be fed to the CT filter. This filter needs to eliminate the remaining noise above f_B. Typically, it is a second- or third-order active-RC circuit, often using the Sallen-Key configuration (see Figure 7.12) [1].

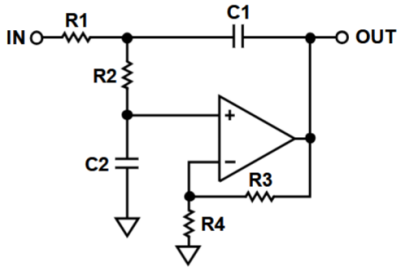


Figure 7.12: Sallen-Key Low-Pass Filter [22].

Chapter 8

Conclusion

The research presented in this Master thesis has successfully demonstrated the design and implementation of a Sigma-Delta Digital-to-Analog Converter (DAC) integrated with a Class-D amplifier using FPGA technology. The thesis focused on key components of the Sigma-Delta DAC, including an optimized interpolation filter and a second-order Sigma-Delta modulator.

Matlab's DSP Toolbox facilitated the design of FIR filters and their conversion into Verilog modules. However, the generated code required significant manual optimization to enhance its usability. These optimizations were implemented using a FSM based solution. Additionally, the Sigma-Delta modulator was modeled as a set of differential equations and integrated with the IF in the top-level Sigma-Delta DAC module.

As part of the analog design, an RC low-pass filter and a Class-D amplifier were employed. The system was tested by generating sinewaves on an FPGA, passing them through the DAC, and then passing the 1-bit PCM output into the analog filter. While the desired frequency was accurately reconstructed in both tests, the output signals exhibited a substantial amount of high-frequency noise, and the proposed analog front-end did not meet the target figures of merit — including SNR, THD, and ENOB.

To achieve better results, several design aspects could be refined. For instance, the Sigma-Delta modulator could be upgraded to a higher-order implementation or expanded into multiple stages (e.g., MASH configuration). The analog stage could likewise be improved by introducing multiple phases and implementing a robust PCB solution to minimize noise from external connections. Finally, audio testing could be enhanced by reproducing audio samples directly from FPGA memory or by using various audio CODECs.

Bibliography

[1] R. Schreier and G. Temes, *Understanding Delta-Sigma Data Converters*. Wiley, 2004, ISBN: 9780471465850. [Online]. Available: https://books.google.ba/books?id=5HNqQgAACAAJ.

[2] K. C. Pohlmann, *Principles of digital audio*. McGraw-Hill Professional, 2000.

[3] S. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub., 1997, ISBN: 9780966017632. [Online]. Available: https://books.google.ba/books?id=rp2VQgAACAAJ.

[4] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, *FPGA-based Implementation of Signal Processing Systems*. Wiley, 2017, ISBN: 9781119077954. [Online]. Available: https://books.google.ba/books?id=77D4DQAAQBAJ.

[5] D. Self, *Audio Power Amplifier Design Handbook*. Taylor & Francis, 2006, ISBN: 9781136123733. [Online]. Available: https://books.google.ba/books?id=-GN-LbTTj7gC.

[6] H. T. Jøsok, "Realization of sigma-delta DAC for audio application on FPGA," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:67349073.

[7] *AN-283 Application note: Sigma-Delta ADCs and DACs*. ANALOG DEVICES.

[8] I. Løkken, "Interpolation filters for oversampled audio DACs," 2006. [Online]. Available: https://pdfcoffee.com/interpolation-filters-for-oversampled-audio-dacs-pdf-free.html#Ivar+L%C3%B8kken.

[9] J. A. Álvarez, "ΣΔ ADCs and DACs for FPGAs." [Online]. Available: http://www.ele.uva.es/~jesus/AD_sigma_delta.pdf.

[10] C. Schmidt, *Interleaving Concepts for Digital-to-Analog Converters*. Springer Vieweg Wiesbaden, 2019, ISBN: 9783658272630. [Online]. Available: https://link.springer.com/book/10.1007/978-3-658-27264-7.

[11] R. G. Lyons, *Understanding digital signal processing, 3/E*. Pearson Education India, 1997.

[12] D. G. Manolakis and V. K. Ingle, *Applied digital signal processing: theory and practice*. Cambridge university press, 2011.

[13] MathWorks. "Filter builder design process." Accessed: 2024-11-30. (2024), [Online]. Available: https://www.mathworks.com/help/signal/ug/filterbuilder-design-process.html.

[14] C. C. AG, *Gatemate fpga product brief*, Accessed: 2025-01-06, Nov. 2023. [Online]. Available: https://www.colognechip.com.

[15] Cologne Chip AG. "Gatemate™ FPGA features." Accessed: 2024-11-30. (2024), [Online]. Available: https://colognechip.com/programmable-logic/gatemate/.

[16] Cologne Chip AG. "Gatemate™ FPGA evaluation board." Accessed: 2024-11-30. (2024), [Online]. Available: https://colognechip.com/programmable-logic/gatemate-evaluation-board/.

[17] Cologne Chip AG, *Gatemate™ FPGA toolchain installation user guide*, UG1002 User Guide, August 2024, Cologne Chip AG, Eintrachtstr. 113, 50668 Köln, Germany, Aug. 2024. [Online]. Available: https://colognechip.com/docs/ug1002-toolchain-install-latest.pdf.

[18] F. Maloberti, *Data Converters*. Springer, 2007, ISBN: 9780387324852. [Online]. Available: https://books.google.ba/books?id=Kvo7cjmaEpkC.

[19] W. Kester, *Oversampling interpolating DACs*, Analog Devices Tutorial MT-017, Accessed: 2024-11-24, 2009. [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-017.pdf.

[20] E. Fun. "Class d amplifier." Accessed: 2024-12-09. (2024), [Online]. Available: https://electronics-fun.com/class-d-amplifier/.

[21] D. Gisselquist. "The quarter wave sine wave table." Accessed: 2025-01-07. (2017), [Online]. Available: https://zipcpu.com/dsp/2017/08/26/quarterwave.html.

[22] H. Zumbahlen, *Mini Tutorial MT-222: Sallen-key filters*, Rev. B, Accessed: 2025-01-08, Analog Devices, Inc., 2017. [Online]. Available: http://www.analog.com/MT-222?doc=MT-222.pdf.

Index of Terms

ADC Analog-to-Digital Converter

BGA Ball Grid Array

CIC Cascaded Integrator-Comb Filter

CPE Cologne Programmable Element

DAC Digital-to-Analog Converter

DM Delta Modulation

DNL Differential Nonlinearity

DSP Digital Signal Processing

ENOB Effective Number of Bits

FoM Figure of Merit

FPGA Field-Programmable Gate Array

FSM Finite State Machine

GPIO General Purpose Input/Output

HDL Hardware Description Language

IMD Intermodulation Distortion

INL Integral Nonlinearity

JTAG Joint Test Action Group

LPF Low-Pass Filter

LUT Look-Up Table

LVDS Low Voltage Differential Signaling

MAC Multiplier-Accumulator

OSR Oversampling Ratio

PCB Printed Circuit Board

PCM Pulse-Code Modulation

PWM Pulse Width Modulation

RC Filter Resistor-Capacitor Filter

ROM Read-only memory

SDM Sigma-Delta Modulation

SerDes Serializer/Deserializer

SFDR Spurious-Free Dynamic Range

SINAD Signal-to-Noise-and-Distortion Ratio

SMA SubMiniature Version A (connector)

SNR Signal-to-Noise Ratio

SPI Serial Peripheral Interface

THD Total Harmonic Distortion

USB Universal Serial Bus