# Power Optimization Methodologies for Digital FIR Decimation Filters

Dissertation zur Erlangung des Doktorgrades
der Technischen Fakultät
der Albert-Ludwigs-Universität Freiburg

Ahmed Shahein

2014

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

Doktor-Ingenieur
in
Microsystems Engineering

according to the examination regulations at the University of Freiburg for the Ph.D. in Microsystems Engineering of 10.05.2000.

Department of Microsystems Engineering – IMTEK
Fritz Huettinger Chair of Microelectronics
University of Freiburg
Freiburg im Breisgau, Germany

**Declaration** according to §5(2f) of the examination regulations

I hereby confirm to have written the following dissertation on my own, not having used any other sources or resources than those listed.

Freiburg, July 10, 2014

Ahmed Shahein

**To my beloved parents - my father and best friend Prof. Dr.-Eng. Hussein Shahein and my mother Eng. Magda Farouk. Who gave me all the support and motivation I need.**

# Contents

# Abstract

There is a trend to replace analog by digital components, resulting in digital filters to become ubiquitous in signal processing applications. Especially for digital finite impulse response (FIR) filters, the power consumption though can be considerable and become a limiting factor in numerous scenarios, e.g., mobile and automotive applications. One major focus of this research work therefore is the optimization of power consumption in digital FIR filters, both on system and register transfer level (RTL).

The practical design tools implemented within the framework of this thesis include a consolidated toolbox for design, optimization and RTL implementation of digital FIR decimation filters, and a set of RTL-based soft IPs for several digital FIR filter topologies. The power optimization in this tool-chain is based on an optimization of filter coefficients, a beneficial combination of filter topologies and an improved multiplier architecture.

The complexity and therewith power consumption of hardwired multipliers in FIR filters is directly related to the number of ones in the binary representation of the filter coefficients. The optimization of filter coefficients is achieved by reducing the number of ones. A heuristic algorithm to minimize this coefficient cost is presented. The proposed algorithm achieves a remarkable reduction in the computation time compared to the state-of-the-art algorithms due to an extensive preprocessing analysis on the filter coefficients and a novel allocation scheme. A reduction in the run-time by a factor of 400 is achieved by the presented algorithm compared to the state-of-the-art algorithms.

The improved filter topologies are accomplished through a power aware combination criterion for different FIR digital filter topologies. A reduction in power dissipation up to 15% is achieved by the proposed criterion compared to the conventional filter architecture.

An improved multiplier architecture is proposed implementing a novel encoding scheme for shift-and-add multiplier-less architectures. A reduction in hardware cost by 25% is achieved compared to the state-of-the-art scheme.

The second main focus of this work is the power optimization of digital receivers, which incorporate complex FIR filters as discussed above, but furthermore require numerically controlled oscillators (NCOs) and a careful topological design. A comprehensive study for ROM-based NCOs is carried out. The study employs various state-of-the-art methodologies for performance enhancement such as symmetry, dithering and linear approximation. The result is a design scheme for calculating the implementation parameters for a low complexity ROM-based NCO architecture.

Finally, a power optimized digital front end for a tunable narrow band FM digital receiver is designed and synthesized in a 130 nm CMOS technology. A power reduction by 60% is achieved by the proposed design compared to the conventional architecture.

# Zusammenfassung

Es ist der Trend zu beobachten, dass analoge Komponenten zunehmend durch digitale ersetzt werden. Digitale Filter für die Signalverarbeitung in unterschiedlichen Anwendungsfeldern sind dadurch allgegenwärtig geworden. Insbesondere die Leistungsaufnahme von digitalen FIR (finite impulse response)-Filtern kann allerdings erheblich sein, was die Anwendbarkeit in vielen Fällen einschränken kann, z.B. in mobilen Anwendungen und in der Automobiltechnik. Ein Schwerpunkt dieser Forschungsarbeit ist daher die Optimierung der Leistungsaufnahme in digitalen FIR-Filtern, sowohl auf der System-, wie auch auf der Register-Transfer-Ebene (register transfer level, RTL).

Die hierfür realisierten Entwurfswerkzeuge beinhalten einen umfassenden Satz an Werkzeugen für das Design, die Optimierung und RTL-Implementierung von digitalen FIR Dezimationsfiltern, und einen Satz von RTL-basierten soft-IPs für unterschiedliche, digitale FIR-Filter-Topologien. Die Optimierung der Leistungsaufnahme in diesen Werkzeugen erfolgt mittels einer Optimierung der Filterkoeffizienten, einer vorteilhaften Kombination von Filtertopologien und einer verbesserten Architektur für die verwendeten Multiplizierer.

Die Komplexität und damit auch Leistungsaufnahme der festverdrahteten Multiplizierer in FIR Filtern steht in direktem Bezug zu der Anzahl an Einsen in der Binärdarstellung der Filterkoeffizienten. Die Optimierung der Filterkoeffizienten erfolgt durch eine Reduzierung der Anzahl an Einsen. Es wird ein heuristischer Algorithmus zur Minimierung dieses Koeffizienten-Kostenfaktors vorgestellt. Der vorgeschlagene Algorithmus erreicht eine bemerkenswerte Verringerung der Rechenzeit verglichen mit anderen State-of-the-Art Algorithmen, was durch eine eingehende Analyse der Filterkoeffizienten vor der eigentlichen Berechnung und einen neuartigen Ablauf der Koeffizientenzuordnung erreicht wird. Die erzielte Verbesserung der Laufzeit im Vergleich zu anderen State-of-the-Art Algorithmen liegt bei einem Faktor 400.

Eine Verbesserung der Filterstruktur wird durch ein Auswahlverfahren für unterschiedliche digitale FIR Filtertopologien erzielt, das speziell im Hinblick auf die Leistungsaufnahme entwickelt wurde. Die Leistungsaufnahme kann durch das vorgeschlagene Verfahren im Vergleich zu einer konventionellen Filterstruktur um bis zu 15% reduziert werden.

Es wird eine verbesserte Architektur für Multiplizierer vorgestellt, die einen neuartigen Kodierungsansatz für multipliziererlose Shift-and-Add Architekturen verwendet. Hierdurch wird eine Reduktion des Hardwareaufwands um 25% im Vergleich zu bestehenden State-of-the-Art Ansätzen erzielt.

Der zweite Schwerpunkt dieser Arbeit liegt bei der Optimierung der Leistungsaufnahme von digitalen Receivern, die komplexe FIR Filter beinhalten und damit ein Anwendungsfall für die bereits diskutieren Verfahren darstellen, darüber hinaus aber auch numerisch kontrollierte Oszillatoren (numerically controlled oscillators, NCOs) benötigen und eine wohlüberlegte topologische Auslegung erfordern. Es erfolgt eine umfassende Betrachtung von ROM-basierten NCOs, welche unterschiedliche State-of-the-Art Ansätze zur Verbesserung der Leistungsfähigkeit beinhaltet, insbesondere die Ausnutzung von Symmetrien, Dithering und lineare Approximation. Als Ergebnis ergibt sich ein Designansatz, der eine Berechnung der Parameter für einen ROM-basierten NCO mit möglichst geringer Komplexität erlaubt.

Abschließend wird ein leistungsoptimiertes digitales Front-End für einen schmalbandigen, abstimmbaren digitalen FM Receiver in einer 130 nm CMOS Technologie entworfen und synthetisiert. Verglichen mit der konventionellen Architektur wird dabei eine Verbesserung der Leistungsaufnahme um 60% erzielt.

# List of Abbreviations

| | |
|---|---|
| AAF | Anti-Aliasing Filter |
| AD | ADder cell |
| ADC | Analog-to-Digital Converter |
| ASIC | Application Specific Integrated Circuits |
| BW | Band Width |
| CIC | Cascaded Integrator Comb |
| CMOS | Complementary Metal-Oxide-Semiconductor |
| CT | Continuous Time |
| DAC | Digital-to-Analog Converter |
| DDC | Digital Down Converter |
| DDS | Direct Digital Synthesizer |
| DF | Direct-Form |
| DFE | Digital Front End |
| DFF | Delay Flip Flop |
| DL | DeLay cell |
| DSP | Digital Signal Processing |
| DT | Discrete Time |
| EDA | Electronic Design Automation |
| FA | Full-Adder |
| FB | Feed-back |
| FCW | Frequency Control Word |
| FF | Feed-forward |
| FIR | Finite Impulse Response |
| FPGA | Field Programmable Gate Array |
| HB | Half-Band |
| HVL | Hardware Verification Language |
| IBN | In-Band Noise |
| IF | Intermediate Frequency |
| IIR | Infinite Impulse Response |
| IP | Intellectual Property |
| LAP | Linear APproximation |
| LPF | LowPass Filter |
| LSB | Least Significant Bit |
| LUT | Look-Up-Table |
| MA | Multiplier Adder |
| MB | Multi-Band |
| Mb | Multi-bit |

| | |
|---|---|
| MILP | Mixed Integer Linear Programming |
| MOA | Multi-Operand Adder |
| MSB | Most Significant Bit |
| MU | MUltiplier cell |
| NB | Narrow-band |
| NCO | Numerically Controlled Oscillator |
| OSR | OverSampling Ratio |
| P-M-E | Parks-McClellan Equiripple |
| POT | Power-Of-Two |
| PP | Polynomial Programming |
| PPD | PolyPhase Decomposition |
| PSD | Power Spectral Density |
| RCA | Ripple Carry Adder |
| RF | Radio Frequency |
| ROM | Read Only Memory |
| RTL | Register Transfer Level |
| SA | Structural Adder |
| Sb | Single-bit |
| SDC | Synopsys Design Constraints |
| SDF | Standard Delay Format |
| SFDR | Spurious-Free Dynamic Ratio |
| SNR | Signal-to-Noise Ratio |
| TF | Transposed-Form |
| Tcl | Tool command language |
| VCD | Value Change Dump |
| VHDL | VHSIC Hardware Description Language |
| $\Sigma\Delta$M | Sigma Delta Modulator |

# List of Symbols

| | |
|---|---|
| $f_s$ | Sampling frequency in Hz |
| $f_{pb}$ | Passband frequency in Hz |
| $f_{sb}$ | Stopband frequency in Hz |
| $\Delta_f$ | Transition bandwidth in Hz |
| $f_c$ | Center frequency |
| $f_B$ | Base-band frequency |
| $f_{signal}$ | Signal frequency |
| $\delta f$ | Channel band-width |
| $\omega_s$ | Sampling frequency in rad/s |
| $\omega_{pb}$ | Passband frequency in rad/s |
| $\omega_{sb}$ | Stopband frequency in rad/s |
| $\Delta_\omega$ | Transition bandwidth in rad/s |
| $\delta_{pb}$ | Passband ripples |
| $\delta_{sb}$ | Stopband ripples |
| $A_{sb}$ | Stopband attenuation |
| $N$ | Filter order |
| $Q$ | Quantization bit-width |
| $k$ | Number of decimation stages |
| $M$ | Decimation factor |
| $W_{i/o}$ | Input/Output bit-width |
| $R_T$ | Computation effort |
| $h_k$ | Filter coefficients |
| $\hat{h}_k$ | Scaled filter coefficients |
| $S_n$ | Coefficient sensitivity |
| $\Delta_{h_k}$ | Coefficient deviation |
| $U_b$ | Upper bound |
| $L_b$ | Lower bound |
| $E^i$ | Elements function with $i$ non-zero term |
| $m$ | ROM address bit-width |
| $k$ | ROM word-length |
| $n$ | Phase bit-width |
| $r$ | Approximation bit-width |
| $p$ | Phase dithering bit-width |
| $a$ | Amplitude dithering bit-width |
| $l$ | Approximation dithering bit-width |

# 1. Overview

The ongoing need for portable and hand-held personal communication systems has encouraged the development of low power, small size, and high performance devices. Therefore, a reliable, flexible, and efficient signal processing is required. Furthermore, due to the rapid scaling and progress of silicon CMOS technology, analog signal processing systems are replaced with their digital counterparts. Finite Impulse Response (FIR) filters are widely used in digital signal processing applications due to their stability and linear phase characteristics. Therefore, the optimized implementation of FIR filters in hardware has gained significant attention in the research community. One crucial optimization aspect is the low power design of the FIR filters, which also is the main interest of this thesis.

## 1.1. Motivation

'The world at your fingertip'
'The world is becoming more digital'

These phrases interpret speed and mobility of data and information. What people considered as luxury a couple of decades ago becomes a necessity of life nowadays. Hand-held devices, portable communication, voice over internet protocol (VOIP), medical devices, etc., imply the rapidly growing demand for a wide range of consumer applications and medical devices with higher battery life time. Therefore low power design is an essential trend in research nowadays.
The reduction of power dissipation can be achieved on system level, register transfer level (RTL), and gate level. On the one hand, the **effort** for reducing power dissipation is increased going from system level to gate level. On the other hand, the **effect** of power reduction is maximized at the system level compared to the gate level, at least regarding dynamic power consumption. Dynamic power is the most dominant component of power consumption in digital filters. Hence, this work concentrates on power optimization methodologies on system and RTL levels.

## 1.2. Outline

- Chapter 2 reviews Sigma Delta Modulator fundamentals, classifications, and architectures.
- Chapter 3 reviews the different sources of power dissipation and the power optimization paradigm.
- Chapter 4 highlights the algorithmic system level nomenclature and the problem notation. Then the description of the proposed novel allocation scheme is given.
- Chapter 5 presents the proposed algorithmic optimization methodology for digital filters. The chapter starts with a graphical explanation of the problem statement. Then a detailed explanation for the proposed algorithm is given and results from the performance evaluation for a set of benchmark filters are presented.
- Chapter 6 shows two methodologies for architectural power optimization. First, a power aware criterion for combining direct-form and transposed-form digital filters to achieve further power reduction is presented. Then, constructing multi-bit decimation filters using single bit filters is given.
- Chapter 7 outlines the backbone assets for this work which are the design procedure and implementation models. This chapter gives a detailed explanation of the deliverables of this work which are a MATLAB toolbox and a set of RTL-based soft IPs.
- Chapter 8 describes the proposed encoding scheme for shift-and-add multipliers to reduce the internal bit-width, consequently reducing the number of full-adders employed by the multiplier.
- Chapter 9 presents the power optimized digital front end for tunable narrowband digital FM receivers.
- Chapter 10 presents the conclusion and dissertation outlook.

## 1.3. Contributions

This work proposes a novel allocation scheme entitled coefficient deviation ($\Delta_{h_k}$). The proposed allocation scheme is used in the pre-processing stage to assure faster computation time for optimization problems such as polynomial programming and mixed integer linear programming. The preliminary idea was published in 2008 [1]. Moreover, a remarkable savings in the computation time of the FIR filter optimization problem has been achieved throughout extensive pre-processing analysis. A reduction by more than a factor of 400 was achieved by the proposed algorithm with the developed heuristic solver, published in 2012 [2].

On the system architectural level, a power aware combination criterion to combine direct-form and transposed-form FIR digital decimation filters which deliver a reduction in power consumption up to 15% compared to the conventional architecture was introduced and published in 2009 [3]. Further, a novel encoding scheme for shift-and-add multiplierless architectures entitled by nested multiplication was presented. The proposed scheme delivers a reduction in the number of full-adders of up to 25% compared to the state-of-the-art encoding schemes employing binary and canonic signed digit (CSD) representations.
On the application level, a power optimized digital front end architecture for narrow-band tunable FM digital receivers was proposed and synthesized. The proposed architecture achieved a reduction in power by more than 60% compared to the conventional architecture and was published in 2010 [4]. A comprehensive study has been carried out on ROM-based numerical controlled oscillators (NCO) which led to developing a design methodology for power optimized NCOs. Further, a numerical controlled oscillator was implemented[1] and a power reduction of 25% compared to the state-of-the-art architectures and 70% reduction compared to the conventional architecture.

## 1.4. Deliverables

- **MSD-toolbox**: a consolidate design framework for digital decimation filters based on Matlab functions and scripts. The developed toolbox supports the design, optimization and implementation[2] of generic multi-stage decimation filters. The MSD-toolbox employs state-of-the-art design methodologies to assure efficient and reliable implementation of the decimation filter. Further, the toolbox integrates a wide set of attributes to enhance the optimization of the decimation filter.

- **VHDL IPs**: a set of RTL-based soft intellectual properties (IPs) have been developed and synthesized for ASIC and FPGA to facilitate the verification of the proposed power optimization methodologies. The delivered IPs are:

  - PPD - a Polyphase Decimation filter employing direct-form and transposed-form structures.

  - CIC - a Cascaded Integrator Comb filter employing non-pipelined and pipelined architectures.

  - NCO - a Numerical Controlled Oscillator employing $\pi/2$ or $\pi/4$ sinusoidal symmetry and piecewise linear approximation.

[1] RTL synthesis + FPGA prototype
[2] RTL synthesis + place-and-route

- DFE - a Digital Front End module employing a quadrature mixer, complex mixer, decimation stage, numerical controlled oscillator, and a digital control word.

# 2. Sigma Delta ADCs

## 2.1. Introduction

This chapter gives a brief review of decimation filtering fundamentals. Primarily, survey summarizing the state-of-the-art analog-to-digital converters (ADCs) is given. The predominance of sigma delta ($\Sigma\Delta$) analog-to-digital converter for high conversion accuracy and low power applications is stated afterwards. Detailed explanations on the fundamentals for $\Sigma\Delta$ modulation is given in [5], [6], [7]. Next, several classifications of Sigma Delta modulators is presented. Further, a review of the decimation fundamentals and classifications is exhibited.

## 2.2. Need of Data Converters

The aggressive scaling of integrated circuit technology, has enabled binary computations to be performed by very complex data applications with low energy levels and higher speed [8], [9]. Digital circuits are robust and every year the speed and density of digital circuits are increased, leading to the dominance of digital integrated circuits in a wide range of applications [5]. Recently, digital signal processing and computing are the main drivers in modern electronic systems [9]. Though, analog circuits can benefit from technology scaling, analog circuits suffer from several limitations. Those limitations are defined by the analog design octagon [10]. Therefore, designers lean toward a system with minimum analog components. Since, real life signals remain analog, data converters are needed to interface with digital signal processing cores [5], [9].

## 2.3. Survey of Analog-to-Digital Converters

The analog to digital converters (ADCs) have progressed greatly in the last two decades [8]. ADCs can be classified into two main categories: Nyquist-rate and oversampled converters. Nyquist-rate ADCs individually convert each analog input word into a digital output sample, without reference to earlier inputs. Nyquist-rate ADCs can be classified based on the conversion algorithm used into Flash (parallel), successive approximation (Serial), Interpolating (Counting) and Pipeline [11]. Oversampling converters use a sample frequency much higher than the Nyquist

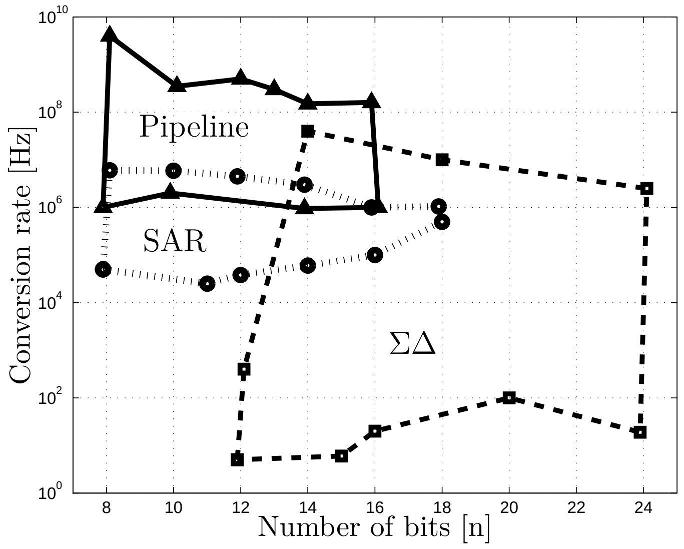


**Figure 2.1.:** Resolution and conversion frequency for more than 500 commercial ADC [8].

frequency of the signal being converted [6].
A learning methodology for ADCs was presented in [8], towards understanding the ADCs functionality and working fundamentals. Furthermore, a quantitative study for the state-of-art ADCs was presented in [12], which examines where ADCs are and where they are headed in terms of enabling performance. This study has examined several performance characteristics for data converters and examined their effect on system performance. The study concluded that, "However, it is clear that despite a general lag behind advances in digital processing, ADCs are nonetheless rapidly improving in all areas of figures-of-merit and will continue to do so, at least in the foreseeable future" [12]. Moreover, a quantitative survey for ADCs is posted on [13], considering various figures-of-merit and design parameters.
Figure 2.1 [8] presents the region of operation for the state-of-the-art ADCs in terms of conversion rate. It shows that, Sigma Delta (ΣΔ) ADCs provides the highest resolution while still achieving high speed [8]. Figure 2.2 shows power consumption in the state-of-the-art ADCs in terms of conversion rate. The power results in the figure are for Nyquist (SAR, Flash and Pipeline) and oversampling (ΣΔ) ADCs. The power consumption for decimation is not included for ΣΔ ADCs. The figure reveals the contribution of ΣΔ ADCs for low power and moderate to high speed applications [13].

## 2.4. Sigma Delta ADC

Oversampling converters have become very popular during the last decade. Figure 2.3 shows the block diagram of an oversampling ΣΔ ADC that includes anti-aliasing filter (AAF), sigma delta modulator (ΣΔM), and decimation filter (digital

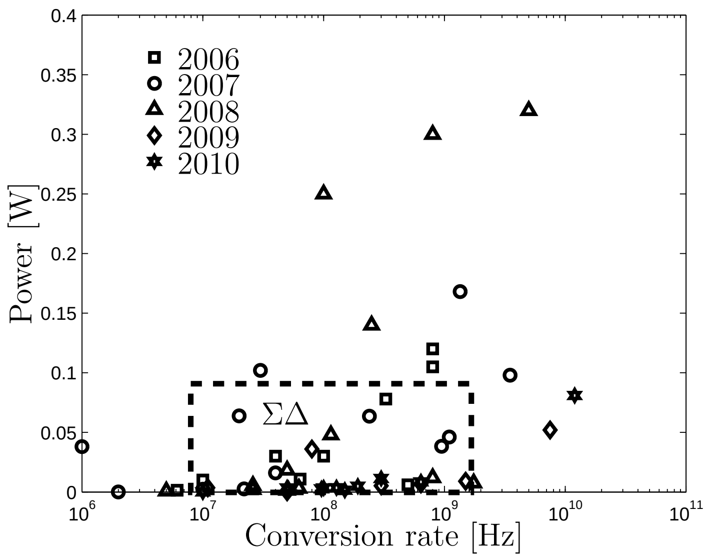


**Figure 2.2.:** Power consumption and conversion frequency for state-of-the-art ADCs, survey 2006-2010 [13].

filter+downsample), where $f_s$ is the sampling frequency, $f_B$ is the maximum signal frequency, OSR is the oversampling ratio defined by $f_B = f_s/2\text{OSR}$, and $M$ is the decimation factor for downsampling. The AAF prior the ΣΔM is used to remove the out-of-band noise to prevent it from aliasing back by oversampling. The ΣΔM, known also as the noise shaping modulator, shapes the quantization noise by having a small fraction of the quantization noise within the band of interest and spreading most of it to the out-of-band range. The modulator consists of a loop filter $H(z)$, a quantizer (single or multi-bit), and a feedback digital-to-analog converter (DAC).

## 2.4.1. Sigma Delta Modulator Classifications

Sigma Delta modulators (ΣΔMs) are classified according to different criteria such as topology, implementation, structure, and architecture. The choice of the mod-

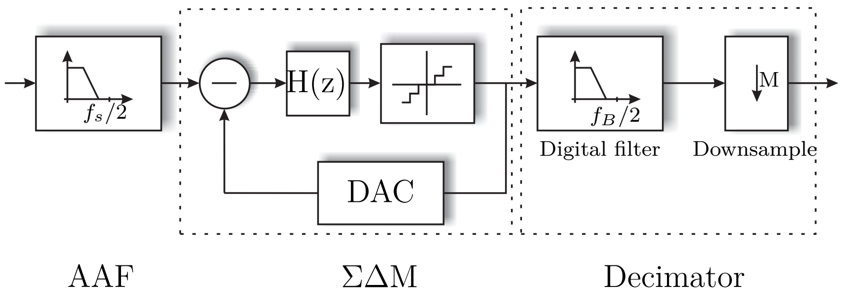


**Figure 2.3.:** Sigma Delta ADC block diagram.

**Table 2.1.:** Summary of Sigma Delta topologies advantages and drawbacks [6], [7], [18]

| | **$\Sigma\Delta$Ms** | | | |
|---|---|---|---|---|
| | Single loop | | MASH | Multi-bit |
| | Low order FB | High order FF | | |
| **Advantages** | • Simple circuitry<br>• High DR<br>• Stability<br>• Better anti-aliasing | • Large SNR for low OSR<br>• Smaller noise pattern | • Large SNR for low OSR<br>• High DR<br>• Guaranteed stability | • Large SNR for low OSR<br>• Better stability<br>• Smaller noise pattern |
| **Drawbacks** | • High OSR<br>• Presence of noise patterns | • Conditional stability<br>• Need low gain integrator | • Complex digital circuitry<br>• Sensitivity to circuit imperfections | • Complex analog and digital circuitry<br>• Sensitivity to DAC non-linearity |
| **Remarks** | • Preferred for low power design | | • Preferred for bandwidth demanding applications | • Preferred for bandwidth demanding applications |

ulator depends on the performance requirements and the application. Some applications require high speed and high resolution converters. Other applications require low power with moderate resolution.

$\Sigma\Delta$Ms have several topologies known as single loop feedback, single loop feedforward, and cascaded or Multi-Stage Noise Shaping (MASH). On the one hand, single loop, low order $\Sigma\Delta$Ms with feedback topology exhibits a stable modulator with simple circuitry suitable for low power applications but with low signal-to-noise ratio (SNR). On the other hand, high order, single loop $\Sigma\Delta$Ms with feedforward topology exhibits higher SNR but suffers from conditional stability. Although, single bit quantizer affords simple circuitry, multi-bit quantizer provides large SNR with associated complex circuitry. However, a modulator with a multi-bit quantizer with a single bit in feedback combines the advantages of single bit and multi-bit quantizers [14]. A detailed study for $\Sigma\Delta$M topologies can be found in [15]. A detailed study for state-of-the-art MASH $\Sigma\Delta$Ms is given in [16]. While, a brief comparative study has been carried out in [17].

$\Sigma\Delta$Ms are implemented by continuous-time (CT) or discrete-time (DT) circuitry.

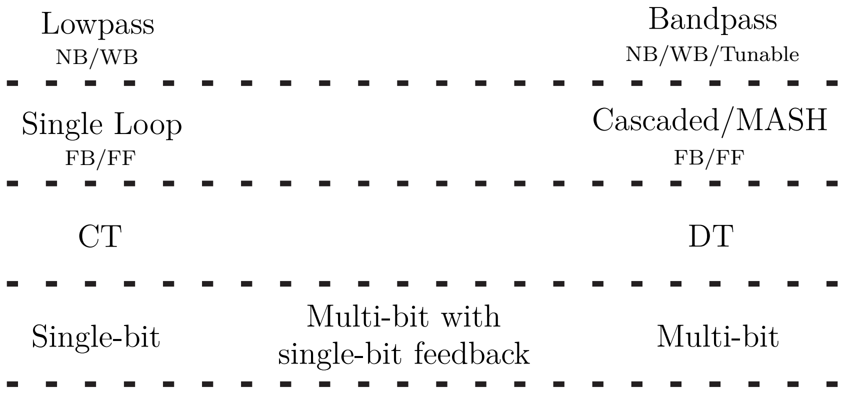


**Figure 2.4.:** Sigma Delta modulator classifications, where NB for narrow band architecture, WB for wide band architecture, FB for feedback topology, FF for feed-forward topology, CT for continuous-time, and DT for discrete-time.

CT ΣΔMs have smaller area than their DT counterparts. Contrary to a CT modulator, and they can operate at high clock speed with less power consumption [18]. On the other hand, DT ΣΔMs achieve excellent matching of the time constants in the modulator [18]. Major advantages of the CT over the DT are the AAF effect and the low power. Discrete-time system can be mapped into continuous-time system by the inverse transformation. Moreover, a DT simulation for the CT ΣΔMs is efficiently done by using the DISCO MATLAB toolbox [19]. A detailed analysis for CT ΣΔMs can be found in [20]. A full study for CT ΣΔMs can be found in [16], [21]. The ΣΔM is suitable for baseband applications (lowpass) as well as RF or IF applications (bandpass). Figure 2.4 summarizes the ΣΔM classifications. Table 2.1 summarizes the advantages and drawbacks for the different ΣΔM topologies.
A concise overview for ΣΔ ADCs can be found in [22]. Moreover, a detailed explanations on the fundamentals of ΣΔ ADCs are given in [5], [6], [7]. Furthermore, a quantitative analysis for ΣΔ ADCs can be looked at [23]. Following, the digital filtering and downsampling stage, known as the decimation filter are discussed.

### 2.4.2. Sigma Delta Decimation Filter

The ΣΔ modulator output consists of the input signal together with noise components. However, there are different noise components within the modulator output. These noise components are the out-of-band noise, the modulation noise, the in-band noise, and the quantization noise. The designers put a lot of effort to assure that the quantization noise is the dominant noise component, since the modulator react as a noise shaper for this particular noise component [5], [6], [7].
The noise shaper (ΣΔM) shapes the noise in a way, so that a small fraction of the quantization noise lies within the in-band range (low frequencies) and it grows rapidly with increasing frequencies, as shown in Fig. 2.5 by the solid line. As shown in Fig. 2.5 the in-band range is defined as $[0 - f_B]$.

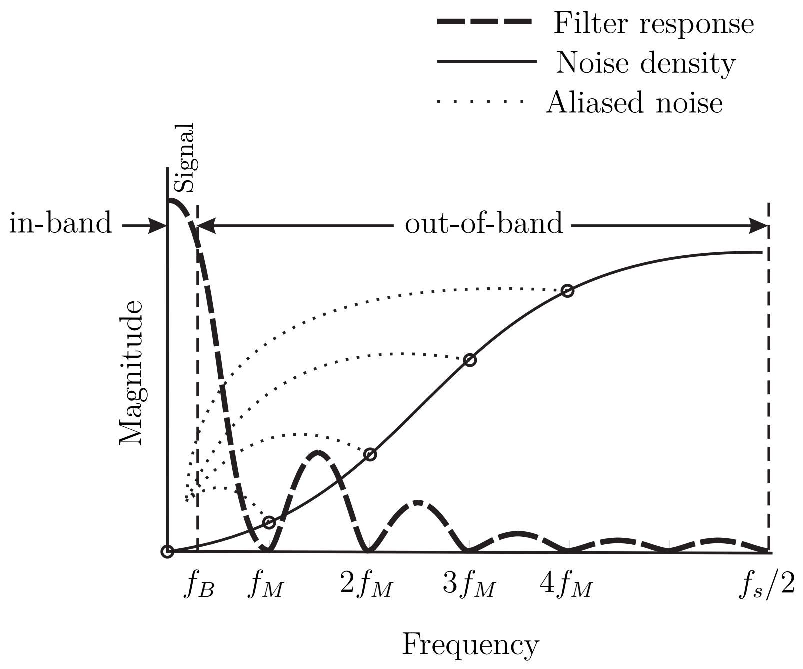


**Figure 2.5.:** Desired decimation response.

A filter stage is needed after the modulator to remove the out-of-band noise and to attenuate the folded noise. The folded noise is represented by the dotted lines in Fig. 2.5. This aliasing effect causes a noise penalty in the in-band range. Therefore, critical filtering is done in the digital domain where it is more robust against circuit imperfections [6].
A single filtering stage with such a steep response is not practically favorable. Consequently, filtering is performed by multiple filtering stages [24]. Oversampling converters relax the requirements of the analog circuitry, at the expense of faster, more complex digital circuitry [6]. After each filtering stage a downsampling stage exists, to decrease the sampling rate through removing the redundant part of the information due to the oversampling. Thus, a decimation stage consists of filtering stage in addition to downsampling stage. The decimation process is known in the literature as sampling rate conversion as well. The downsample process fundamentals are given in Appendix A.
The decimation filter is categorized by architecture, implementation, topology, and structure. In accordance with ΣΔM architecture the decimation filter must be the same, whether lowpass or bandpass architecture. However, for a particular application a bandpass decimation filter is implemented using a downmixing stage followed by a lowpass decimation stage.
Figure 2.6 shows the spectral explanation of the decimation process. The ΣΔM output signal $x(n)$ is first filtered to isolate the band of interest. Afterwards, the resulting signal is directly reduced in sampling rate by $M$. Bandpass and lowpass decimation filters are identical, with the exception of using bandpass filter ($h_{BP}$) rather than lowpass filter ($h_{LP}$), as shown in Fig. 2.6. Sampling rate conversion

**Table 2.2.:** Decimation filter classifications

| | |
|---|---|
| Frequency Response | |
| Lowpass | Bandpass |
| Impulse Response | |
| FIR | IIR |
| Topology | |
| Direct-form | Transposed-form |
| Structure | |
| PPD | CIC |
| Implementation | |
| HB | MB |

process (decimation) can be regarded as a modulation process, as the spectrum of the signal $x(n)$ is located at harmonics of the sampling frequency, which is translated back to the baseband [25], [26], [27]. Hence, the downsampling process is an implicit down mixing process [25], as shown in Fig. 2.6 by the translation of the bandpass signal $X_{BP}$ to baseband signal $Y$ after downsampling.
The state-of-the-art decimation structures are the polyphase decomposition (PPD) filter, the half-band (HB) filter, and the cascaded integrator comb (CIC) filter.
The transposed-form topology is often used for multiple constant multiplication (multiplier block) optimization, hence the filter coefficients form a common space. Hence, FIR filters have a flat group delay response [24], as commonly used in audio applications. On the other hand, IIR filters are preferable in sensor applications such as in gyroscopes. Table 2.2 summarizes the decimation filter classifications. It should be noted that only the topologies and structures for the FIR filters are presented. The advantages and drawbacks of each FIR structure are discussed in more details in chapter 7.
Decimation filter efficiency is directly related to the decimation filter type, order and the architecture used in the implementation [7]. The cascade of two or more decimation stages is described for reduction in filter arithmetic [24]. Whereas, the sampling rate is reduced gradually resulting in much less filtering requirements on the lowpass filters at each stage [28]. A procedure for the proper choice for decimation factor of the stages was presented in [24] and [28]. Detailed analysis for multistage decimation filtering is given in [25], [26], [28]. Numerical examples for multistage decimation are given in [7], [25]. Figure 2.7 shows the multi-stage decimation network and the frequency specifications for $i$-stage, where $x(n)$ is the ΣΔM output, $y(d)$ is the decimation filter output. Illustratively, the first lowpass filter stage (LPF) has an input sampling rate $f_{s0}$, output sampling rate $f_{s1} = f_{s0}/M_1$, passband frequency $f_{pd}$ remains the same, and stopband frequency $f_{sb} = f_{s1} - f_{sb}$ which is very relaxed in comparison to $f_{sb}$. Consequently, this relaxed filter specifications leads to less filter order.

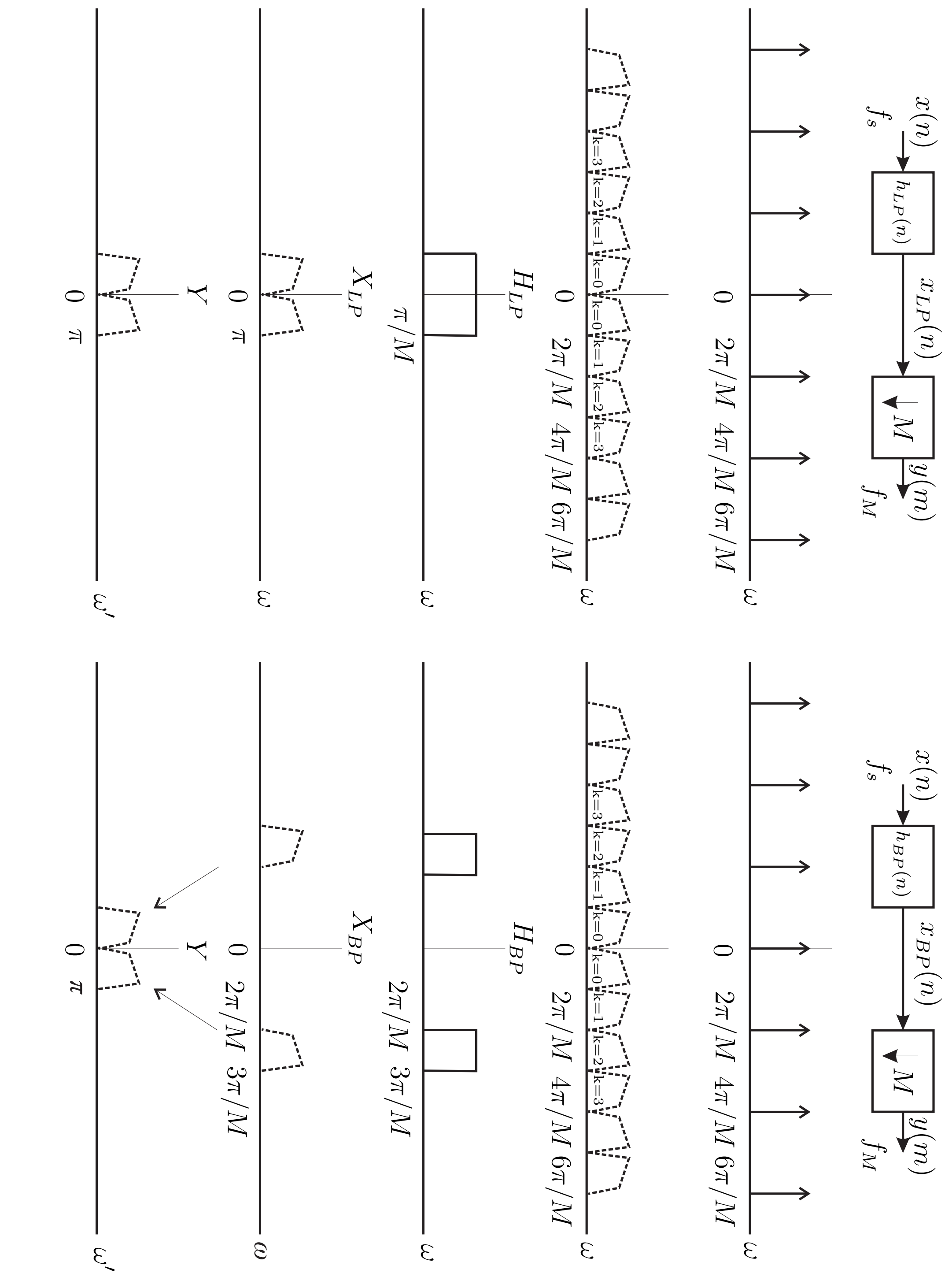


**Figure 2.6.:** Decimation spectral.

The overall decimation factor $M$ is given as

$$\begin{aligned} M &= \prod_{i=1}^{k} M_i \\ &= f_{s0}/f_{sk} \end{aligned} \tag{2.1}$$

The intermediate sampling frequencies are given by

$$f_{si} = \frac{f_{s(i-1)}}{M_i}, \qquad i = 1, 2, \cdots \tag{2.2}$$

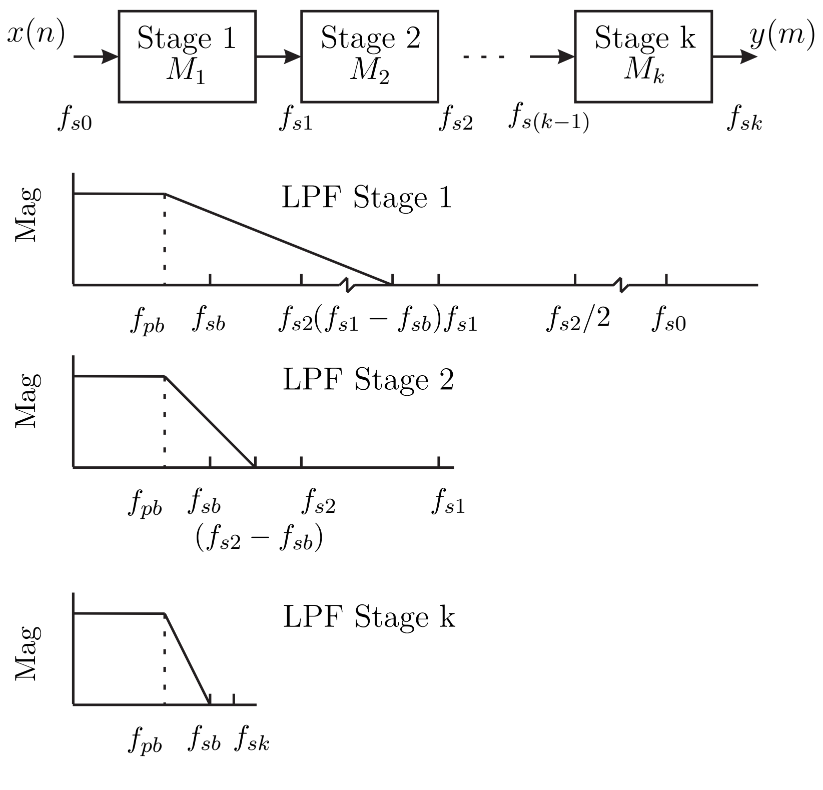


**Figure 2.7.:** Typical spectra for multi-stage decimation filter, where $M_i$ is the decimation factor for stage $i$, $f_{pd}$ is the passband frequency, $f_{sb}$ is the stopband frequency, $f_{s0}$ is the initial sampling frequency, $f_{sk}$ is the final sampling frequency with intermediate sampling frequencies $f_{s1}, f_{s2}, \cdots, f_{s(i-1)}$.

**Table 2.3.:** FIR vs. IIR digital filters

| FIR | IIR |
|---|---|
| Linear phase | No exact linear phase |
| Always stable | Critically stable |
| Good quantization properties | Poor quantization properties |
| High order filters | Low order efficient filters |
| Very efficient for decimation | Less efficient for decimation |

**Table 2.4.:** Digital filter design and implementation parameters

| Design Parameter | Symbol |
|---|---|
| Passband frequency | $f_{pb}$ |
| Stopband frequency | $f_{sb}$ |
| Passband ripples | $\delta_{pb}$ |
| Stopband ripples | $\delta_{sb}$ |
| Stopband attenuation | $A_{sb}$ |
| Sampling frequency | $f_s$ |
| Filter order | $N$ |
| Filter coefficients | $h_k$ |
| Quantization bit-width | $Q$ |
| number of decimation stages | $k$ |
| decimation factor | $M$ |

## 2.5. Decimation Filter Design Parameters

Sampling rate converters involve classical filters, whether lowpass or bandpass, for band selection and noise filtering [29]. The tendency towards digital signal processing (DSP) is increasing and extending simultaneously with the continuous advances in silicon technology [5]. Therefore, digital filters are the main concern for this work. Digital filters can be implemented either as a finite impulse response (FIR) or infinite impulse response (IIR) filter. Table 2.3 summarizes the advantages and disadvantages for both FIR and IIR digital filters [29]. This work considers only FIR decimation filter. Digital FIR filters can be designed using several techniques such as windowing and equiripple techniques. This work considers the Parks-McClellan equiripple algorithm for designing an optimal FIR filter. The filter design process imports the design specifications and exports the implementation parameters. The filter design specifications and implementation parameters are given in Table 2.4. The resultant filter coefficients are with infinite precision. Consequently, the filter coefficients are presented in fixed precision format for practical hardware implementation. Subsequently, the precise number of bits required for representing the filter coefficients in fixed precision has to be determined, which is defined by $Q$.

## 2.6. Summary

The primary purpose of the decimation filter is filtering the noise aliased back into the in-band range due to oversampling. Furthermore, the secondary purpose is to transform a high rate narrow bit-width data stream of the $\Sigma\Delta$M to low rate wide bit-width data stream (high resolution). Decimation is performed by multiple cascaded decimation stages.

# 3. Low Power Design Aspects

## 3.1. Introduction

This chapter addresses concisely the components of power consumption in VLSI circuits, power optimization at multiple design levels, power optimization approaches at each individual design level, and power analysis employing modern electronic design automation tools (EDAs).

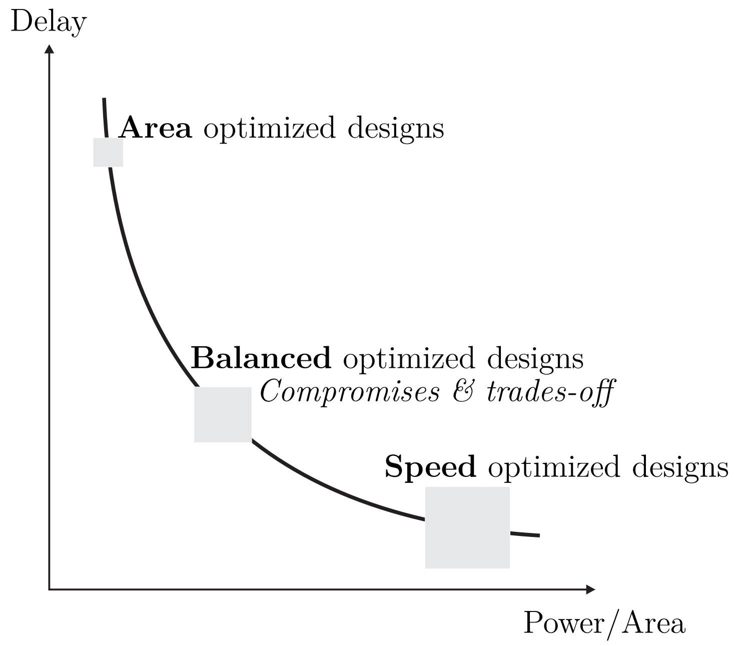

Delay
**Area** optimized designs
**Balanced** optimized designs
*Compromises & trades-off*
**Speed** optimized designs
Power/Area

**Figure 3.1.:** VLSI design trades-off.

Low power design has become a major concern in VLSI circuit design, due to the continuous advances in silicon technology. The aggressive shrinking in the silicon technology has increased the gate density and the clock frequency [9]. But power dissipation rises simultaneously with increasing clock frequencies. Persistently, a power speed trades-off or power area trades-off is a crucial issue in modern VLSI design. On the one hand, small area designs suffer from drastic latency [30], [31], as shown in Fig. 3.1. On the other hand, fastest designs suffer from drastic power and area overhead [30], [31], as shown in Fig. 3.1. Therefore, the major trend for

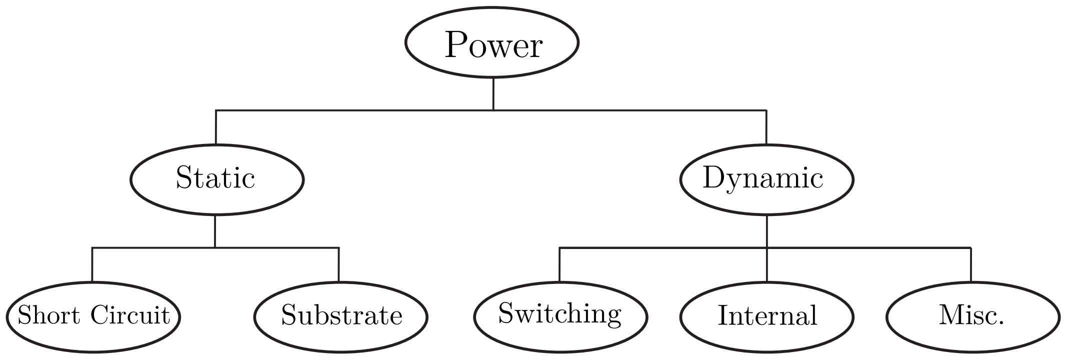


**Figure 3.2.:** Components of power consumption in VLSI circuits.

designers is to balance the trade-off while meeting the performance requirements. The power dissipation can be scaled from one CMOS technology node to the next based on dedicated scaling algorithms. Power dissipation is scaled according to constant voltage scaling by $\alpha$ [32], where $\alpha$ is the ratio between the CMOS technology nodes. On the other hand, power dissipation is scaled by $1/\alpha^2$ according to constant field scaling [32].

## 3.2. Sources of Power Dissipation

Power consumption is a process specific criterion, since each technology has different power profile characteristics. Therefore, at first, the components of power consumption in VLSI circuit are defined. The major components of power dissipation in VLSI circuit are briefly summarized in Fig. 3.2, mathematically analyzed by (3.1–3.6), and modeled as shown in Fig. 3.3. The total power ($P_{Total}$) consumption of a cell is defined by (3.1).

$$P_{Total} = P_{Static} + P_{Dynamic} \tag{3.1}$$

The dynamic power ($P_{Dynamic}$) is the power dissipated when the circuit is active. In other words, when the voltage of a net is changing or switching between high and low [33], [34]. The dynamic power components of the total power are depicted in (3.2).

$$P_{Dynamic} = P_{Switching} + P_{Internal} + P_{Misc} \tag{3.2}$$

where the switching power ($P_{Switching}$) is the power dissipated by charging and discharging a load capacitance ($C_L$) [33], [34]. The load capacitance is defined at the output node of the cell by the sum of wire ($C_W$) and gate capacitances ($C_g$). The process of charging and discharging the output load capacitance of a cell is emulated by the switching current ($I_{SW}$) shown in Fig. 3.3.a. The switching power is proportional to the supply voltage ($V_{dd}$), the load capacitance ($C_L$), the net toggle rate ($T_R$), and the clock frequency ($f$), as depicted in (3.3). Equation 3.3

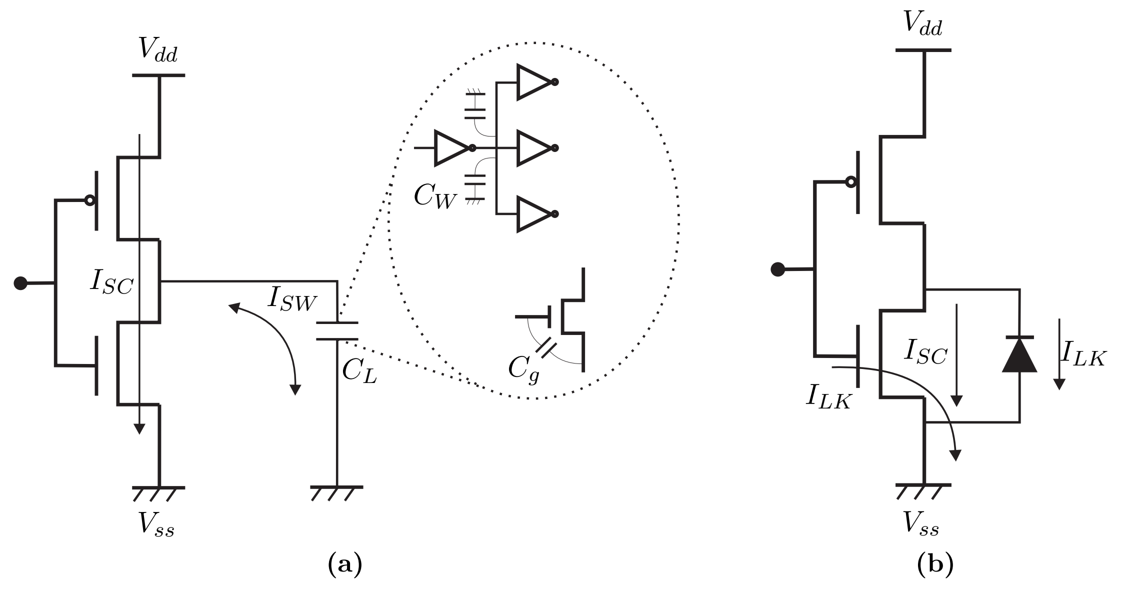


**Figure 3.3.:** Modeling of power consumption components (a) dynamic (b) static.

reveals the dominance of the supply voltage $V_{dd}$ and the clock frequency $f$ in reducing dynamic power dissipation by reducing either of them or both.

$$P_{Switching} \propto T_R \times C_L \times {V_{dd}}^2 \times f_s \tag{3.3}$$

The internal power ($P_{Internal}$) is the power dissipated within a cell itself by charging and discharging internal capacitance. Moreover, the momentary short circuit between the P and N transistors dissipates an amount of power considered also as internal power, emulated by the short circuit current ($I_{SC}$) shown in Fig. 3.3.a. The internal capacitance is a function of the input transition time ($t_T$) and the output capacitance ($C_o$) [33], [34].

$$P_{Internal} \propto F(t_T, C_o) \tag{3.4}$$

Moreover, there are the power dissipated by the clock tree and the internal glitches. This is modeled as miscellaneous components of power dissipation. The miscellaneous power consumption can be defined by

$$P_{Misc} = P_{Clock} + P_{Glitch} \tag{3.5}$$

Whereas, the static power ($P_{Static}$) is the power dissipated by the circuit when it is inactive [33], [34]. The static power components of the total power consumption is depicted by

$$P_{Static} = P_{ShortCircuit} + P_{Substrate} \tag{3.6}$$

The short circuit power ($P_{ShortCircuit}$) is the power dissipated by the source-to-drain sub-threshold leakage [33], [34]. The source-to-drain short circuit leakage is emulated by the short circuit current ($I_{SC}$) shown in Fig. 3.3.b. The substrate power ($P_{Substrate}$) is the power dissipated in the diffusion layer and the substrate. The power dissipated by the bulk leakage is emulated by the bulk leakage current ($I_{LK}$) shown in Fig. 3.3.b.

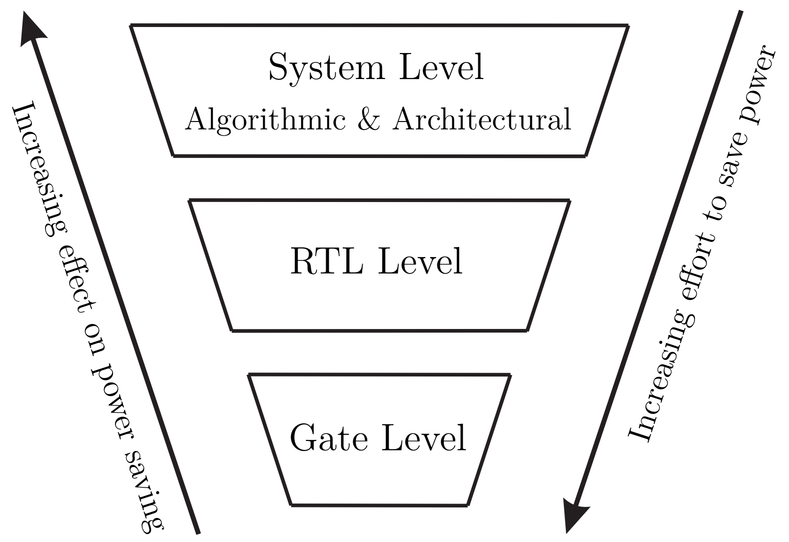

System Level
Algorithmic & Architectural
RTL Level
Gate Level
Increasing effect on power saving
Increasing effort to save power

**Figure 3.4.:** Design levels.

## 3.3. Power Optimization Approaches

A reduction of the power dissipation can be achieved at several levels in the design process, as shown in Fig. 3.4. The design process is categorized into three major levels: the system level, the register transfer level (RTL), and the gate level (also known as physical level). The system level is the highest design level, where one can model or define the design abstractly, algorithmically, or architecturally. On the RTL level (front-end) the designer has to have a full awareness of the design details. The physical or the gate level (back-end) is the lowest level in the design hierarchy and the most sophisticated level. Each level of design has different power optimization approaches with certain impact on the reduction of power consumption, whether static or dynamic power dissipation. Not all optimization techniques may be suitable for the design, so that the designer has to assess the different techniques for complexity, risk, and cost [30]. Excessive effort is spent on the power optimization at the gate level for slight reduction in the power consumption (if the leakage power is not important), as shown in Fig. 3.4. Table 3.1 exhibits several power optimization approaches at the individual design levels and their impact on power reduction. Apprehensively, the gate level power optimization approaches have a direct dominant effect on the leakage power consumption reduction. Thus, the back-end power optimization approaches is used after exhausting all the approaches for dynamic power reduction [30]. Otherwise, where the reduction of the leakage power dissipation is of importance. For significant power saving, a combination of two or more optimization approaches is preferable. As an example, combining clock gating and power gating proposed in [35] for optimizing both dynamic and leakage power. Moreover, controlling $V_{dd}$ and $V_{TH}$ simultaneously as proposed in [36] for optimizing both dynamic and leakage power.
Since, the physical level optimization methodologies are beyond the scope of this

**Table 3.1.:** Power optimization approaches at several design levels and their impact on power optimization

| Level | Approach | Impact |
|---|---|---|
| System | Frequency scaling | Dynamic |
| | Multi-$V_{dd}$[1] | Dynamic |
| | Power efficient IPs | Dynamic |
| | Power optimization algorithms | Dynamic |
| RTL | Pipelining | Dynamic |
| | Clock gating | Dynamic |
| | Power gating[1] | Leakage |
| | Multi-$V_{TH}$[1] | Leakage |
| Gate | High-K | Leakage |
| | Copper/Gold interconnects | Leakage |
| | Body bias | Leakage |
| | SOI | Leakage |
| | SiGe substrate | Leakage |

[1] at gate level as well.

work, the major focus of this work is on the reduction of the dynamic power through system and RTL levels. However, detailed analysis for gate level power optimization methodologies can be found in [37] and [38].

## 3.4. Power Analysis

The continuous development on silicon technology increases the design complexity implying increase in the power dissipation. An efficient power budget analysis at different design phases can lead to a successful power management for power reduction. Thus, power analysis is integrated into the design flow at the individual design level to address power dissipation and power distribution problems. Power analysis is performed with the aid of electronic design automation (EDA) tools to estimate and analyze the power consumption in circuit designs by building a detailed power profile based on circuit connectivity and switching activity [33], [39]. Table 3.2 shows the different types of power analysis using state-of-the-art EDA. Rough estimate of the power consumption can be obtained from the power spreadsheets. However, accurate power analysis is performed on the RTL and gate level by using a netlist and simulation activity files. A netlist describes the connectivity of a design and it is exported from a logic synthesis tool (front-end) or place-and-route (PAR) tool (back-end). Switching activity files are generated from simulation tools and contain the design nets probability and toggle rate. The switching activity can be exported as a SAIF (Switching Activity Interchange Format) file or as a VCD (Value Change Dump) file. The SAIF or VCD files are used for average and peak power analysis, respectively. The VCD captures the activity and time of

**Table 3.2.:** Types of power analysis in modern EDA

| | **Power Analysis** | |
|---|---|---|
| Type | Statistical activity based | Event-based |
| Level | RTL or Gate | RTL or Gate |
| Simulation activity | SAIF | VCD |
| Accuracy | Fair | Very accurate |
| Analysis type | Average | Peak |

every event on each net [39]. The VCD-based analysis is extremely accurate since all the factors contributing to power consumption are supported in an accurate form [39].
Modern EDA tools export a detailed result of the power analysis, as shown in Fig. 3.5. Figure 3.5 presents the power analysis results for an empirical example of a FIR filter. As exhibited, the total power consumption is distributed over all the filter components such as multipliers, adders, registers, and other components (clock buffers, IOs ... etc). The power consumed in each individual block is then divided to its dynamic and static components.

## 3.5. Summary

On one hand, power optimization approaches at the system and RTL levels have a remarkable impact on the reduction of the dynamic power consumption. On the other hand, the power optimization approaches at the gate level have a remarkable impact on the reduction of the static leakage power consumption. Accurate power analysis is carried out by modern EDA tools using event-based analysis with VCD simulation activity files.

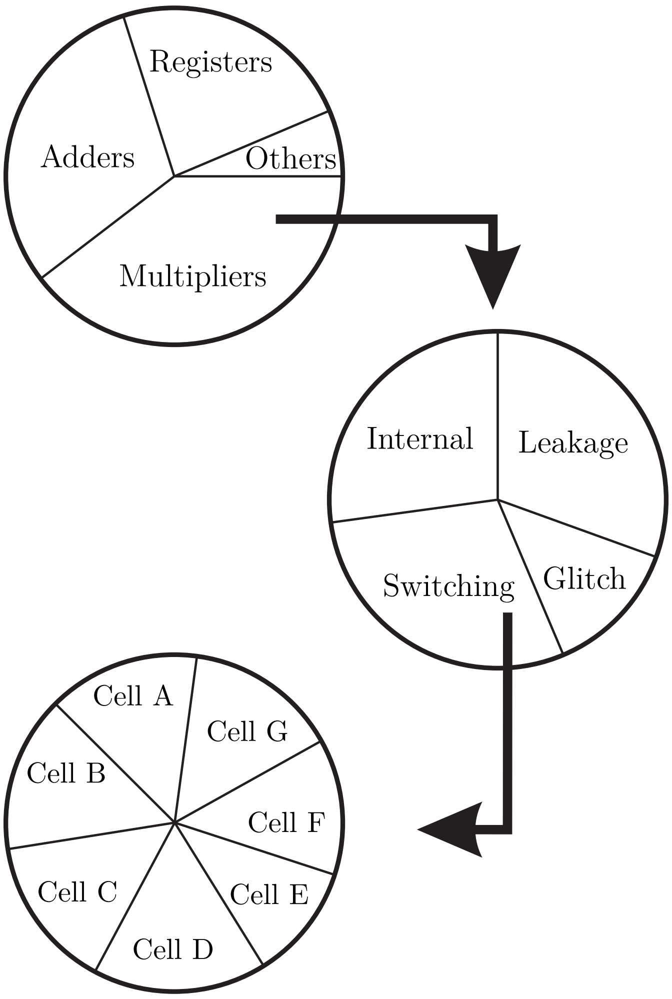


**Figure 3.5.:** Power analysis breakdown.

# 4. Algorithmic System Level Terminology

## 4.1. Introduction

This chapter highlights the nomenclature and terminology used in algorithmic optimization for digital FIR filters. Further, the proposed allocation schemes entitled by deviation, cost and hybrid are given in this chapter. This is a foundation chapter for chapter 5 on algorithmic system level power optimization.

## 4.2. FIR Filter Conventions

The frequency response of a symmetric linear phase FIR filter, as shown in Fig. 4.1, can be separated into a real valued function $H_R(\omega T)$ and a real valued phase function $\Theta(\omega T)$ as depicted in (4.1) [40], [41]

$$H(e^{j\omega T}) = H_R(\omega T)e^{j\omega T} \tag{4.1}$$

where $H_R(\omega T)$ is the zero phase frequency response. For an $N^{th}$ length linear phase FIR filter the zero phase frequency response can be written as depicted in (4.2) [40]

$$H_R(\omega T) = \sum_{k=1}^{M} h_k c(k, \omega T) \tag{4.2}$$

where $h_k$ is the filter coefficients. For a symmetric impulse response and odd filter length $N$ [40]

$$\begin{aligned} c(k, \omega T) =& x_k \cos(\omega T[k-1]) \\ x_k =& \begin{cases} 1, k = 1 \\ 2, k = 2, 3, \cdots, M \end{cases} \\ M =& N/2 + 1 \end{aligned}$$

while for symmetric impulse response and even filter length $N$ [40]

$$\begin{aligned} c(k,\omega T) =& 2\cos(\omega T[k-1]) \\ k =& 1,2,3,\cdots,M \\ M =& (N+1)/2 \end{aligned}$$

The lowpass filter specifications are expected to be given in a form as in (4.3)

$$\begin{aligned} 1-\delta_{pb} \leq & H_R(\omega T) \leq 1+\delta_{pb}, \qquad && \omega T \in [0,\omega_{pb}] \\ -\delta_{sb} \leq & H_R(\omega T) \leq \delta_{sb}, && \omega T \in [\omega_{sb},\pi] \end{aligned} \tag{4.3}$$

where $[0,\omega_{pb}]$ and $[\omega_{sb},\pi]$ are the passband and stopband, respectively. The filter design parameters are defined in Table 4.1. The FIR filter problem is formulated in a mathematical model. In addition to the formulated mathematical model, the distinct formation of the object and subject determines the problem and solver types. The feasible region is depicted between the Gray shaded regions for passband and stopband, as shown in Fig. 4.1.

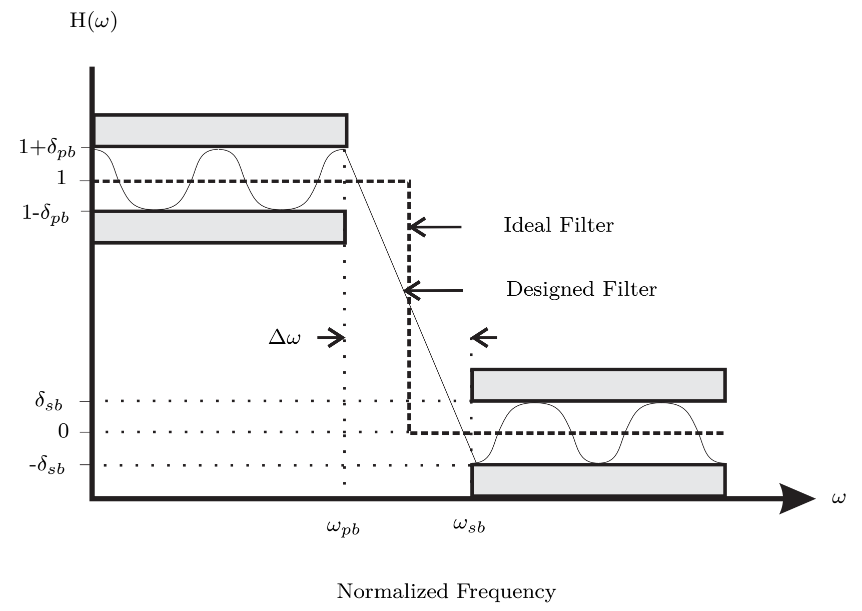


**Figure 4.1.:** Lowpass filter response.

### 4.2.1. Coefficient Quantization

An initial set of FIR coefficients satisfying the required specifications is designed using the Remez exchange or the Parks McClellan algorithms, which are minimax

**Table 4.1.:** FIR Filter Design Parameters

| Symbol | Abbreviation |
|---|---|
| $f_s$ | Sampling frequency |
| $f_{pb}$ | Passband frequency in Hertz or normalized |
| $f_{sb}$ | Stopband frequency in Hertz or normalized |
| $\omega_{pb}$ | Passband frequency in rad/s |
| $\omega_{sb}$ | Stopband frequency in rad/s |
| $\delta_{pb}$ | Passband ripple |
| $\delta_{sb}$ | Stopband ripple |
| $A_{sb}$ | Stopband attenuation |
| $\Delta\omega$ | Transition band |
| $N$ | Filter length |

algorithms. The resulting filter coefficients are given as floating point with infinite precision. In practice, filter coefficients have to be constrained to a finite number of bits in a fixed-point notation. Therefore, the filter coefficients are quantized to finite precision. The difference between the quantized output and input is referred to as the quantization error as given by (4.4).

$$\varepsilon = Q[x] - x \tag{4.4}$$

where Q[x] is the quantized output, x is the input and $\varepsilon$ is the quantization error. For example, consider the number 0.46484375, which needs at least 8-bits to be represented in binary format $0.01110111_2$. If the quantization bit-width is 7-bits then the binary representation would be $0.0111011_2$ which corresponds to 0.4609375. The quantization error would be $\varepsilon = 0.46484375 - 0.4609375 = 3.9 \times 10^{-3}$ for this single coefficient.
A detailed analysis concerning the effect of finite word length is given in [42]. The major effect of reducing the quantization bit-width is observed in the stop and transition bands [42]. The choice of the quantization bit-width is governed by many design parameters such as required speed, power consumption, cost and signal-to-noise ratio (SNR). The quantization process is characterized by two parameters, the binary point position relative to its least significant bit and its quantization mode being either truncation (floor) or round-off (round). A detailed quantization analysis is given in [43]. Although in [42] the appropriate quantization bit-width is estimated according to a heuristic analysis preserving the $\delta_{pb}$ and $\delta_{sb}$, the work in [44] proposes a closed form expression preserving the normalized peak ripple .

### 4.2.2. Coefficient Scaling

After quantization to a finite bit-width the rational coefficients can be presented as given by (4.5).

$$h_k = S_k \sum_{i=1}^{Q-1} b_{k,i} 2^{-i}, \qquad b_{k,i} \in \{0, 1\} \tag{4.5}$$

where $h_k$ are the quantized filter coefficients, $Q$ is the quantization bit-width, and

$$S_k = \begin{cases} 1, & h_k \geq 0 \\ -1, & h_k < 0 \end{cases}$$

Subsequently, the coefficients have to be scaled to be usable in practical hardware implementations. There are two methods of operating on fixed point data: integer and fractional [45]. The integer method interprets the data as integers and performs integer arithmetic [45]. The fractional method assumes the data to be fixed-point rationales bound between -1 and +1 [45]. In this work, the integer fixed-point method is utilized, as integer representation is preferred for RTL modeling, e.g. VHDL or Verilog. Further, it can be easily transformed to other representations, i.e. binary or signed-digit.
Equation (4.6) is used for filter coefficients scaling [40], [45].

$$\begin{aligned} SF &= 2^{S-1} - 1 \\ \hat{h}_k &= \left\lfloor \frac{h_k}{\max|h_k|} \times SF \right\rceil \\ h_k &= \frac{\hat{h}_k}{SF} \Big/ \sum \frac{\hat{h}_k}{SF} \end{aligned} \tag{4.6}$$

where $SF$ is the scaling factor and $S$ is the scaling bit-width, $S \leq Q$ ($S = Q$ is the default setting in this work).
The advantage of using this scaling method is that it normalizes the coefficients with high dynamic range. However, it has a drawback in the rescaling process and the number of non-zero terms compared to the quantized one [45]. The number of non-zero terms is defined as the number of power-of-two (POT) terms within each coefficient. Consider, for example, the coefficient $h_k \approx 0.2795$, as the maximum coefficient, with its corresponding binary representation $h_k = 0.010001111000110_2$, which has 7-POT terms. Scaling of $h_k$ using (4.6) results in $\hat{h}_k = 32767$ with its corresponding binary representation $\hat{h}_k = 0111111111111111_2$ which has 15-POT terms. Consider using the scaling criterion given by (4.7),

$$\begin{aligned} SF &= 2^{S-1} \\ \hat{h}_k &= \lfloor h_k \times SF \rceil \\ h_k &= \hat{h}_k / SF \end{aligned} \tag{4.7}$$

and $\hat{h}_k = 9158$ with its corresponding binary representation $\hat{h}_k = 0010001111000110_2$, which has 7-POT terms exactly as $h_k$. The scaling bit-width ($S$) should be smaller than or equal $Q$ to avoid an overflow of the coefficient largest magnitude. To ensure a constant number of POT terms before and after the scaling process, the scaling criterion given by (4.7) is used in most of this work. Figure 4.2 shows the difference in cost according to scaling (4.6) and (4.7), for the benchmarks defined at chapter 7, section 7.6. The cost indicates the number of non-zero terms. There is about 30% overhead in the number of POT when scaling the filter coefficients as in (4.6).

## 4.2.3. Coefficient Representation

An adequate representation for the filter coefficients is required for efficient hardware implementation. After limiting the infinite precision coefficient through quantization process and scaling it to fixed integer, the representation for hardware implementation needs to be covered. Two common representations of discrete fixed point coefficients are used: binary and signed-digit. Binary representation uses only addition, such as $15_{10} = 01111_2$ or $-15_{10} = 10001_2$, is represented by (+) non-zero terms only. While, signed-digit representation uses addition and subtraction as well, such as $15_{10} = 16_{10} - 1_{10} = 1000\bar{1}_2$ (where $\bar{1}$=$-1$), is represented by (+/−) non-zero terms. Thus the discrete filter coefficients are represented as sums of either power-of-two (POT) or signed-power-of-two (SPT) terms. Binary representation includes one's complement, two's complement and sign-magnitude representations with the coefficient bits $\in \{0, 1\}$. SPT representation, also called signed-digit (SD) representation, on the other hand includes canonic signed digit (CSD) and minimal signed digit (MSD) representations with the coefficient bits $\in \{-1, 0, 1\}$. Often, the SPT representation is considered, because it results in fewer non-zero bits in each multiplier compared to the binary representation [40], [46], [47], [48]. By using the CSD representation, there is a trade-off between the gain of reduced number of non-zero terms in the coefficient and the reduction of further optimization possibilities when combined with approaches like CSE (common sub-expression elimination) [49], [50], [51]. The work in [52] showed that using the binary coefficient representation leads to a great reduction in the complexity of the multipliers compared to using the CSD coefficient representation. As well, the work on [53] revealed the superiority of binary representation over the CSD representation. Although, the results presented by [53] were discussed and criticized in [54], it is agreed that binary CSE has slight advan-

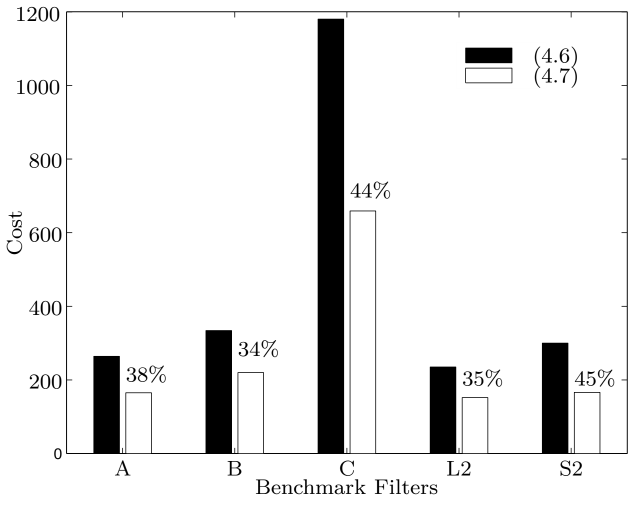


**Figure 4.2.:** Scaling effect on number of non-zero terms using (4.6) and (4.7).

**Table 4.2.:** Two's complement to CSD conversion

| $a'_{i+1}$ | $a'_i$ | $c_i$ | $a_i$ | $c_{i+1}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | $\bar{1}$ | 1 |
| 1 | 1 | 0 | $\bar{1}$ | 1 |
| 1 | 1 | 1 | 0 | 1 |

tage over CSD CSE in terms of logic operations (LO) but CSD is recommended for lower logic depth (LD). LD has a direct influence on the filter critical path delay and indirect effect on the power consumption, whereas, LO has a direct impact on power consumption [55]. The multiplier LD is restricted by the number of non-zero bits of a coefficient [55]. Moreover, a qualitative and quantitative analysis were carried out in [50] and [51] to compare between the different number representations for minimum number of required operations for implementation. These analyses revealed that binary representation has more non-zero bits which makes more redundant common patterns [51]. Binary representation is preferred when the number of constants is increased [50]. CSD representation is preferable for minimum number of non-zero terms, whereas binary representation is desirable for CSE. The conversion can be done according to Table 4.2 [56], where $a'_i$ are the 2's complement bits to be converted to CSD, $a_i$ are the CSD bits after conversion, and $c_{i+1}$ is a generated carry bit. The $c_i$ is initially zero, then it is updated from $c_{i+1}$. Instead, the algorithm described by the following Pseudocode [57] can be used for conversion:

```
$a'_{-1} = 0;$
$\gamma_{-1} = 0;$
$a'_w = a'_{w-1};$
for($i = 0$ to $w - 1$)
{
    $\theta_i = a'_i \oplus a'_{i-1};$
    $\gamma_i = \gamma_{i-1}.\theta_i;$
    $a_i = (1 - 2a'_{i+1})\gamma_i;$
}
```

where $a'_{w-1}.a'_{w-2}...a'_1.a'_0$ is the 2's complement number and its CSD representation is $a_{w-1}.a_{w-2}...a_1.a_0$. Further, an efficient implementation of the binary-to-CSD conversion is found in [58]. It should be noted that, binary-to-CSD conversion can be used for the reduction of number of non-zero terms as well [55]. Furthermore, conversion algorithms to convert from 2's complement to minimal signed digit (MSD) can be found in [59]. The advantage of CSD over MSD, is that CSD has a unique representation for each coefficient. Whereas, MSD has multiple representations with the same number of non-zero terms as CSD [59].

## 4.3. Bounded Search Space

In order to speed up the computation time by reducing the search space candidates for each coefficient, upper bound ($U_b$) and lower bound ($L_b$) are used to limit the search space between these two values. The zero value is considered before it is assumed that at least a single POT is required to represent a coefficient. Modeling the bounds is given in (4.8)

$$L_b(\hat{h}_k) = \begin{cases} 2^L, L = \lfloor \log_2(|\hat{h}_k|) \rfloor, \hat{h}_k > 0 \\ -2^L, L = \lfloor \log_2(|\hat{h}_k|) \rfloor, \hat{h}_k < 0 \\ 0, h_k = 0 \end{cases}$$
$$U_b(\hat{h}_k) = \begin{cases} 2^U, U = \lceil \log_2(|\hat{h}_k|) \rceil, \hat{h}_k > 0 \\ -2^U, U = \lceil \log_2(|\hat{h}_k|) \rceil, \hat{h}_k < 0 \\ 0, h_k = 0 \end{cases} \quad (4.8)$$

A similar approach for bounding the search space was presented in [40], [60], [61], and [62]. In [40] and [60] the upper and lower bounds were determined while finding the value for each coefficient which meets the filter specifications. Thus, these bounding criteria require long computation time.

## 4.4. Coefficient Deviation

Only one allocation scheme based on coefficient sensitivity (described later in section 4.5.) is conventionally used in literature. The allocation scheme is used to presort the coefficients for optimization instead of a systematic ascending/descending allocation. The motivation behind this section is to propose and investigate a new allocation scheme.
The coefficient deviation ($\Delta_{h_k}$) presents the distance between the coefficient and the nearest N-POT integer, where N-POT is the number of non-zero terms. The $\Delta_{h_k}$ is calculated as given in (4.9) and illustrated in Fig. 4.3.a.

$$\Delta_{h_k} = \min(U_b - \hat{h}_k, \hat{h}_k - L_b) \quad (4.9)$$

where $U_b$ and $L_b$ are the scaled upper and lower bounds, respectively as defined by 4.8 and $\hat{h}_k$ is the scaled coefficient. The initial idea of coefficient deviation was driven only for 1-POT and 2-POT terms, which is published in [1]. The motivation behind considering only 1 and 2 POT terms is to achieve aggressive reduction in number of non-zero terms. Since for 1-POT coefficients there is a zero adder and for 2-POT there is 1 adder required. This idea is illustrated by numerical example

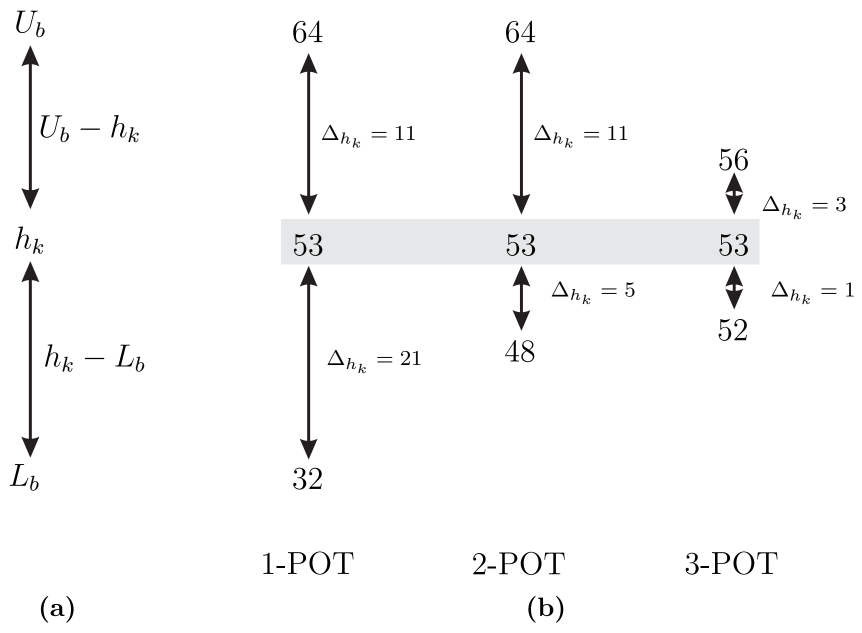


**Figure 4.3.:** Proposed allocation scheme $\Delta_{h_k} = \min(U_b - \hat{h}_k, \hat{h}_k - L_b)$ (a) definition (b) Illustrative example.

in Fig. 4.3.b. The coefficient deviation parameter is equivalent to the minimum distance (discrepancy) between the coefficient and its $U_b$ and $L_b$. The $U_b$ and $L_b$ can be set by (4.8), which is equivalent to 1-POT bounds only. Moreover, the bounds can include N-POT where $N = Q$, where $Q$ is the quantization bit-width. At N-POT there are multiple integers having the same cost. Therefore, the proposed criterion considers the integer with the minimum distance. Illustratively, consider the coefficient 53. Its 2-POT lower bound is:

$$\text{2-POT}(53) = [33, 34, 36, 40, 48]$$

These 5 integers have the same cost of 2 non-zero terms, however, 48 is the nearest lower bound to 53. Similarly, for 3-POT bound is:

$$\text{3-POT}(53) = [35, 37, 38, 41, 42, 44, 49, 50, 52, 56]$$

There are 10 integers having the same cost of 3 non-zero terms, however, 52 is the nearest lower bound and 56 is the nearest upper bound for 53. The $\Delta_{h_k}$ depends on the coefficient, the designated upper and lower bounds, and the quantization bit-width. In order to illustrate the influence of each parameter on the coefficient deviation criterion, the following numerical examples are presented for individual coefficients. Then a simple illustrative example is considered to verify the concept. In order to perform an efficient preprocessing for the problem search space, it is desired to perform quantitative analysis on the coefficients. Therefore, a function is developed to generate a table consisting of three columns. The first column holds the number of POT terms. The second column holds the coefficient deviation value.

The third column holds the integer corresponds to the dedicated cost. Consider 53 and 766, the generated tables are given in Table 4.3 and Table 4.4, respectively.

**Table 4.3.:** Deviation table for 53

| N-POT | $\Delta_{h_k}$ | $\hat{h}_k$ |
|---|---|---|
| 1 | 11 | 64 |
| 2 | 5 | 48 |
| 3 | 1 | 52 |
| 4 | 0 | 53 |
| 5 | 6 | 47 |
| 6 | 10 | 63 |

**Table 4.4.:** Deviation table 766

| N-POT | $\Delta_{h_k}$ | $\hat{h}_k$ |
|---|---|---|
| 1 | 254 | 512 |
| 2 | 2 | 768 |
| 3 | 3 | 769 |
| 4 | 30 | 736 |
| 5 | 14 | 752 |
| 6 | 6 | 760 |
| 7 | 2 | 764 |
| 8 | 0 | 766 |
| 9 | 1 | 767 |
| 10 | 257 | 1023 |

A linear-phase low-pass FIR filter with normalized passband and stopband edge frequencies at $f_{pb} = 0.05$ and $f_{sb} = 0.25$, respectively, is considered as a case study. The desired ripple in the passband and the stopband are $\delta_{pb} = 0.0575$ and $A_{sb} = 30$ dB, respectively. The quantization bit-width $Q = 10$ and the filter length $N = 12$. Due to symmetry the optimization is processed on half of the coefficients. The quantized filter coefficients ($h_k$) are:

$$h_k = [0.13935, 0.12651, 0.10363, 0.075543, 0.047644, 0.029569]$$

The corresponding scaled coefficients ($\hat{h}_k$) are:

$$\hat{h}_k = [143, 130, 106, 77, 49, 30]$$

Figure 4.4 shows the generated deviation tables using the developed function for 1-POT, 2-POT, and $(m-1)$-POT, respectively, where $m$ is the number of non-zero terms for each coefficient. Figure 4.5 shows the frequency response for an optimized coefficients without considering the allocation scheme, i.e. with a systematic order on one hand. On the other hand, for 1-POT, 2-POT and (m-1)-POT, the corresponding tables for each coefficient is given in Fig. 4.4. Rounding to the nearest integer with minimum $\Delta_{h_k}$ is done according to the second column. The optimization procedure is a straight forward process. It is achieved by going through the coefficient (with their systematic order), then rounding the coefficient to its nearest integer with minimum deviation. At 1-POT there are only two candidates for rounding the coefficients, 130 and 30. Since they have the minimum $\Delta_{h_k} = 2$. The frequency response after rounding 130 to 128 and its corresponding mean error (ME) and cost are shown in Fig 4.6.a. Then comes the frequency response after rounding 30 to 32 and its corresponding mean error (ME) and cost are shown in Fig 4.6.b. Finally, the frequency response after rounding 77 to 64 and its corresponding mean error (ME) and cost are shown in Fig 4.6.c. It is obvious that the

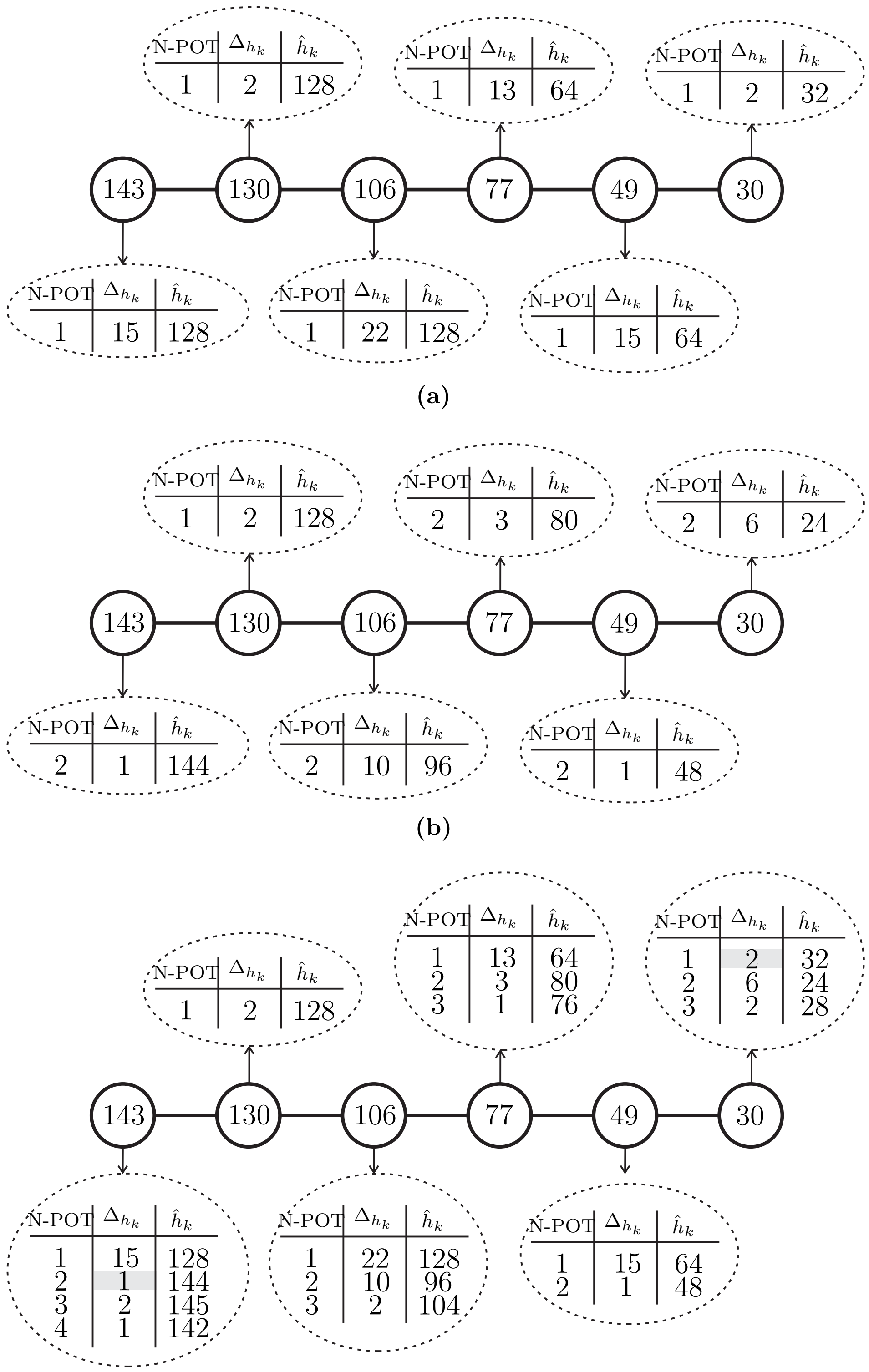


**Figure 4.4.:** Generated deviation tables for (a) 1-POT (b) 2-POT (c) $(m-1)$-POT for $\hat{h}_k = [143, 130, 106, 77, 49, 30]$.

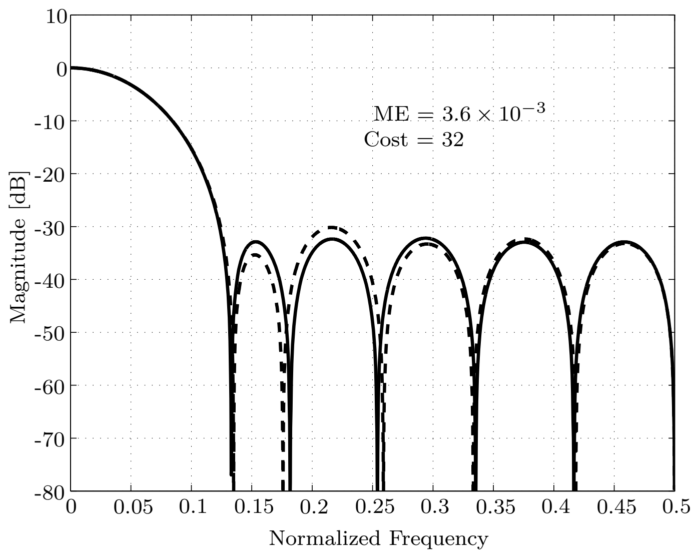
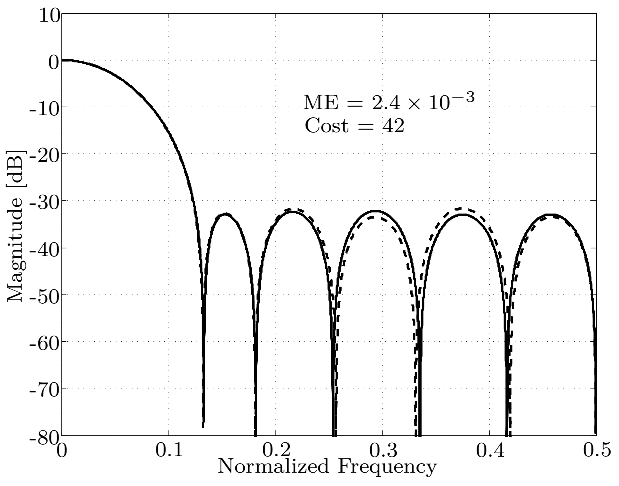
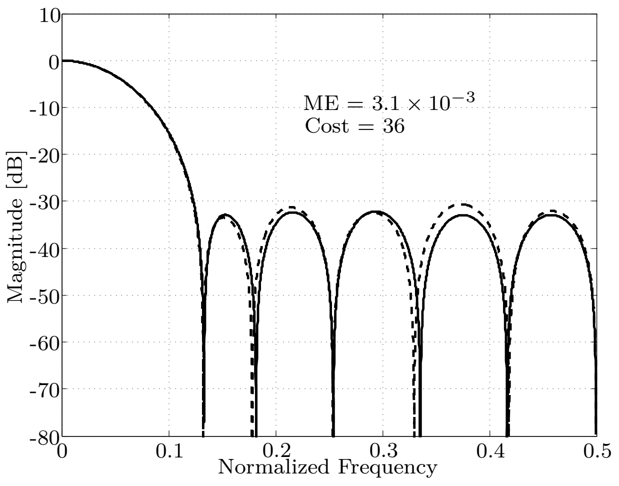


**Figure 4.5.:** Frequency response for N-POT without presorted allocation scheme (143→144, 130→128, 106→104, 77→76) for $\hat{h}_k = [143, 130, 106, 77, 49, 30]$.

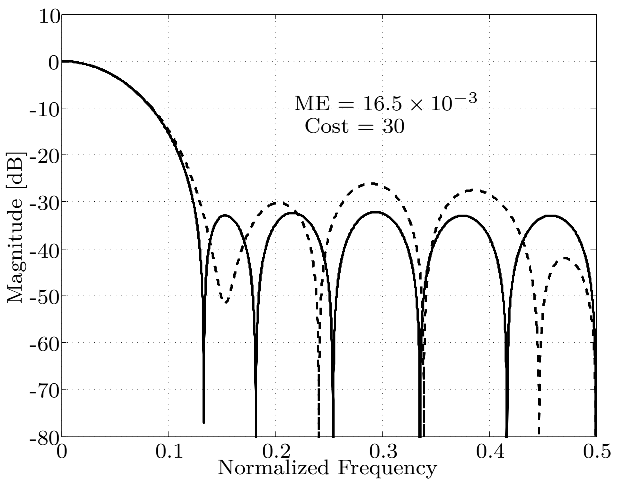
third coefficient is excluded since it violets the filter response. It should be noted that, such optimization process requires two optimization iterations, i.e. to loop through the coefficients set two times. At the first loop, the coefficients 130 and 30 are optimized. Afterwards, in the second loop the coefficient 77 is optimized. At each coefficient, the response is checked in order to preserve the filter specifications. It shows that, for 1-POT the number of non-zero terms is reduced from 44 to 36 non-zero terms which is about 18% for an acceptable error range.
In contrast, at 2-POT there are four candidates for optimization the coefficients: 143, 49, 130, and 77 (according to minimum deviation) compared to 1-POT scheme. The coefficient 143 is rounded to 144 and the corresponding frequency response with its ME and cost are shown in Fig. 4.7.a. Then 49 is rounded to 48 and the corresponding frequency response with its ME and cost are shown in Fig. 4.7.b. Then comes 130 to be rounded to 128, and 77 is rounded to 80 with the relative responses shown in Fig. 4.7.c and Fig. 4.7.d. Last but not least, comes 30 which is rounded to 32, but it violets the filter response specifications with larger mean error, as depicted in Fig. 4.7.e. The optimization process, therefore, should stop at this step and consider only the previous coefficients. However, a further attempt was introduced by considering 1-POT and 2-POT deviation schemes for the coefficient 30. The response corresponds to employing multiple iterations and the corresponding mean error and cost is shown in Fig. 4.7.f. It takes more computation time, whereas, it achieved about 50% reduction in the number of non-zero terms.
Two important observations at this step: consider multiple optimization iterations instead of a single optimization run. Do not bound the deviation to individual POT value, i.e., 1-POT or 2-POT. However, make it a range from 1-POT to



**(a)**

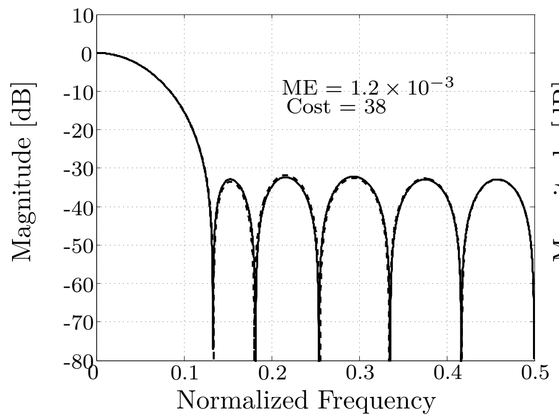


**(b)**

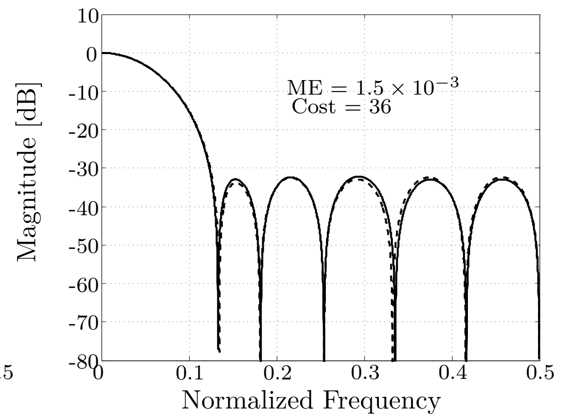


**(c)**

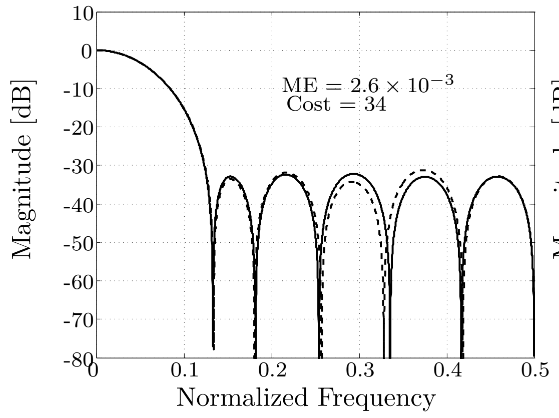
**Figure 4.6.:** Frequency response for 1-POT coefficient deviation (a) 130→128 (b) 30→32 (c) 77→64 for $\hat{h}_k = [143, 130, 106, 77, 49, 30]$.



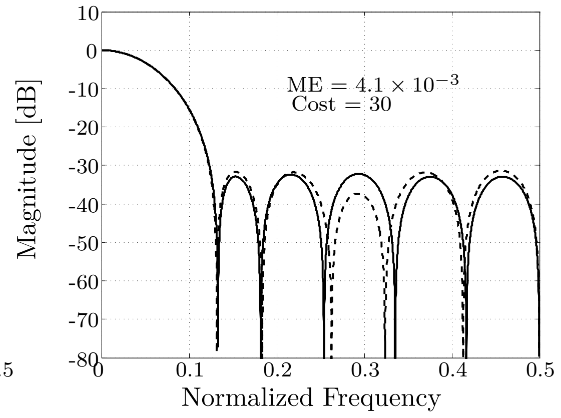
**(a)**



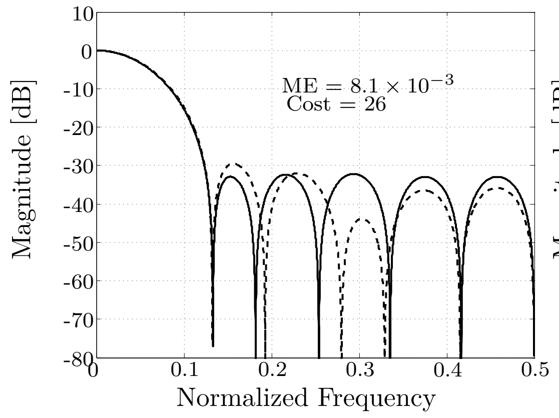
**(b)**



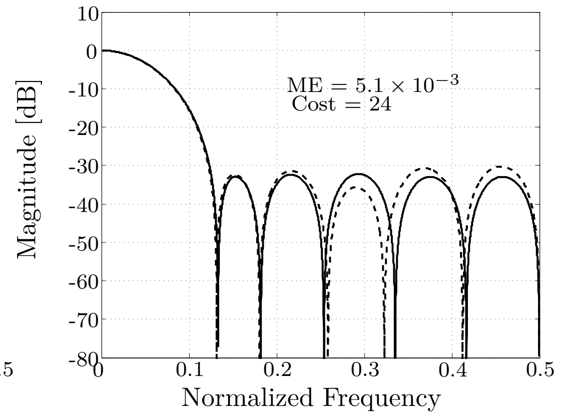
**(c)**



**(d)**



**(e)**



**(f)**

**Figure 4.7.:** Frequency response for 2-POT coefficient deviation (a) 143→144 (b) 49→48 (c) 130→128 (d) 77→76 (e) 30→24 (f) 30→32 for $\hat{h}_k = [143, 130, 106, 77, 49, 30]$.

$(m-1)$-POT, where $m$ is the number of non-zero terms of the coefficient. Considering those observations, the final form of the deviation criterion is presented. The modified criterion is illustrated in Fig. 4.4.c.

## 4.5. Allocation Schemes

Instead of a tree sequence, a presorted allocation scheme is used to allow faster computing time compared to the conventional criteria. The presorted search space is represented by the allocation scheme as given in (4.10)

$$\text{Aloc}[\hat{h}_k] = \{\hat{h}_{k_1}, \hat{h}_{k_2}, \ldots \hat{h}_{k_M}\} \tag{4.10}$$

Conventionally, coefficient sensitivity ($S_n$) [47], [63] is the only available criterion in literature as an allocation scheme. The allocation scheme based on Sensitivity is given by (4.11)

$$\begin{gathered} S_n = \tfrac{1}{N_{FFT}} \sqrt{\sum_{i=1}^{N_{FFT}} [A_n(\omega_i) - A'_n(\omega_i)]^2} \\ S_n(\hat{h}_{k_1}) \leq S_n(\hat{h}_{k_2}) \leq \ldots \leq S_n(\hat{h}_{k_M}) \end{gathered} \tag{4.11}$$

where $A_n(\omega)$ and $A'_n(\omega)$ are the frequency responses of the real and the quantized filters, respectively and $N_{FFT}$ is the number of discrete sampling points over the entire frequency response. $S_n$ presents the effect of each coefficient in the filter response. Coefficients with low sensitivity have less influence on the filter response compared to coefficients with higher sensitivity. Therefore, least sensitive coefficients are good candidates for reducing their number of non-zero terms with minimum change in the filter response. The present work presents two allocation schemes, and a hybrid allocation scheme. The hybrid scheme combines the resultant effect for presorting the coefficients. The proposed allocation schemes are presented and discussed in the following subsections.

### 4.5.1. Deviation

The deviation criterion presented previously in section 4.4 is adopted as an allocation scheme for coefficients presorting. For the same filter, different allocation scheme patterns can be generated according to the reference N-POT chosen for the deviation criterion. For N-POT the allocation scheme presents the deviation between each coefficient and the nearest upper or lower N-POT bound. Figure 4.8 shows the different allocation scheme patterns for the same filter (filter given in section 4.4).

$$\begin{gathered} \Delta_{h_k} = \min(U_b - \hat{h}_k, \hat{h}_k - L_b) \\ \Delta_{h_k}(\hat{h}_{k_1}) \leq \Delta_{h_k}(\hat{h}_{k_2}) \leq \cdots \leq \Delta_{h_k}(\hat{h}_{k_M}) \end{gathered} \tag{4.12}$$

where $U_b$ and $L_b$ are the scaled upper and lower bounds, respectively, as depicted by 4.8 and $\hat{h}_k$ is the scaled coefficient.

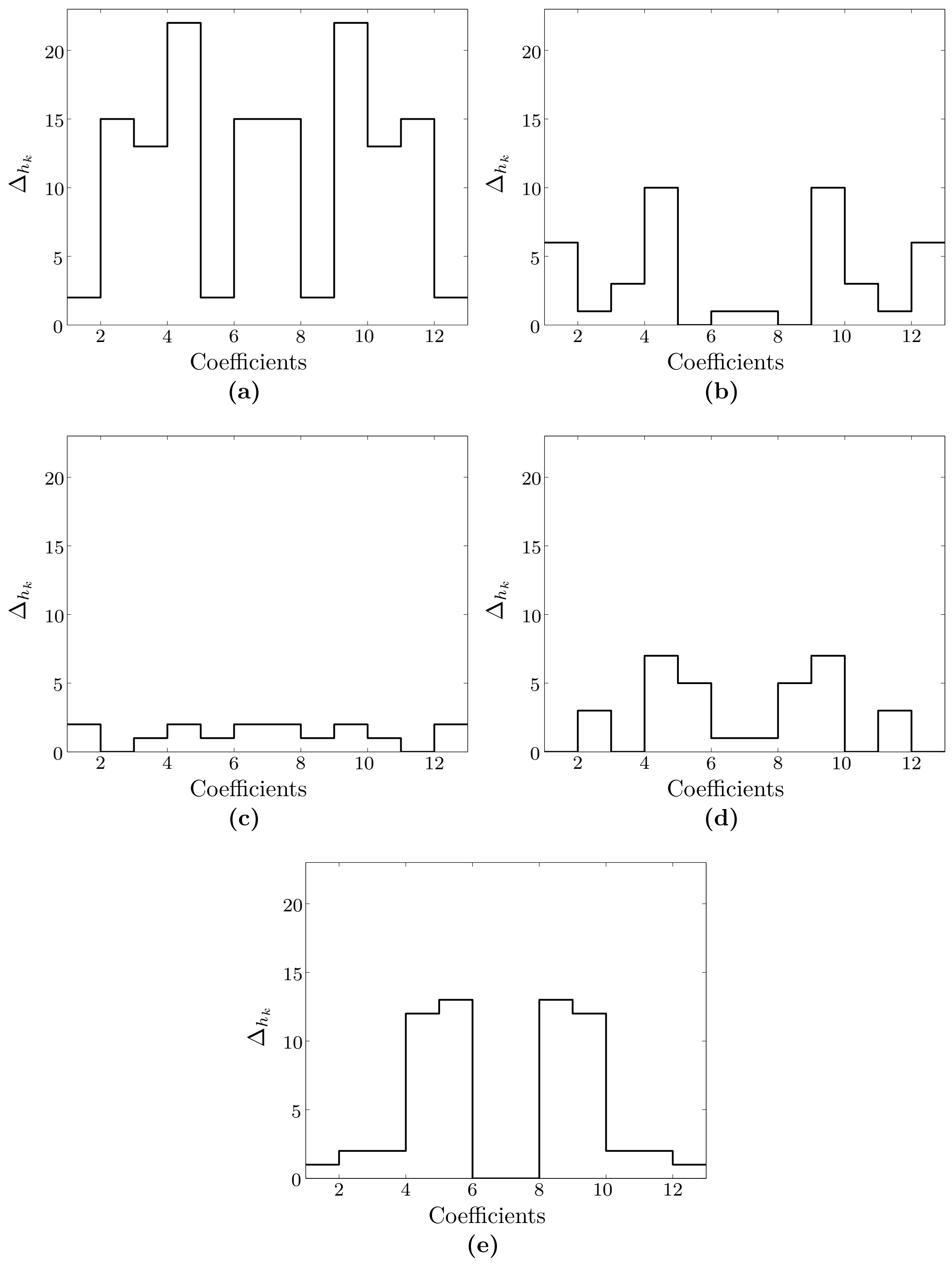


**Figure 4.8.:** Allocation schemes based on multiple N-POT (a) 1-POT (b) 2-POT (c) 3-POT (d) 4-POT (e) 5-POT for benchmark filters given in Table 5.2.

### 4.5.2. Cost

Another allocation scheme is proposed based on the cost of the filter coefficients. The cost is equivalent to the number of non-zero terms per coefficient. The cost function is used as allocation scheme as given by (4.13).

$$\begin{gathered} Cost(\hat{h}_k) = \sum_{i=1}^{Q} \hat{h}_{k,i} \\ Cost(\hat{h}_{k_1}) \geq Cost(\hat{h}_{k_2}) \geq \ldots \geq Cost(\hat{h}_{k_M}) \end{gathered} \tag{4.13}$$

where $Q$ is the quantization bit-width, $\hat{h}_k$ is the scaled filter coefficient, $Cost(\hat{h}_k)$ is the number of none zero terms in the scaled coefficient $\hat{h}_k$, and $\hat{h}_{k,i}$ is the $i^{th}$ bit of the binary representation of $\hat{h}_k$.

### 4.5.3. Hybrid

Further, a hybrid allocation scheme has been developed to consider more than a single criterion for the allocation scheme. However, the coefficients are sorted according to their $S_n$ or $\Delta_{h_k}$ in ascending order, or according to their $Cost$ in descending order. Figure 4.9 shows the different allocation schemes for a single filter. The schemes generated by the resultant of $S_n \times \Delta_{h_k}$ and $S_n \times Cost^{-1}$ are shown in Fig. 4.9 c and d, respectively. If more than one coefficient has the same cost or sensitivity, the first coefficient comes first in the coefficients sequence.
Figure 4.10 summarizes the allocation schemes.

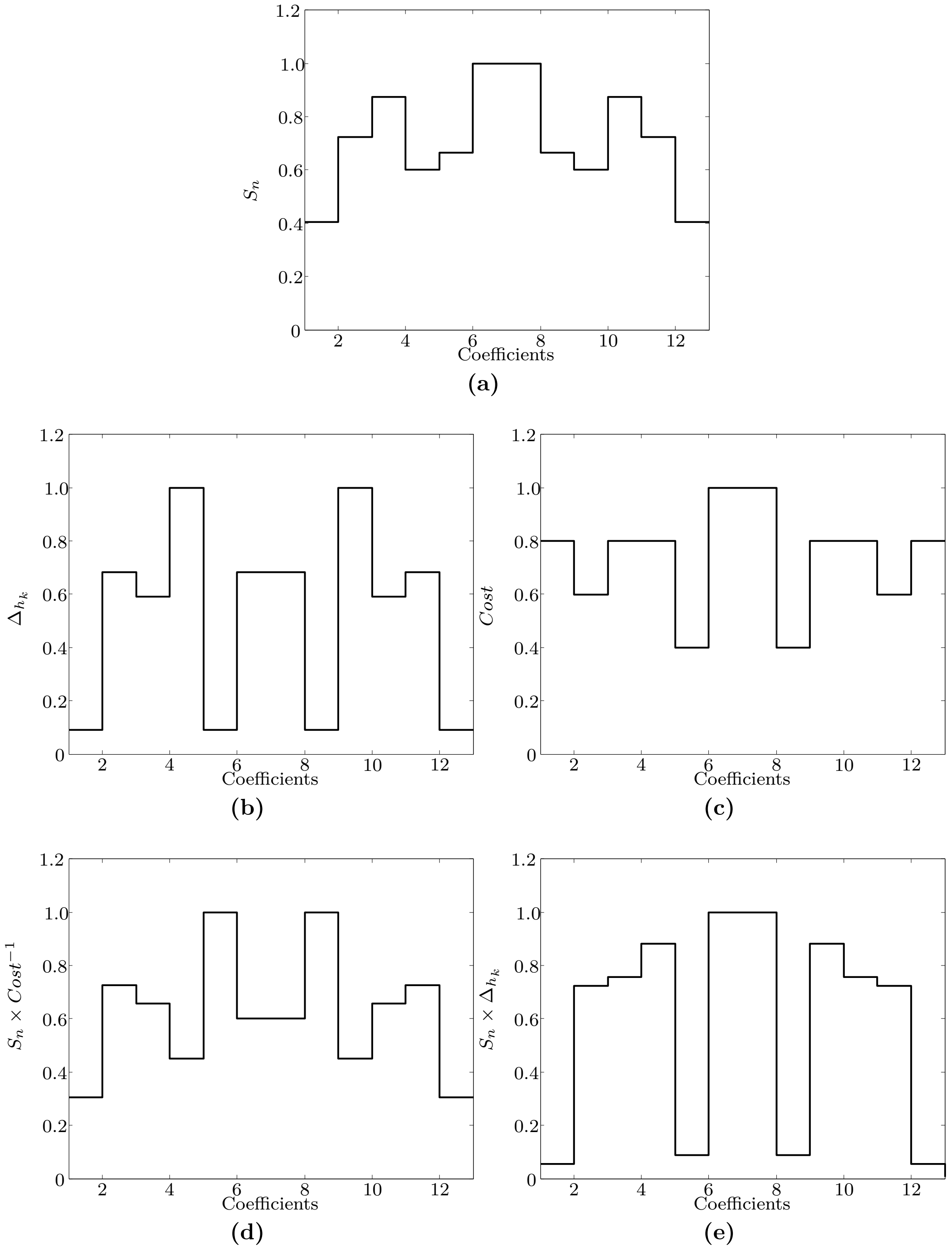


**Figure 4.9.:** Allocation schemes patterns for the same filter according to (a) sensitivity (b) 1-POT deviation (c) cost (d) $S_n \times Cost^{-1}$ (e) $S_n \times \Delta_{h_k}$.

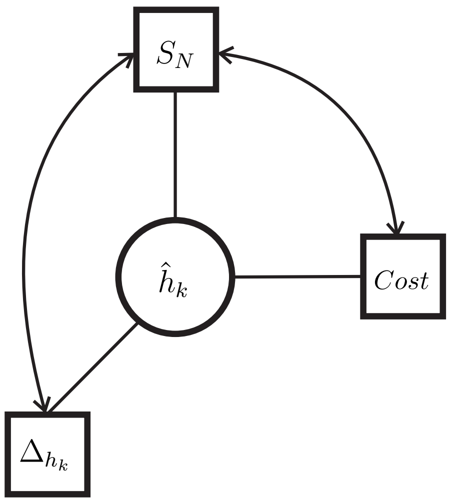


**Figure 4.10.:** Allocation schemes.

# 5. Algorithmic System Level Power Optimization

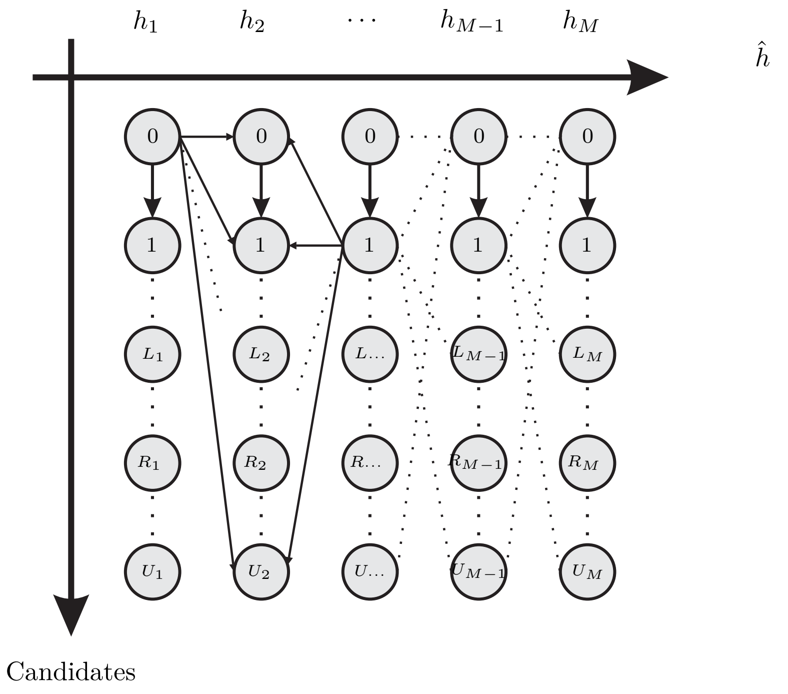
## 5.1. Introduction

The purpose of this chapter is to present the formulation of the FIR filter problem in a mathematical form which can be solved and optimized by an adequate solver. The formulation of the FIR problem mathematically is well covered in the literature. However, the present work proposes: a) the saving in computation time by using the developed allocation scheme by coefficient deviation (as presented in the previous chapter) through polynomial programming and b) a local search algorithm for optimizing (local minimum) a mixed integer linear problem using a heuristic method.
The problem of minimizing the number of non-zero terms in a FIR coefficient set is mathematically modeled in the form of an object and a subject. The object is the problem to be solved, i.e., minimize the number of non-zero terms (**min**(POT)), where POT is the Power-of-Two terms in a binary representation. The subject is the constraints which should be preserved while solving the problem object, i.e., the problem is subject to the passband and stopband ripple ($\delta_{pb}, \delta_{sb}$). The subject inequalities determine the feasible region to solutions that satisfy all the constraints [64]. The modeled problem is thus optimized using either linear or nonlinear programming methods. Various optimization methods have been proposed in the literature such as linear programming, polynomial programming, convex programming, and semidefinite programming. The choice of the optimization method needs to be in accordance with the problem modeling. The linear optimization methods, such as linear programming (LP), are employed, if and only if the object and the subject are linear functions. For the special case where the object and the subject variables are integers, the problem is called integer linear programming (ILP). However, if only some of the object or subject variables are required to be integers, the problem is solved using a mixed integer linear programming (MILP) method. The nonlinear optimization methods, such as polynomial programming (PP), convex programming and semidefinite programming (SDF), are utilized when the object and/or the subject are nonlinear function.

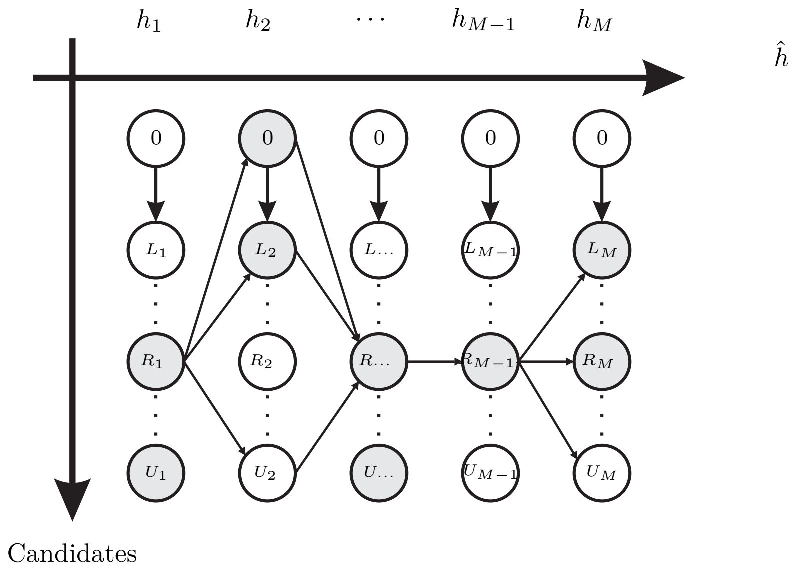
## 5.2. Problem Statement

Power reduction in constant multiplier, known as shift-and-add or multiplierless structures as well, is achieved through the reduction of the number of multiplier adder (MA) and the number of full-adder (NFA) as well. The reduction of MA can be achieved by reducing the number of non-zero terms and eliminating the redundant patterns within the filter coefficients. The number of non-zero terms is reduced through mathematical optimization or CSD representation of the coefficient instead of the binary representation. On the other hand, eliminating the redundant patterns is achieved by considering common sub-expression elimination (CSE) criterion. This chapter presents the proposed methods for reducing the number of non-zero terms through optimization and common sub-expression elimination as well. Then, the following chapter presents the proposed criterion for reducing the number of full-adders.
The FIR optimization problem is well covered in literature. It is commonly described analytically in mathematical models. The author describes the problem graphically as a 2-dimensional (2D) model, in order to have a clear and simple illustration of the problem and make it easier to derive a reliable solution in efficient time. For an FIR filter with $N$ taps and $Q$ quantization bit-width, the optimization procedure goes through $M$ coefficients out of $N$, due to symmetry where $M$ is half the filter length, and $2^Q$ times for each coefficient. This can be shown in Fig. 5.1.a, where the $x$-axis shows the filter coefficients $\{h_1 \cdots h_{M-1}, h_M\}$, and the $y$-axis shows the candidates for each coefficient. This matrix construction represents the search space for the defined optimization problem. In order to find the global optimum of the optimization problem, a systematic solver has to consider all the possibilities (candidates) in the search space. This requires an expensive and massive computation time. This is illustrated by the Gray shaded circles in Fig. 5.1.a. This work aims to reduce the work space candidates, based on qualitative and quantitative analysis in a preprocessing step. If a distribution of candidates as seen in Fig. 5.1.b, where the gray shaded circles have been reduced compared to those in Fig. 5.1.a, the goal is achieved. The upper bound is the nearest highest integer with single non-zero term. The lower bound is the nearest lowest integer with single non-zero term. The problem can be expressed as a 2D problem. The first dimension is expressed by sorting the filter coefficients. This issue is solved by using an allocation scheme. Conventionally, there is only a single known allocation scheme based on coefficient sensitivity. This work proposes two novel allocation schemes. Moreover, it combines the different allocation schemes in a hybrid allocation scheme, which is better than a single allocation scheme. The second dimension of the problem is expressed by the number of candidates for each single coefficient. For a local optimum, this issue can be processed in infinite number of ways by preprocessing, such as; a) bounding the candidates between upper and lower bounds, b) calculating the candidates that can replace the coefficient without violating the constraints. However, in this chapter a novel algorithm is proposed and evaluated to improve the computation time remarkably.

The FIR filter conventions are presented in section 5.3, and a review about the state-of-the-art algorithms is given in section 5.4.



**(a)**



**(b)**

**Figure 5.1.:** Graphical illustration for the problem statement, where $L$ is the lower bound, $U$ is the upper bound, $R$ is a random candidate, $h$ is the filter coefficient, and $\hat{h}_k$ is the scaled filter coefficients.

## 5.3. State-of-the-Art

Methods proposed in the literature consider different optimization aspects, as shown in Table 5.1. The proposed methods aiming for the reduction of the hardware cost have different methodologies such as minimizing the number of SPT/POT terms, normalized peak ripple, ripple optimization, and quantization error minimization, but they are all comparable to each other.
Methods for finding the optimal solution include mixed integer linear programming (MILP) [40], [46], [48], [65], [63], [66], [67], polynomial programming (PP) [68] and semidefinite programming (SDP) [69]. The conventional MILP algorithm has the advantage that it guarantees producing the optimum design (global minimum), but it requires excessive computing resources if the filter length is long. MILP has been used for optimization with different objects and different subjects. Table 5.1 presents several trends for filter optimization using MILP by means of different objects and subjects. Furthermore, several approaches have combined minimizing the number of non-zero terms and the common subexpression elimination [70], [71], [61] in order to achieve effective reduction in the hardware complexity.

**Table 5.1.:** State-of-the-art MILP Optimization Trends

| Object | Subject | Ref. | Remarks |
|---|---|---|---|
| **min**($\delta_s$) | $\delta_p, \delta_s$ | [67] | Variable $Q$ & $N$ |
| **min**($\varepsilon$) | SPT | [62] | Local, bound |
| **min**($\varepsilon$) | SPT & $\varepsilon$ | [48] | Preprocessing |
| **min**(NPR) | SPT | [46] | Local, global |
| **min**(NPR) | $\varepsilon$ | [66] | Bound NPR |
| **min**(NPR) | $\delta_p, \delta_s$ | [70] | CSE, bound |
| **min**(NPR) | SPT | [71] | CSE |
| **min**(NPR) | SPT & $\delta_p, \delta_s$ | [61] | CSE, bound |
| **min**(SPT) | NPR | [47] | Local, tree |
| **min**(SPT) | $\varepsilon$ | [65] | bound |
| **min**(SPT) | $N, Q$ | [40] | Preprocessing, bound |
| **min**(SPT,$Q$) | $\delta_p, \delta_s$ & NPR | [60] | CSE, bound |
| **min**(POT) | $\delta_p, \delta_s$ | Proposed | Local, bound |

$\delta_{pb}$ is the passband ripple, $\delta_{sb}$ is the stopband ripple, $N$ is the filter order,
$Q$ is the coefficient quantization bit-width, $\varepsilon$ is the quantization error,
NPR is the normalized peak ripple, and
SPT|POT is the number of none zero terms.

Several former works propose methods for finding the optimal solution that is limited to smaller filter orders due to their excessive run-time. Therefore, local search methods (also known as sub-optimal methods) are presented in [47], [72], [73], [74]. Furthermore, heuristic algorithms are presented in [60] and [62]. They offer fast run-time ($R_T$) providing most of the time sub-optimal (local optimum) results. A least mean square (LMS) discrete space optimization procedure was developed

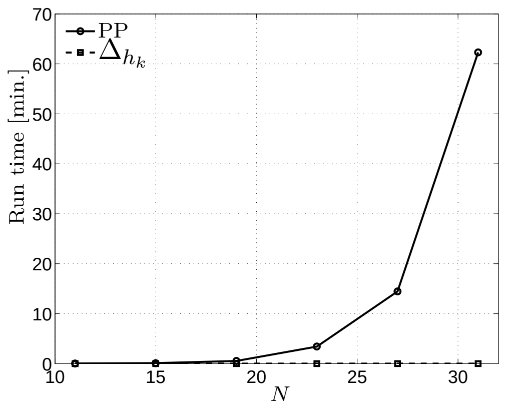
in [73], which is suitable for filter orders up to 90. The run-time for the LMS problem is $N^3$ [73]. A number of these local search methodologies have been developed with an object to minimize the SPT terms subject to normalized peak ripple (NPR) [72], [47].
Approaches presented in [60], [70], [71], and [61] integrate the CSE into the problem formulation. The preprocessing is involved to reduce the number of subproblems to limit the feasible region in advance, as proposed in [40] and [48]. The approaches given in [46], [47], [62] and this work proposes a local search method.
The FIR filter optimization problem has also been formulated as a non-integer optimization problem and solved using polynomial programming (PP) [68], semidefinite programming (SDP) [69], and convex relaxation [75]. Polynomial programming (PP) deals with a class of optimization problems where both the object function and the subject functions are multivariate polynomials [68]. The non-integer problem in [68], [69], and [75] is formulated with an object to minimize the error in the frequency response subject to SPT terms of the discrete filter coefficients. PP reaches globally optimal results up to a filter order of 30 in acceptable time, but the solution for larger filter orders are sub-optimal [68].
In [40] the optimization problem is modeled as mixed integer problem with the object to minimize the number of SPT terms ($\mathbf{min}$(SPT)) subject to the hardware specifications (filter order $N$ and word-length $Q$). In [66] the objective was to minimize the ripple ($\mathbf{min}(\delta_p, \delta_s)$) subject to the number of SPT terms, and in [67] subject to hardware specifications. When the object is to minimize the relative attenuation between the pass and stop bands (NPR), this gives some flexibility in the optimization problem [66]. The optimization algorithm proposed in [60] is performed in two steps. First, find all filter coefficient values (which sustain the filter specification) with an infinite precision. Then, bound each coefficient by the largest and smallest values and scale them to limited bit-width, in order to speed the computing time by limiting the search space. Second, perform common sub-expression elimination. This algorithm is relatively fast but has a drawback for high order filters.
Several non-commercial and commercial solvers are available for solving such optimization problems. YALMIP, [1] a free non-commercial MATLAB toolbox given in [76] offers a modeling language for several optimization problems relying on various external solvers. Other alternative non-commercial MATLAB toolboxes are SeDuMi [2] [77], and GLPK [3]. GloptiPoly [5] [78] is a non-commercial MATLAB toolbox for polynomial programming, and it invokes SeDuMi [77]. CVX [6] is a MATLAB-based modeling system for convex optimization. Further, SCIP [7] is a recent non-commercial solver, though, not compatible with MATLAB. CPLEX [4]

[1] [Online] http://yalmip.org/
[2] [Online] http://sedumi.ie.lehigh.edu/
[3] [Online] http://gnu.org/software/glpk/glpk.html
[5] [Online] http://homepages.laas.fr/henrion/software/gloptipoly3/
[6] [Online] http://cvxr.com/cvx/
[7] [Online] http://scip.zib.de/
[4] [Online] http://www-01.ibm.com/software/websphere/products/optimization/

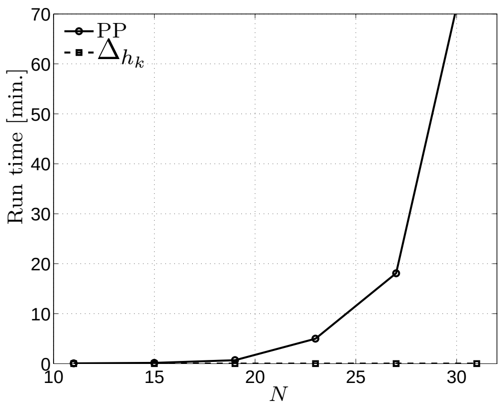
is a commercial solver. Benchmark filters have been adopted from FIRsuite [8] [79] to ease the comparison of different fixed coefficient FIR filter implementations.

## 5.4. Polynomial Programing

The FIR optimization problem can be defined as a quadratic subject by minimizing the error in frequency response. Hence, it is optimized by polynomial programming (PP) solvers. The detailed mathematical derivation is given in Appendix A.
In order to validate the performance of the proposed coefficient deviation scheme, a comparison with PP systematic solvers was carried out. The formulation of PP problem was published in [68] and solved using the toolboxes presented in [77] and [78]. The following simulations were carried out for an FIR filter with passband frequency of 0.2, stopband frequency of 0.25 and filter orders between $[11 - 31]$ with a step of 4. The passband and stopband frequencies are normalized. The 2-POT and 4-POT search spaces were considered by the simulation.
The results depicted for run-time and gain on Figs. 5.2 and 5.3, respectively. In the one hand, the proposed allocation scheme outperforms the PP solver, as shown in Fig. 5.2.a. On the other hand, the PP solver achieves higher gain as given in Fig. 5.3.
Hence, the coefficient deviation scheme shows a considerable saving in computation time compared to other systematic solver. However, it has much lower gain in reducing the total number of non-zero elements. Therefore, the following sections will illustrate the proposed criteria to enhance this drawback.



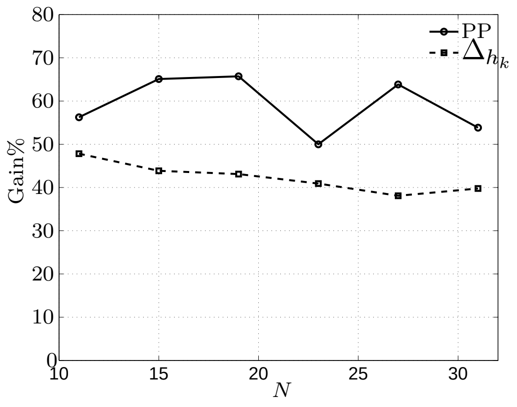
(a)



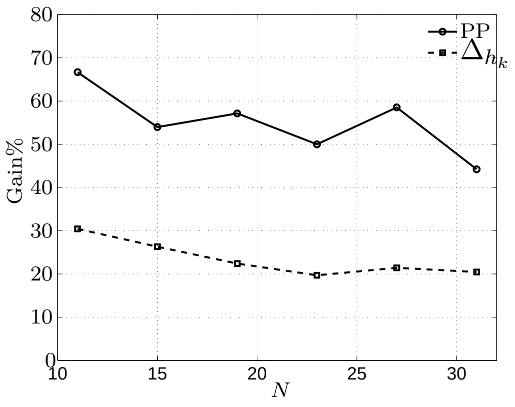
(b)

**Figure 5.2.:** Optimization run-time using PP versus $\Delta_{h_k}$ at (a) 2-POT and (b) 4-POT (filter specs $f_{pb} = 0.2$, $f_{sb} = 0.25$ and $N = [11 : +4 : 31]$).

[8] [Online] http://firsuite.net/



**(a)**



**(b)**

**Figure 5.3.:** Optimization gain using PP versus $\Delta_{h_k}$ at (a) 2-POT and (b) 4-POT (filter specs $f_{pb} = 0.2$, $f_{sb} = 0.25$ and $N = [11 : +4 : 31]$).

## 5.5. Mixed Integer Linear Programming

The conventional mixed integer linear programming (MILP) finds the global minimum for the FIR filter problem, but requires excessive computation time. For each single discrete filter coefficient, there are $[0 : 2^{m+1}]$ candidates, where $m$ is the number of POT terms in the discrete coefficient. Consequently, this generates a dense searching space formed in a nested tree structure searching space, as shown in Fig. 5.4. It checks all possibilities in the tree to guarantee the globally optimal solution. Therefore, an excessive run time is required during the optimization process. Figure 5.5 shows the run-time of the MILP algorithm compared to other algorithms [72].

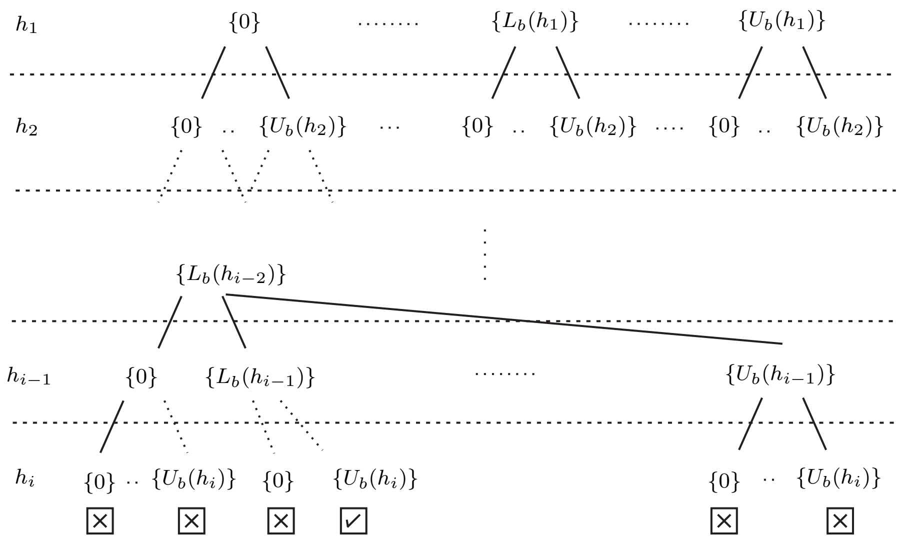


**Figure 5.4.:** Search tree for conventional MILP, where $L$ is the lower bound, $U$ is the upper bound, and $h$ is the filter coefficient.

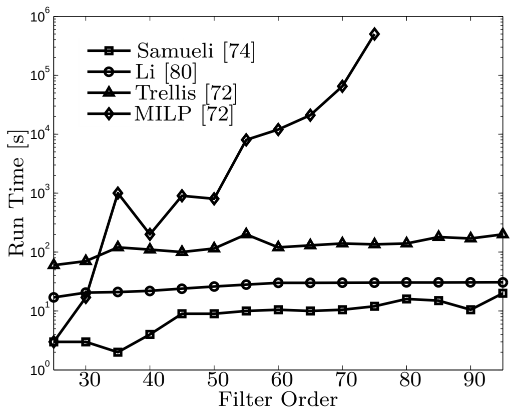


**Figure 5.5.:** MILP runtime using Samueli, Li, Trellis, and MILP algorithms [72].

### 5.5.1. Problem Formulation

The FIR optimization problem is modeled with an integer linear object and a non-integer linear subject; for a given set of filter specifications, such as passband edge $\omega_{pb}$, stopband edge $\omega_{sb}$, passband ripple $\delta_{pb}$, stopband ripple $\delta_{sb}$ and the filter length $N$. The problem is formulated as a mixed integer linear problem. On one hand, it can be solved by employing a MILP solver as a systematic criterion. On the other hand, it can be solved using a heuristic criterion. The optimization problem object function is shown in (5.1) and the subject function in (5.2). Even though, the problem is modeled for lowpass filters, it could be extended to bandpass filters as well.

$$\textbf{minimize} \sum_{k=1}^{M} \text{Cost}(\hat{h}_k) \tag{5.1}$$

**subject to**:

$$\begin{aligned}
\sum_{k=1}^{M} \hat{h}_k c(k,\omega T) &\leq 2^{Q-1} \times (1+\delta_{pb}), \omega T \in [0,\omega_{pb}] \\
-\sum_{k=1}^{M} \hat{h}_k c(k,\omega T) &\leq 2^{Q-1} \times (\delta_{pb}-1), \omega T \in [0,\omega_{pb}] \\
\sum_{k=1}^{M} \hat{h}_k c(k,\omega T) &\leq 2^{Q-1} \times \delta_{sb}, \omega T \in [\omega_{sb},\pi] \\
-\sum_{k=1}^{M} \hat{h}_k c(k,\omega T) &\leq 2^{Q-1} \times \delta_{sb}, \omega T \in [\omega_{sb},\pi]
\end{aligned} \tag{5.2}$$

where $\hat{h}_k$ is the scaled filter coefficients from the quantized filter coefficients $h_k$, and is defined by

$$\hat{h}_k = \begin{cases} \left\lfloor h_k \times 2^{Q-1} + 0.5 \right\rfloor, h_k \geq 0 \\ \\ \left\lceil h_k \times 2^{Q-1} - 0.5 \right\rceil, h_k < 0 \end{cases}$$

and for symmetric impulse response and odd filter length $N$ [40]

$$\begin{aligned}
c(k,\omega T) =& x_k \cos(\omega T[k-1]) \\
x_k =& \begin{cases} 1, k=1 \\ 2, k=2,3,\cdots,M \end{cases} \\
M =& N/2+1
\end{aligned}$$

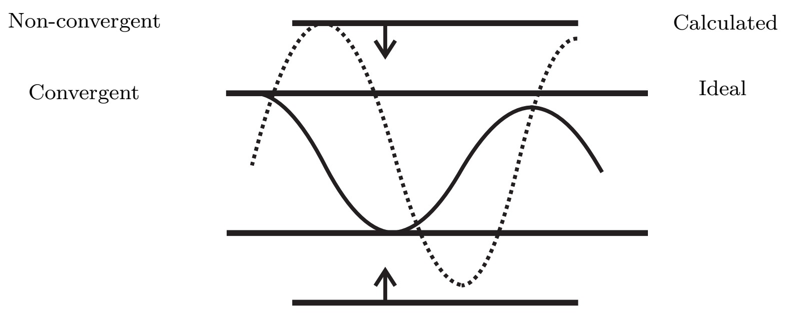


**Figure 5.6.:** Filter ripples converge from calculated (quantized) toward ideal, or nonconverge at calculated.

while for symmetric impulse response and even filter length $N$ [40]

$$\begin{aligned} c(k,\omega T) =& 2\cos(\omega T[k-1]) \\ k =& 1,2,3,\cdots,M \\ M =& (N+1)/2 \end{aligned}$$

where the filter length is equivalent to the filter order plus one.
An independent heuristic solver using Matlab, named POTMILP, has been developed for manipulating the formulated problem and the proposed algorithm.
The number of POT terms is minimized subject to the filter specifications. The subject is formulated for discrete values of $\omega T$ by selecting a number of values in [0,$\omega_{pb}$] and [$\omega_{sb}$,$\pi$] to obtain a grid of values at each constraint. As the specifications are checked only at these points, it is necessary to verify the resulting frequency response with a much finer grid to ensure that a valid coefficient set is obtained [40]. The solution to this optimization problem yields a coefficient set with minimal POT terms for which the ripple in the passband and stopband remain within the allowed bounds. The bounds of the ripple are defined by the user as a hard constraint. If there is a discrepancy between the user defined constraints and the calculated constraints, the developed solver converges towards the minimum ripple as shown in Fig. 5.6. However, the nonconvergent criterion offers more flexibility in the subject constraints, which accordingly results in a higher reduction in non-zero terms.

### 5.5.2. Evolution

The function element ($E^i$) is used to create a set of candidate integers for each scaled coefficient in ascending order constrained by the lower and upper bounds for each scaled coefficient for the evolution ($Evl_i$) space in monotonic steps, with maximum cost=$m-1$, where $m$ is the cost for a discrete coefficient represented in

$Q$-bits. Hence, $E^i$ is a set of values with $i$ POT terms. The $E^i$, is given by (5.3)

$$E^i(\hat{h}_k) = \{n \times b_k \,| n < \left|L_b(\hat{h}_k)\right|, Cost(n) = i\} \tag{5.3}$$

where $b_k = 1$ if $h_k \geq 0$ and $b_k = -1$ if $h_k < 0$. As,

$$E^0(\hat{h}_k) = \{0\}$$
$$E^1(\hat{h}_k) = \{2^0, \ldots, 2^{L-1}; L = \left\lfloor \log_2(\left|\hat{h}_k\right|)\right\rfloor\}$$

are defined as the set of elements with N-POT terms, where $N = m - 1$ and $m$ is the maximum cost for a discrete coefficient represented in $Q$-bits. Thus, for coefficient $\hat{h} = 13$ with $m = 3$ and $Q = 4$, there is $E^0(13) = \{0\}$, $E^1(13) = \{1, 2, 4, 8\}$, and $E^2(13) = \{(1+2), (1+4), (1+8), (2+4), (2+8), (4+8)\}$.
However, the evolution ($Evl_i$) space, as depicted in (5.4)

$$Evl(\hat{h}_{k_i}) = \left\{Evl_0(\hat{h}_{k_i}), Evl_1(\hat{h}_{k_i}), \cdots, Evl_{m-1}(\hat{h}_{k_i})\right\} \tag{5.4}$$

where

$$m = Cost(\hat{h}_{k_i})$$
$$Evl_0(\hat{h}_{k_i}) = \{0\}$$
$$Evl_1(\hat{h}_{k_i}) = \{L_b(\hat{h}_{k_i}), U_b(\hat{h}_{k_i})\}$$
$$Evl_i(\hat{h}_{k_i}) = \{L_b(\hat{h}_{k_i}) + E^{i-1}(\hat{h}_{k_i})\}$$

holds the candidates with equal cost for each discrete coefficient. Thus, for coefficient $\hat{h} = 13$ with $m = 3$ and $Q = 4$, there is $Evl_0(13) = \{0\}$, $Evl_1(13) = \{8, 16\}$, $Evl_2(13) = \{L_b(13) + E^1(13)\} = \{(8+1), (8+2), (8+4)\cdots\}$.

### 5.5.3. Proposed POTx Algorithm

The proposed POTx algorithm as published in [2] is illustrated in the flowchart shown in Fig. 5.7. The basic flow of the algorithm is as follows:

- Inputs to the POTx algorithm are the floating point precision filter coefficients $h_k$, and the quantization bit-width $Q$.
- The input floating point coefficients are quantized. Subsequently, coefficients are scaled. Later on, a sorted search space is created following the concept in section 4.5.
- The algorithm checks all the coefficient sets by
  - defining the maximum cost $m$
  - create a set of $E^i$ and $Evl_i$ candidates as explained in section 5.5.2
  - optimize/round preserving the ripple constraints

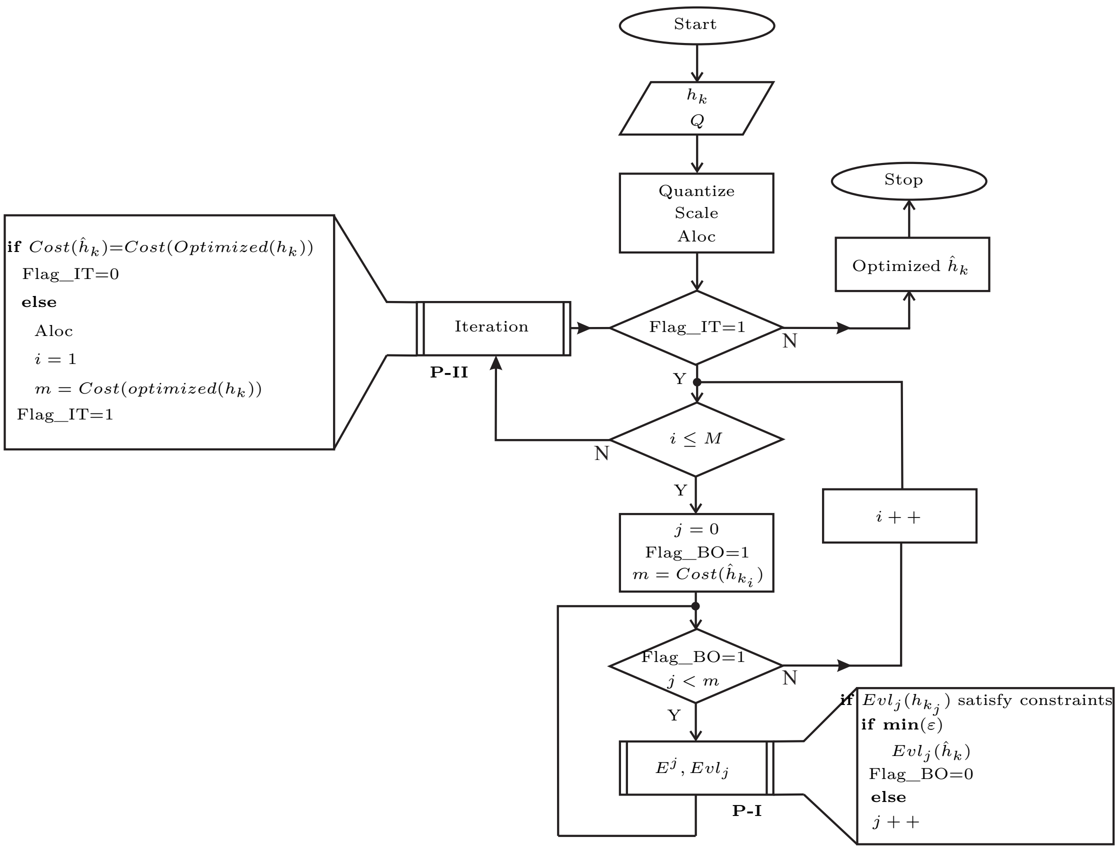


**Figure 5.7.:** POTx algorithm flowchart.

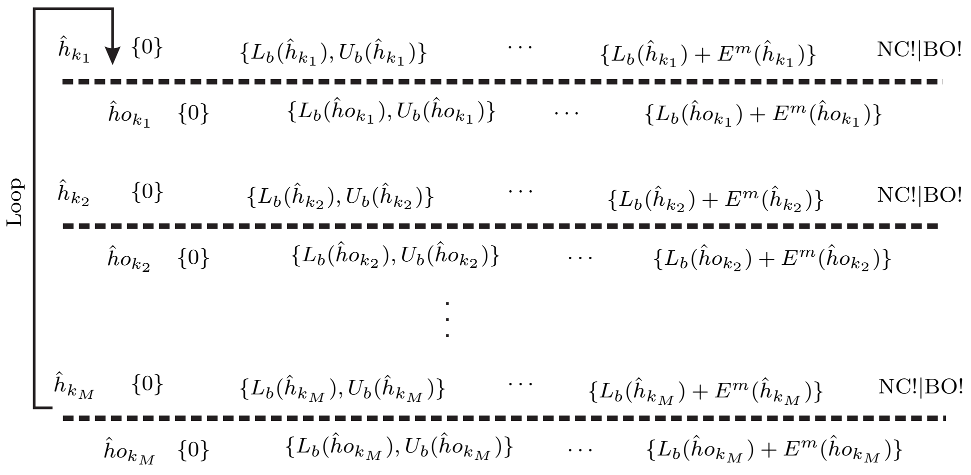

Loop
$\hat{h}_{k_1}$ $\{0\}$ $\{L_b(\hat{h}_{k_1}), U_b(\hat{h}_{k_1})\}$ $\cdots$ $\{L_b(\hat{h}_{k_1}) + E^m(\hat{h}_{k_1})\}$ NC!|BO!
$\hat{h}o_{k_1}$ $\{0\}$ $\{L_b(\hat{h}o_{k_1}), U_b(\hat{h}o_{k_1})\}$ $\cdots$ $\{L_b(\hat{h}o_{k_1}) + E^m(\hat{h}o_{k_1})\}$
$\hat{h}_{k_2}$ $\{0\}$ $\{L_b(\hat{h}_{k_2}), U_b(\hat{h}_{k_2})\}$ $\cdots$ $\{L_b(\hat{h}_{k_2}) + E^m(\hat{h}_{k_2})\}$ NC!|BO!
$\hat{h}o_{k_2}$ $\{0\}$ $\{L_b(\hat{h}o_{k_2}), U_b(\hat{h}o_{k_2})\}$ $\cdots$ $\{L_b(\hat{h}o_{k_2}) + E^m(\hat{h}o_{k_2})\}$
$\vdots$
$\hat{h}_{k_M}$ $\{0\}$ $\{L_b(\hat{h}_{k_M}), U_b(\hat{h}_{k_M})\}$ $\cdots$ $\{L_b(\hat{h}_{k_M}) + E^m(\hat{h}_{k_M})\}$ NC!|BO!
$\hat{h}o_{k_M}$ $\{0\}$ $\{L_b(\hat{h}o_{k_M}), U_b(\hat{h}o_{k_M})\}$ $\cdots$ $\{L_b(\hat{h}o_{k_M}) + E^m(\hat{h}o_{k_M})\}$

**Figure 5.8.:** Proposed POTx algorithm searching space structure, where $L$ is the lower bound, $U$ is the upper bound, $\hat{h}_k$ is the scaled filter coefficients, $E^m$ is the element function, NC! is the break-off state, and BO! is the break-off state.

- Loop through the optimized coefficients if permitted (subject is not violated)

The algorithm consists of two processes. The P-I process, inspects the elements of the evolution sequence $Evl_j$ for a particular coefficient until approaching the break-off (BO!) state or the no-change (NC!) state. The BO! state implies abort at this candidate and does not go through the rest of the $Evl_j$ candidates. While the NC! state implies keeping the coefficient without replacing it. In this process, P-I, the filter constraints in the passband and stopband are probed for each inspected $Evl_j$ vector elements. The $Evl_j$ vector holds the candidates for each coefficient in ascending order, from lowest cost to a cost which is less than the coefficient cost. So, if the optimized coefficient cost is $m$=5, the maximum cost for the candidates in the $Evl_j$ vector is $m$=4. If the subject constraints are preserved for the optimized coefficient the algorithm will reach the break-off (BO!) state. Contrary, the algorithm will reach a no-change (NC!) state. Introducing the BO! state, accelerates the algorithm compared to the conventional MILP algorithm because of its iteration criterion in monotonic steps as shown in Fig. 5.8. Moreover, it offers minimum cost candidate for the optimized coefficient, since it prevents the algorithm from inspecting candidates which have higher cost from the $Evl_j$ vector. The P-II process, resets the internal variables and flags to start a new iteration through the pre-optimized coefficients with a new allocation scheme (Aloc) and new maximum cost ($m$). Process P-II, guarantees efficient optimization process with minimum cost candidates for each coefficient by multiple loops through the optimized coefficients as shown in Fig. 5.8 by the feedback arrow designated by '**Loop**'. Figure 5.8 shows the equivalent searching structure for the POTx algorithm. A detailed illustrative example is presented in section 5.8.5. The POTx algorithm is executed as follows:

```
Quantize and scale coefficients
Create the presorted allocation scheme
Iterate or not
    No: Export optimized coefficients
    Yes: cont.
Inspected all coefficients
    No: Go to P-I
    Yes: Go to P-II
```

By using the nonconvergent (NCV) criterion, as shown in Fig. 5.6, more flexibility is introduced in the constraints so that further reduction in the number of non-zero terms is achieved.

There is a set of control attributes which can be used to optimize the algorithm in terms of allocation scheme, optimization iterations and constraints. To clarify the applied options, the nomenclature POTx.y.z is used in the following. The 'x' represents the allocation scheme for presorting the filter coefficients. The 'y' represents the solver thread. The 'z' represents the constraints. As an example, C.LP.NCV uses the allocation scheme based on Cost, with multiple optimization iterations and nonconvergent constraints. SD uses the hybrid allocation scheme $S_n \times \Delta_{h_k}$.

### 5.5.4. Example

This section presents an example to illustrate the proposed algorithm employing the developed solver. A linear-phase low-pass FIR filter with normalized passband and stopband edge frequencies at $f_{pb} = 0.05$ and $f_{sb} = 0.25$, respectively, is considered as a case study. The desired ripple in the passband and the stopband are $\delta_{pb} = 0.0575$ and $A_{sb} = 30$ dB, respectively. The quantization bit-width $Q = 11$ and the filter length $N = 12$. Due to symmetry the optimization is processed on half of the coefficients. The quantized filter coefficients ($h_k$) are:

$$h_k = [0.13935, 0.12651, 0.10363, 0.075543, 0.047644, 0.029569]$$

The scaled coefficients ($\hat{h}_k$), coefficients cost ($Cost[\hat{h}_k]$), and generated allocation scheme according to cost (Aloc$[\hat{h}_k]$) are

$$\begin{aligned} \hat{h}_k &= [143, 130, 106, 77, 49, 30] \\ Cost[\hat{h}_k] &= [5, 2, 4, 4, 3, 4] \\ \text{Aloc}[\hat{h}_k] &= [143, 106, 77, 30, 49, 130] \end{aligned}$$

Consider the coefficient 143. It has 5-POT terms so the solver will generate the $E^i$ and $Evl_i$ functions up to a maximum of 4-POT terms in monotonic sequence, as follow

$$\begin{aligned} &E^0(143) = \{0\} \\ &E^1(143) = \{1, 2, 4, 8, 16, 32, 64\} \\ &E^2(143) = \{(1+2), (1+4) \cdots (2+4), (2+8) \cdots\} \\ &E^3(143) = \{(1+2+4), (1+2+8) \cdots (1+4+8), (1+4+16) \cdots\} \end{aligned}$$

$$\begin{aligned} Evl_0(143) &= \{0\} \\ Evl_1(143) &= \{128, 256\} \\ Evl_2(143) &= \{128 + E^1(143)\} \\ &= \{(128+1), (128+2), (128+4) \cdots\} \\ Evl_3(143) &= \{128 + E^2(143)\} \\ &= \{(128+1+2), (128+1+4) \cdots (128+2+4) \cdots\} \\ Evl_4(143) &= \{128 + E^3(143)\} \\ &= \{(128+1+2+4), (128+1+2+4) \cdots (128+1+4+8) \cdots\} \end{aligned}$$

The monotonicity in coefficient cost is preserved in one evolution step as

$$Evl_2(143) = \{(128+1), (128+2), (128+4) \cdots\}$$

$$\begin{aligned} 129 &= 1000000\underline{1} \\ 130 &= 100000\underline{1}0 \\ 132 &= 10000\underline{1}00 \\ &\vdots \\ 192 &= 1\underline{1}000000 \end{aligned}$$

as described above As an example, if within $Evl_2$ for coefficient 143, both 136 and 144 do not violate the filter constraints, $\mid 143 - 144 \mid = 1$ while $\mid 143 - 136 \mid = 7$ so 144 is chosen.
The $E^i$ and $Evl_i$ functions are generated for each coefficient. The coefficients are arranged vertically according to the cost given by the allocation space, as shown in Fig. 5.9 by arrow 2. The solver starts in horizontal direction as depicted by arrow 1 in Fig. 5.9 for all coefficients. Then the POTMILP solver proceeds as illustrated in Fig. 5.9. The POTMILP solver then checks for each coefficient the candidates, within the evolutions, starting with $Evl_0$, until it reaches a Break-Off (BO!) or No-Change (NC!) state. Considering the first coefficient, $\hat{h}_k = 143$, the solver substitutes $\{0\}$ instead, as an example, then examines the filter response including the new coefficient value. If the filter constraints are not violated, the coefficient is replaced, and the Break-Off (BO!) state is reached. If the constraints are violated, the POTMILP iterates and continue with the next evolution vector $\{128, 256\}$. The solver has to check a complete $Evl_i$ sequence in one run. If in a $Evl_i$ sequence more than one candidate satisfies the constraints, the value with the minimum difference from the original value is chosen. If none of the coefficient candidates satisfies the constraints, the coefficient is left unchanged, implying the NC! state. If one of the evolutions satisfies the constraints, the tool will abort because there is no need to continue, since the following evolution sequences will have more POT terms which means higher cost. In the complete coefficient set, the first optimization loop results in the $\hat{h}_{k_{opti1}}$ coefficients with a cost reduction from 22-POTs to 13-POTs. This is the cost for just half the coefficients.

$$\begin{aligned}
\hat{h}_{k_{opti1}} &= [144, 128, 104, 80, 48, 28] \\
Cost[\hat{h}_{k_{opti1}}] &= [2, 1, 3, 2, 2, 3] \\
\text{Aloc}[\hat{h}_{k_{opti1}}] &= [104, 28, 144, 80, 48, 128]
\end{aligned}$$

The second optimization iteration is marked in Gray as shown in Fig. 5.9. The second optimization iteration has the presorted allocation scheme defined by Aloc$[\hat{h}_{k_{opti1}}]$. The final result of the optimization process is depicted by the $\hat{h}_{k_{opti2}}$ vector with an overall cost of 12-POTs.

$$\begin{aligned}
\hat{h}_{k_{opti2}} &= [144, 128, 104, 80, 48, 24] \\
Cost[\hat{h}_{k_{opti2}}] &= [2, 1, 3, 2, 2, 2]
\end{aligned}$$

Through using the iterating loops, presented by P-II in Fig. 5.7, the proposed algorithm offers adequate saving in the cost. Figure 5.10 shows the frequency responses of the quantized filter, the intermediate optimized filter, and the final optimized filter. The intermediate optimized filter ($\hat{h}_{k_{opti1}}$) experience a mean square error of $40 \times 10^{-6}$ in its response. Whereas, the final optimized filter ($\hat{h}_{k_{opti2}}$) experience a mean square error of $100 \times 10^{-6}$. Thus, an overall saving by about 50% in the cost is achieved within a run-time of few seconds, for a tolerable penalty in the filter response.

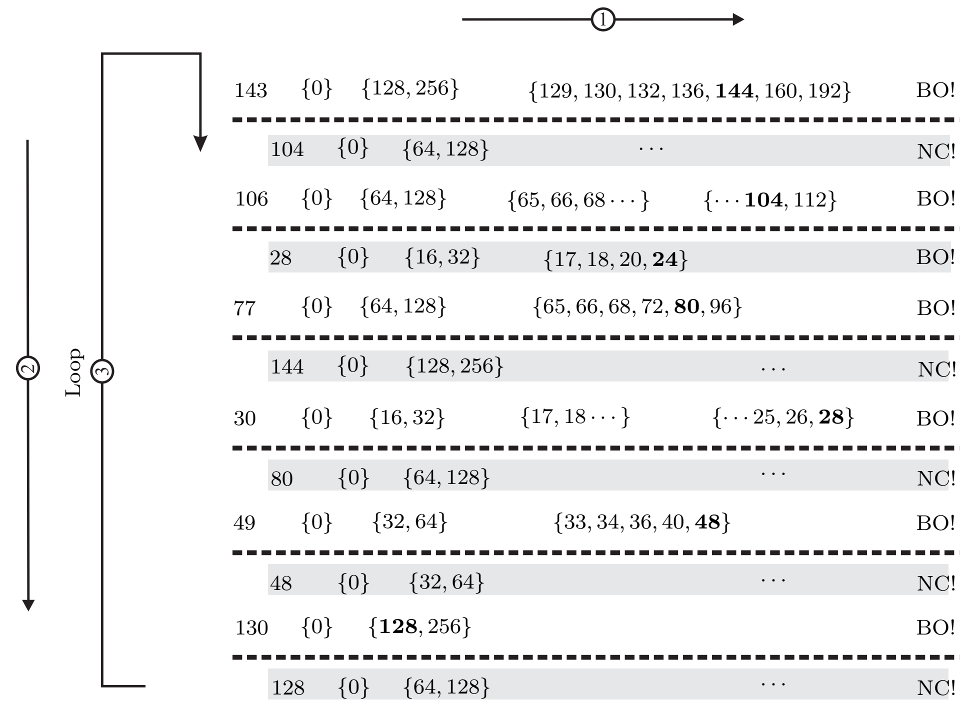

1
2
Loop 3
143 {0} {128, 256} {129, 130, 132, 136, **144**, 160, 192} BO!
104 {0} {64, 128} ··· NC!
106 {0} {64, 128} {65, 66, 68 ···} {··· **104**, 112} BO!
28 {0} {16, 32} {17, 18, 20, **24**} BO!
77 {0} {64, 128} {65, 66, 68, 72, **80**, 96} BO!
144 {0} {128, 256} ··· NC!
30 {0} {16, 32} {17, 18 ···} {··· 25, 26, **28**} BO!
80 {0} {64, 128} ··· NC!
49 {0} {32, 64} {33, 34, 36, 40, **48**} BO!
48 {0} {32, 64} ··· NC!
130 {0} {**128**, 256} BO!
128 {0} {64, 128} ··· NC!

**Figure 5.9.:** Execution of proposed algorithm with multiple optimization iterations.

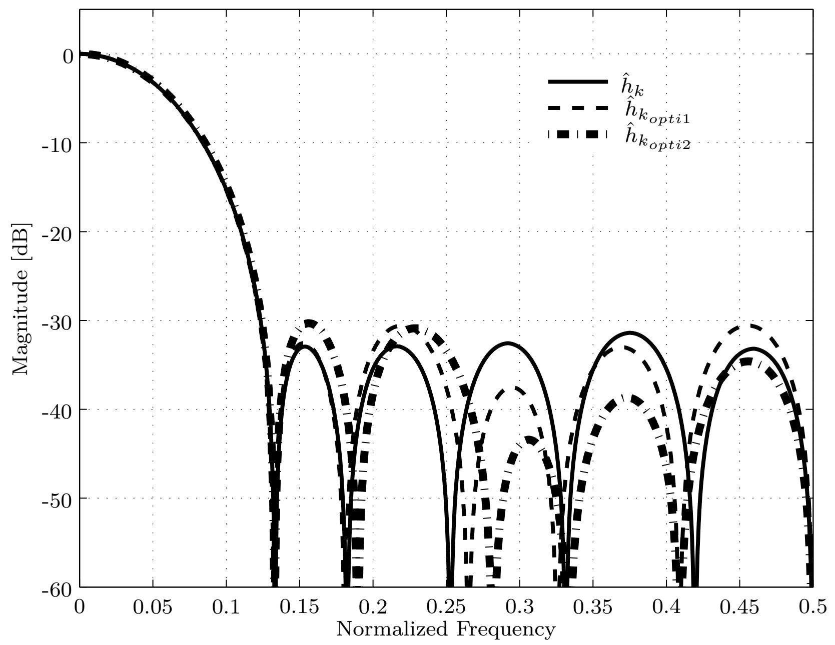


**Figure 5.10.:** Frequency response of filter shown in section 5.5.4.

### 5.5.5. Performance Evaluation and Results

There are two main factors that determine the performance of the proposed POTx algorithm: run-time ($R_T$) and gain (Gain %). The gain represents the percentage of reduction in the number of non-zero terms in the optimized coefficient set compared to the scaled coefficient set. Therefore, higher gain is equivalent to less non-zero terms. Several benchmark filters are optimized using the POTx algorithm and the POTMILP solver. The specifications of the benchmark filters are given in Table 5.2 where all the frequencies are normalized. The implemented algorithm and the developed solver are executed on a 1.6-GHz Pentium processor and 1-GB RAM. The number of POT terms per coefficient was not constrained. Inspired by the method in [42], the filter is designed using the Remez algorithm with the constraints for the stopband attenuation tightened by 3-dB. The value of 3-dB was chosen according to quantitative preliminary simulations. The approach of designing the filter with tighter constraints in the stopband and then optimizing the same filter with the given specifications results in extended flexibility for the optimization. As an example, if the desired stopband attenuation ($A_{sb}$) is -50 dB, the filter is designed for $A_{sb} = -53$ dB. Nonetheless, it is optimized for $A_{sb} = -50$ dB.

**Table 5.2.:** Benchmark filters FIR filters

| Filter | $\delta_{pb}$ | $f_{pb}$ | $\delta_{sb}$ | $f_{sb}$ | $N$ |
|---|---|---|---|---|---|
| A | 0.0100 | 0.0625 | 0.0010 | 0.1125 | 59 |
| B | 0.0100 | 0.1 | 0.0100 | 0.12 | 105 |
| C | 0.0050 | 0.0625 | 0.0050 | 0.07 | 325 |
| L2 | 0.0288 | 0.1 | 0.0010 | 0.14 | 63 |
| S2 | 0.0115 | 0.021 | 0.0010 | 0.07 | 60 |
| LP1 | 0.1000 | 0.3 | 0.0100 | 0.35 | 50 |
| N1 | 0.0005 | 0.2 | 0.0001 | 0.3 | 67 |

The results for the evaluation of the POTx algorithm are shown in Table 5.3. The coefficient deviation allocation scheme results in a lowest reduction in the non-zero terms and minimum run-time for sub-optimal algorithms (single run without iteration) compared to the other POTx settings. The hybrid allocation schemes, on the other hand, achieve the highest reductions in non-zero terms. Adopting the nonconvergent approach generally results in the highest reductions in non-zero terms, because it offers more flexibility in the subject. Figure 5.11 summarizes the results presented in Table 5.3. The ratio of the minimum gain (Gain$^-$) to the maximum gain (Gain$^+$) varies from a factor of 1.7 to a factor of 3.12, as shown in Fig. 5.11. POTSC.LP.NCV using a hybrid allocation scheme, in the majority of the benchmark filters exhibits one of the highest reductions in POT terms and can therefore be considered the best parameter set. However, POTD.NL.CV offers the lowest run-time at acceptable optimization results because it has no iterating optimization loops, and can thus be considered a good trade-off between run-time

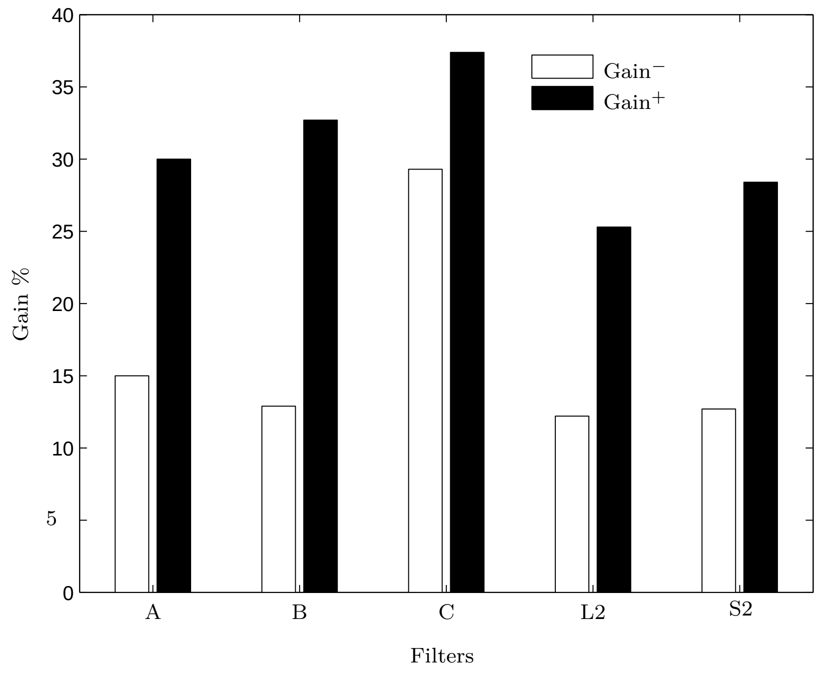


**Figure 5.11.:** Summary of results given in Table 5.3, where Gain$^-$ is the minimum gain and Gain$^+$ is the maximum gain.

and optimization quality. The results reveal that, further reduction in the number of non-zero terms attained by employing multiple optimization iterations with the nonconvergent criterion by the proposed algorithm.
The coefficient set derived using the proposed algorithm is compared to the results of the Remez algorithm (RMZ), Aktan's algorithm [65] (FIRGAM), the Trellis algorithm [72] (TRE), Lim's algorithm [73] (LIM), Samueli's algorithm [74] (SAM), the Li's algorithm [80] (LI), Yao's algorithm (PMILP), Shi's algorithm [81] (SHI), and the MILP. For the Remez algorithm, the Matlab Remez function is used to satisfy the filter specifications, followed by a quantization which allows fulfilling the required specifications. It is used as reference point to quantify the gain achieved using various algorithms. The results for the rest of the algorithms are taken directly from [65]. Table 5.4 summarizes the performance of the proposed algorithm compared to state-of-the-art algorithms. It has to be noted that, the results using the NCV attribute is given in Table 5.4. The number of SPT terms presented in Table 5.4 is obtained by converting the generated filter coefficients from the POTx algorithm using the conversion algorithm described previously.
Afterwards, common sub-expression elimination (CSE) is employed to reduce the number of multiplier adders (MA). The CSE algorithm is described in more detail in the following section. The results of associating the CSE with the optimized filter coefficient set obtained from the convergent POTx algorithm is given in Table 5.11. Alone, the results from the CV attributes are presented to assure consistent comparison with literature algorithm.

**Table 5.3.:** Proposed algorithm evaluation

| POTx | | | SPT | | | | | Time (s) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Allocation | Iteration | Constraints | A | B | C | L2 | S2 | A | B | C | L2 | S2 |
| C | NL | CV | 170 | 180 | 670 | 171 | 157 | 2.64 | 5.30 | 35.4 | 0.72 | 10.3 |
| C | LP | CV | 170 | 180 | 670 | 171 | 157 | 4.41 | 5.30 | 140 | 1.21 | 10.3 |
| C | LP | NCV | 170 | 180 | 670 | 171 | 157 | 5.28 | 6.83 | 136 | 1.40 | 9.37 |
| S | NL | CV | 168 | 198 | 703 | 169 | 161 | 1.72 | 1.40 | 37 | 0.83 | 3.36 |
| S | LP | CV | 168 | 182 | 657 | 169 | 157 | 3.48 | 5.16 | 129 | 1.64 | 8.78 |
| S | LP | NCV | 168 | 182 | 657 | 169 | 157 | 3.51 | 5.90 | 119 | 1.63 | 8.89 |
| D | NL | CV | 182 | 209 | 703 | 173 | 187 | 0.4 | 0.08 | 1.12 | 0.14 | 0.03 |
| D | LP | CV | 144 | 207 | 577 | 173 | 157 | 10.7 | 5.01 | 193 | 3.37 | 9.11 |
| D | LP | NCV | 140 | 195 | 577 | 173 | 157 | 10.7 | 6.08 | 190 | 3.42 | 9.10 |
| SC | NL | CV | 172 | 198 | 703 | 177 | 163 | 2.43 | 1.69 | 33.0 | 1.19 | 3.68 |
| SC | LP | CV | 168 | 194 | 513 | 147 | 146 | 16.8 | 4.01 | 157 | 1.70 | 11.61 |
| SC | LP | NCV | 168 | 194 | 513 | 147 | 146 | 17.1 | 3.03 | 164 | 2.50 | 11.7 |
| SD | NL | CV | 164 | 170 | 703 | 171 | 166 | 2.40 | 2.01 | 40.3 | 1.22 | 3.97 |
| SD | LP | CV | 160 | 168 | 513 | 171 | 157 | 4.19 | 7.27 | 176 | 1.73 | 9.43 |
| SD | LP | NCV | 160 | 154 | 513 | 171 | 157 | 4.18 | 9.13 | 176 | 1.66 | 9.48 |

C:Cost, S:Sensitivity, D:Deviation, SD:Sensitivity.Deviation, SC:Sensitivity.Cost,
NL:No Looping, LP:Looping, CV:Convergent, NCV:Non-convergent.

**Table 5.4.:** POTx algorithm vs. State-of-the-art algorithms

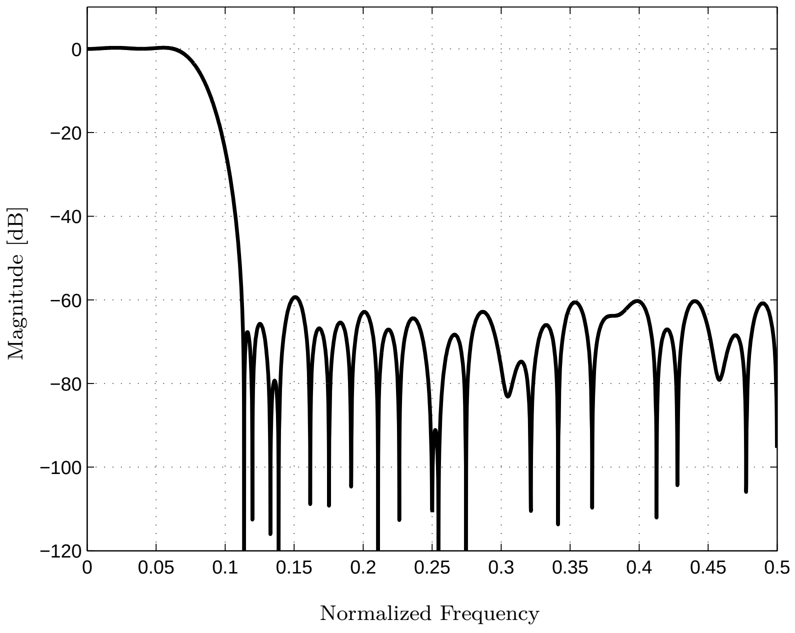
| Filter | Algorithm | $N$ | $Q$ | SPT | Gain (%) |
|---|---|---|---|---|---|
| A | RMZ | 59 | 15 | 200 | - |
| | TRE | 59 | 13 | 160 | 20 |
| | LI | 59 | 13 | 151 | 24.5 |
| | MILP | 59 | 13 | 145 | 27.5 |
| | FIRGAM | 59 | 13 | 145 | 27.5 |
| | POTx | 58 | 13 | 140 | 30 |
| B | RMZ | 105 | 13 | 232 | - |
| | TRE | 105 | 12 | 199 | 14.2 |
| | LI | 105 | 12 | 212 | 8.6 |
| | MILP | - | - | - | - |
| | FIRGAM | 105 | 11 | 169 | 27.2 |
| | POTx | 105 | 11 | 156 | 32.7 |
| C | RMZ | 325 | 15 | 820 | - |
| | TRE | 325 | 14 | 743 | 9.4 |
| | LI | 325 | 14 | 740 | 9.8 |
| | MILP | - | - | - | - |
| | FIRGAM | 325 | 13 | 549 | 33.0 |
| | POTx | 325 | 13 | 513 | 29.6 |
| L2 | RMZ | 63 | 15 | 197 | - |
| | LIM | 63 | 13 | 159 | 19.3 |
| | PMILP | 63 | - | 163 | 17.3 |
| | FIRGAM | 63 | 13 | 140 | 28.9 |
| | POTx | 61 | 13 | 147 | 25.3 |
| S2 | RMZ | 60 | 16 | 204 | - |
| | SAM | 60 | - | 174 | 14.7 |
| | PMILP | 60 | - | 174 | 14.7 |
| | FIRGAM | 60 | 15 | 160 | 21.6 |
| | POTx | 60 | 15 | 146 | 28.4 |

The time invested by POTx (Gray columns) to find the filter coefficients is given in Table 5.5. The remarkable reduction in computation time compared to the FIRGAM can be perceived from Table 5.5. The run time corresponds to the time spent by all the POTx settings, the 15 altered settings for allocation, iteration, and constraints (presented by the first three columns in Table III). Whereas, the best solution time is the time spent by POTx to achieve the filter coefficient set with the minimum number of non-zero terms.
The generated filter coefficients and the corresponding frequency response for filters A, B, S2, LP1, N1, using the NCV attribute, are given in Fig. 5.12, Fig. 5.13, Fig. 5.14, Fig. 5.15, and Fig. 5.16, respectively. Further, the generated filter coefficients and the corresponding frequency response for filters A, B, employing the CV attribute, are given in Fig. 5.17, Fig. 5.18, respectively. The presented coefficients

**Table 5.5.:** Computation times for FIRGAM vs. POTx

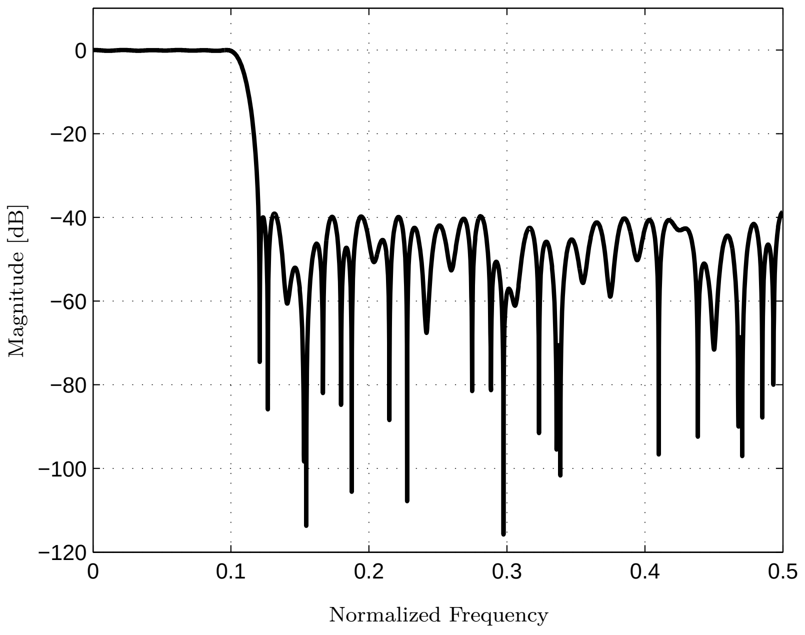
| Filter | Best Solution Time | | | Run Time | | |
|---|---|---|---|---|---|---|
| | FIRGAM | | POTx | FIRGAM | | POTx |
| A | 3h | 2m | 0.28m | 4h | 14m | 1.49m |
| B | | 9m | 0.05m | 24h | | 1.13m |
| C | 13h | 47m | 1.98m | 24h | | 28.7m |
| L2 | | 26m | 0.03m | | 54m | 0.40m |
| S2 | | 23m | 0.15m | | 27m | 1.98m |

are for half the filter due to symmetry.



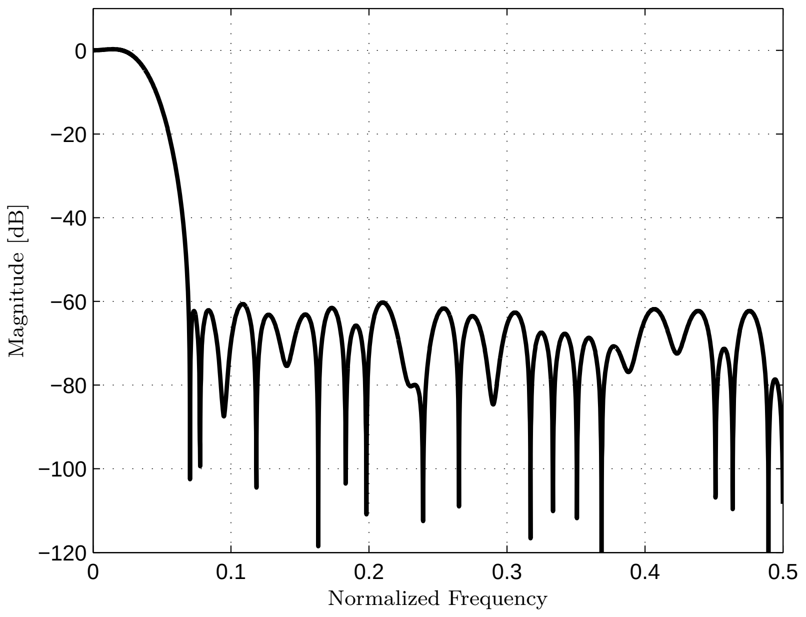
| | | | | |
|---|---|---|---|---|
| h(0)= 2 | h(6)= -32 | h(12)= 64 | h(18)= -146 | h(24)= 400 |
| h(1)= 4 | h(7)= -40 | h(13)= 92 | h(19)= -218 | h(25)= 704 |
| h(2)= 2 | h(8)= -40 | h(14)= 100 | h(20)= -246 | h(26)= 988 |
| h(3)= -2 | h(9)= -29 | h(15)= 78 | h(21)= -206 | h(27)= 1212 |
| h(4)= -10 | h(10)= -5 | h(16)= 24 | h(22)= -80 | h(28)= 1334 |
| h(5)= -20 | h(11)= 29 | h(17)= -56 | h(23)= 128 | |

**Figure 5.12.:** Filter A optimized coefficient set and its corresponding frequency response.



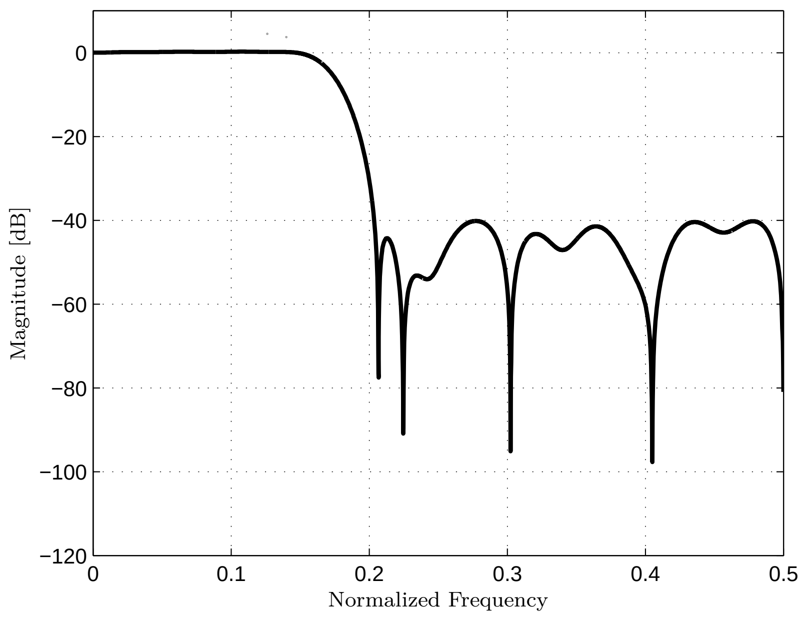
| | | | | |
|---|---|---|---|---|
| h(0)= -4 | h(11)= 0 | h(22)= 12 | h(33)= 16 | h(44)= -56 |
| h(1)= 0 | h(12)= 4 | h(23)= 14 | h(34)= -4 | h(45)= -88 |
| h(2)= 2 | h(13)= 8 | h(24)= 8 | h(35)= -24 | h(46)= -90 |
| h(3)= 4 | h(14)= 8 | h(25)= -4 | h(36)= -36 | h(47)= -40 |
| h(4)= 4 | h(15)= 4 | h(26)= -14 | h(37)= -32 | h(48)= 60 |
| h(5)= 4 | h(16)= -2 | h(27)= -20 | h(38)= -8 | h(49)= 190 |
| h(6)= 2 | h(17)= -8 | h(28)= -16 | h(39)= 20 | h(50)= 320 |
| h(7)= 0 | h(18)= -12 | h(29)= -4 | h(40)= 46 | h(51)= 416 |
| h(8)= -4 | h(19)= -8 | h(30)= 12 | h(41)= 52 | h(52)= 448 |
| h(9)= -4 | h(20)= 0 | h(31)= 24 | h(42)= 36 | |
| h(10)= -4 | h(21)= 8 | h(32)= 24 | h(43)= -4 | |

**Figure 5.13.:** Filter B optimized coefficient set and its corresponding frequency response.



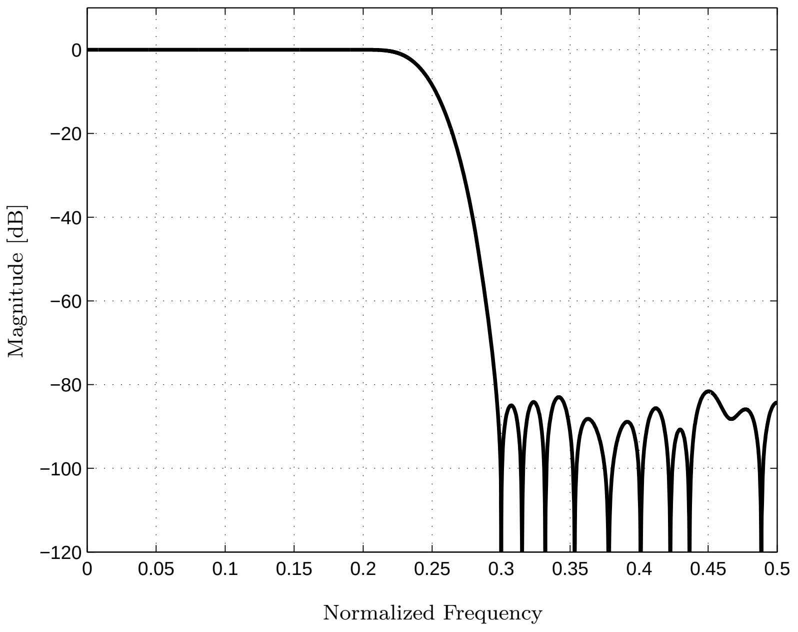
| | | | | |
|---|---|---|---|---|
| h(0)= 4 | h(6)= -32 | h(12)= -136 | h(18)= 80 | h(24)= 872 |
| h(1)= 2 | h(7)= -48 | h(13)= -140 | h(19)= 184 | h(25)= 1008 |
| h(2)= 0 | h(8)= -68 | h(14)= -132 | h(20)= 304 | h(26)= 1124 |
| h(3)= -4 | h(9)= -88 | h(15)= -106 | h(21)= 436 | h(27)= 1220 |
| h(4)= -10 | h(10)= -108 | h(16)= -64 | h(22)= 580 | h(28)= 1284 |
| h(5)= -20 | h(11)= -124 | h(17)= -2 | h(23)= 728 | h(29)= 1320 |

**Figure 5.14.:** Filter S2 optimized coefficient set and its corresponding frequency response.



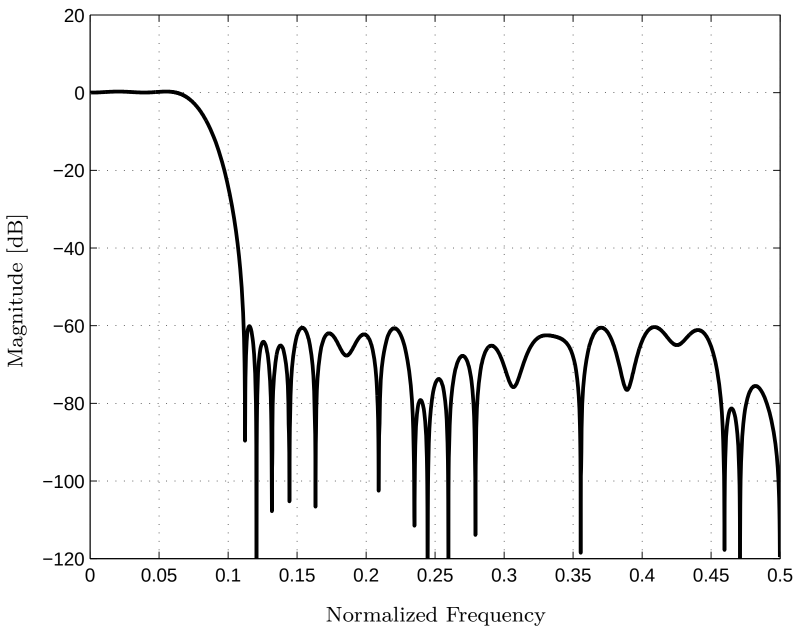
| | | | | |
|---|---|---|---|---|
| b(0)= 0 | b(5)= 1 | b(10)= -2 | b(15)= -21 | b(20)= -64 |
| b(1)= 0 | b(6)= 4 | b(11)= 8 | b(16)= 2 | b(21)= -58 |
| b(2)= -1 | b(7)= 2 | b(12)= 12 | b(17)= 32 | b(22)= 48 |
| b(3)= -2 | b(8)= -4 | b(13)= 1 | b(18)= 32 | b(23)= 212 |
| b(4)= -1 | b(9)= -8 | b(14)= -17 | b(19)= -16 | b(24)= 336 |

**Figure 5.15.:** Filter LP1 optimized coefficient set and its corresponding frequency response.



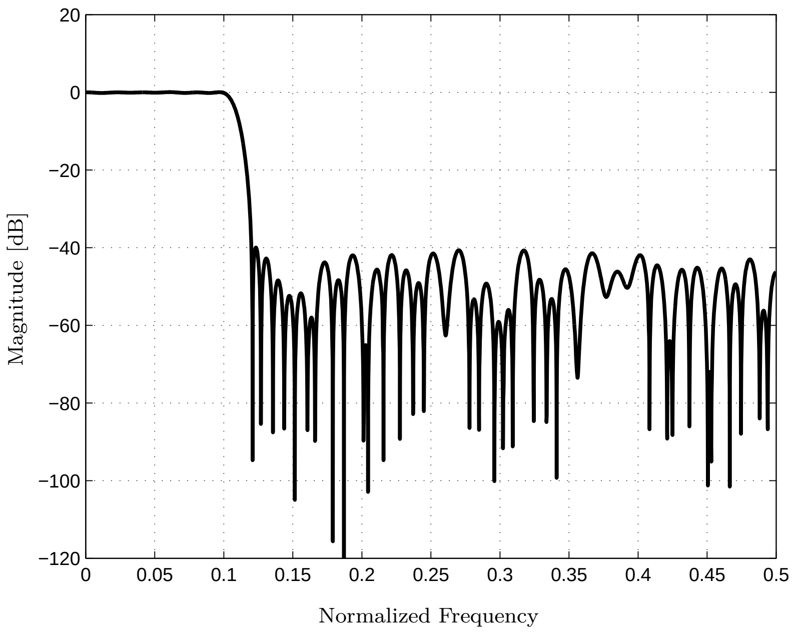
| | | | | |
|---|---|---|---|---|
| h(0)= -1 | h(7)= 20 | h(14)= -168 | h(21)= -326 | h(28)= 3706 |
| h(1)= -2 | h(8)= 22 | h(15)= 142 | h(22)= -1055 | h(29)= -562 |
| h(2)= 0 | h(9)= -38 | h(16)= 282 | h(23)= 394 | h(30)= -6664 |
| h(3)= 4 | h(10)= -48 | h(17)= -197 | h(24)= 1572 | h(31)= 592 |
| h(4)= 2 | h(11)= 64 | h(18)= -453 | h(25)= -460 | h(32)= 20762 |
| h(5)= -11 | h(12)= 93 | h(19)= 258 | h(26)= -2361 | h(33)= 32166 |
| h(6)= -9 | h(13)= -99 | h(20)= 700 | h(27)= 516 | |

**Figure 5.16.:** Filter N1 optimized coefficient set and its corresponding frequency response.



| | | | | |
|---|---|---|---|---|
| h(0)= 2 | h(6)= -32 | h(12)= 64 | h(18)= -146 | h(24)= 400 |
| h(1)= 3 | h(7)= -40 | h(13)= 92 | h(19)= -218 | h(25)= 704 |
| h(2)= 2 | h(8)= -40 | h(14)= 100 | h(20)= -246 | h(26)= 988 |
| h(3)= -2 | h(9)= -29 | h(15)= 78 | h(21)= -206 | h(27)= 1212 |
| h(4)= -10 | h(10)= -5 | h(16)= 24 | h(22)= -80 | h(28)= 1334 |
| h(5)= -20 | h(11)= 29 | h(17)= -57 | h(23)= 128 | |

**Figure 5.17.:** Filter A optimized coefficient set using CV and its corresponding frequency response.



| | | | | |
|---|---|---|---|---|
| h(0)= -4 | h(11)= 0 | h(22)= 13 | h(33)= 16 | h(44)= -56 |
| h(1)= 0 | h(12)= 4 | h(23)= 14 | h(34)= -4 | h(45)= -90 |
| h(2)= 2 | h(13)= 8 | h(24)= 8 | h(35)= -24 | h(46)= -90 |
| h(3)= 4 | h(14)= 8 | h(25)= -4 | h(36)= -36 | h(47)= -40 |
| h(4)= 4 | h(15)= 4 | h(26)= -14 | h(37)= -32 | h(48)= 60 |
| h(5)= 4 | h(16)= -2 | h(27)= -20 | h(38)= -10 | h(49)= 190 |
| h(6)= 2 | h(17)= -8 | h(28)= -16 | h(39)= 20 | h(50)= 320 |
| h(7)= -1 | h(18)= -10 | h(29)= -4 | h(40)= 46 | h(51)= 416 |
| h(8)= -4 | h(19)= -8 | h(30)= 12 | h(41)= 52 | h(52)= 448 |
| h(9)= -6 | h(20)= 0 | h(31)= 24 | h(42)= 36 | |
| h(10)= -4 | h(21)= 8 | h(32)= 25 | h(43)= -4 | |

**Figure 5.18.:** Filter B optimized coefficient set using CV and its corresponding frequency response.

## 5.6. Common Sub-expression Elimination

Common sub-expression elimination (CSE) aims to allocate and eliminate the redundant computations consisting of two non-zero terms [82]. CSE leads to about 50% reduction in the multiplier adders (MA) [83]. Whereas, considering only the most common sub-expression leads to 33% reduction in the MAs [83]. The two frequent common sub-expressions are the 3-bit common sub-expression (3CS) 101 and $10\bar{1}$ [83]. Then comes the 4-bit common sub-expression (4CS) the 1001 and $100\bar{1}$ [84]. Several CSE approaches [82], [83], [85], [86], [87] have been proposed in literature which consider the pattern search criterion. The CSE algorithms given in [82], [83] and [85] depend on the frequency of the most occurring CS. The work given in [83] considers super sub-expression (SS), e.g. searching for various CS patterns such as 101 and 1001 within the same coefficient set. However, SS is recommended for coefficients with large bit-width [85]. A recent comparison for the different CSE algorithms can be found in [86]. The major trade-off in CSE is between computation time and allocating the CS with the maximum number of occurrence. The choice of the CS is a substantial process in order to achieve the maximum reduction in the MA, which depends on the frequency of occurrence and the estimated reduction in MA [83]. Throughout CSE each individual bit is not allowed to occur in more than one pair to assure overlapping is not counted twice [83].
The author developed a CSE algorithm which combines several approaches in order to achieve a fair balance between computation time and reliable reduction in the number of operations. The CSE is used later on, in the following chapter, to steer the constant multiplier construction on the RTL level. The proposed CSE algorithm can be explained as follow:

- CSE Pre-processing
  - Calculate the number of occurrences for 3CS and 4CSs in overall 'overlapped'
  - Calculate the number of occurrence of each individual CS
- CSE Overlapping
  - Exclude overlapping between CS patterns according to the most frequent occurred CS and in order to maximize the overall number of CS as well

On the one hand, the purpose of the pre-processing step is to process the statistics of CS patterns within the filter coefficients to use it later to steer the process of eliminating the overlapping between CSs. On the other hand, excluding overlapping between CSs has a critical impact on the reduction of number of operations. As an example, consider the following coefficients:

$h_k(1) = 10100101$
$h_k(2) = 10100\bar{1}$
$h_k(3) = 100\bar{1}0101$

The coefficient $h_k(1)$ has three CSs, 101 occurred two times and 1001 occurred one time. The choice between 3CS or 4CS pattern will consequently influence the number of reduction in MAs. However, the coefficient $h_k(3)$ has three CSs as well, the allocation of which 3CS, $\bar{1}01$ or 101, will affect the number of MAs, i.e. if $\bar{1}01$ is chosen instead of 101 consequently $100\bar{1}$ will not be considered. While $h_k(2)$ has only two CSs, the choice should be for the most frequently occurring CS, 3CS or 4CS, which will be determined from the pre-processing step. The detailed pseudocode of the proposed algorithm is given in Fig. 5.19, where the symbols and their d are given in Table 5.6. The CS patterns are defined as

$$\begin{array}{lcccl} r_1 & = & 101 & \sim & 2^0 + 2^{-2} \\ r_2 & = & 10\bar{1} & \sim & 2^0 - 2^{-2} \\ r_3 & = & 1001 & \sim & 2^0 + 2^{-3} \\ r_4 & = & 100\bar{1} & \sim & 2^0 - 2^{-3} \end{array}$$

It should be noted that, before CSE a reduction redundancy step is performed. The redundancy reduction removes zeros, ones, power-of-two terms, tailing shifts, and redundant terms (explained later numerically). The ones, the power-of-two terms, and the tailing shifts are excluded before CSE as they are hard-wire routes without hardware overhead for implementation. The CS pairs, and their negated versions, used in this work are the 3CS and 4CS. The algorithm is applicable for binary and CSD coefficients, however the presented results consider CSD only for consistent comparison with the literature.

Illustratively, CSE detailed analysis is performed and presented for benchmark filter S2. The unique coefficient set is
$\hat{u}_k = [3, 5, 11, 17, 23, 27, 33, 35, 63, 65, 91, 109, 145, 151, 165, 281, 305, 321]$. The output from the pre-processing step is given in Table 5.7 and Table 5.8, which shows the statistics and distribution of 3CS and 4CS within the filter coefficients. The CSD representation of the unique filter coefficient set, i.e. after excluding redundancy, is shown in Table 5.9. In order to verify the consistency of both tables, as an example for 3CS, the $r_1$ and $r_2$ columns of Table 5.8 is added together which should be equal to the summation of the multiplying *cnt_3cs* and *cnt_r_3cs* columns of Table 5.7. Table 5.9 shows the final output after excluding the overlapping between CSs. Table 5.12 shows the number of multiplier adders (MA) for the benchmark filters using the proposed algorithm compared to the Spiral and the FIRGAM algorithms. The result shows that, there are 17 CS, non-overlapped, which leads to (17+4) MAs as shown in Table 5.10. Comparing Fig. 5.14 and Table 5.10 for filter S2, it can be observed that there are four absent coefficients (after excluding the zeros and power-of-two coefficients) which are: $80, 136, 1008,$ and $872$. They are implicitly implemented through other coefficients as follows: $80 = 2^4 \times 5$,

```
1:  procedure CSE Pre-processing($\hat{h}_k$)
2:      $h_k \leftarrow$ Binary $\vee$ CSD convert $\hat{h}_k$
3:      for $i \rightarrow 1, M$ do
4:          for $j \rightarrow 1, Q$ do
5:              $tmp\_r_x \leftarrow r_x$ find $h_k(i, j)$                ▷ overlapped patterns
6:              if not empty $tmp\_r_x$ then
7:                  $cnt\_r_x$ $++$
8:                  $k++$
9:              end if
10:         end for
11:     end for
12:     $cnt\_3cs = \Sigma_{i=1}^{M} \Sigma_{j=1}^{k} r_1(i, j) + r_2(i, j)$
13:     $cnt\_4cs = \Sigma_{i=1}^{M} \Sigma_{j=1}^{k} r_3(i, j) + r_4(i, j)$
14:     $cnt\_r_1 = \Sigma_{i=1}^{M} \Sigma_{j=1}^{k} r_1(i, j)$
15:     $cnt\_r_2 = \Sigma_{i=1}^{M} \Sigma_{j=1}^{k} r_2(i, j)$
16:     $cnt\_r_3 = \Sigma_{i=1}^{M} \Sigma_{j=1}^{k} r_3(i, j)$
17:     $cnt\_r_4 = \Sigma_{i=1}^{M} \Sigma_{j=1}^{k} r_4(i, j)$
18: end procedure
19: Form analysis tables for total and individual CS patterns
20: procedure CSE Overlapping($\hat{h}_k$)
21:     for $i = 1 \rightarrow M$ do
22:         if $cnt\_3cs > cnt\_4cs$ then
23:             $flag \leftarrow 3cs$
24:         else
25:             $flag \leftarrow 4cs$
26:         end if
27:         if $cnt\_cs = cnt\_rcs \wedge cnt\_cs = 1$ then
28:             $cnt(i) \leftarrow cnt\_r$
29:         else if $cnt\_cs < cnt\_rcs \wedge cnt\_cs = 1$ then
30:             $cnt(i) \leftarrow cnt\_r$
31:         else if $cnt\_cs > cnt\_rcs \wedge cnt\_cs > 1$ then
32:             if $OL$ then                                              ▷ flag
33:                 $cnt(i)++$
34:                 if $OL$ then                                      ▷ not flag
35:                     $pattern \leftarrow$ pattern not overlapped
36:                 end if
37:             else
38:                 $cnt(i) = cnt\_r$
39:                 $pattern$
40:             end if
41:         else
42:         end if
43:     end for
44: end procedure
```

**Figure 5.19.:** Pseudocode for CSE algorithm.

**Table 5.6.:** Symbol definition for CSE algorithm given in Figure 5.19

| Symbol | Abbreviation |
|---|---|
| $\hat{h}_k$ | Scaled filter coefficients |
| $h_k$ | Binary or CSD representation of the filter coefficients |
| $M$ | Half the filter length |
| $Q$ | Quantization bit-width |
| $cnt\_3cs$ | Total number of occurrence for 3CS patterns ‘overlapping is not excluded’ |
| $cnt\_4cs$ | Total number of occurrence for 4CS patterns ‘overlapping is not excluded’ |
| $cnt\_r_1$ | Number of occurrence for the CS pattern $r_1$ |
| $cnt\_r_2$ | Number of occurrence for the CS pattern $r_2$ |
| $cnt\_r_3$ | Number of occurrence for the CS pattern $r_3$ |
| $cnt\_r_4$ | Number of occurrence for the CS pattern $r_4$ |
| $k$ | Number of occurrence of the an individual CS pattern within the same coefficient |
| $cnt$ | Total number of occurrence of CS (3CS and 4CS) considering excluded overlapping |
| $pattern$ | Variable holds the identity of the CS pattern within each coefficient after excluding overlapping |
| $OL$ | Overlapping |
| $r_x$ | CS pattern where $x \in \{1, 2, 3, 4\}$ |

$136 = 2^3 \times 17$, $1008 = 2^4 \times 126$, and $872 = 2^3 \times 436$. Consequently, redundant terms have been eliminated from the filter coefficient set by preserving the odd fundamentals only [55]. It has to be noted, that the number of MAs is calculated according to the number of non-zero terms in each coefficient and the number of CS patterns within the coefficient. The number of MA for the optimized benchmark filters is given in Table 5.11.
Further, the proposed algorithm was compared to the $H_{cub}$ algorithm [87] using the online SPIRAL generator [88], as shown in Table 5.12.

**Table 5.7.:** CSE pre-processing results for 3CS and 4CS

| $\hat{h}_k$ | 3CS | | 4CS | |
|---|---|---|---|---|
| | $cnt\_3cs$ | $cnt\_r\_3cs$ | $cnt\_4cs$ | $cnt\_r\_4cs$ |
| 1 | 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 2 | 1 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 |
| 5 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 |
| 7 | 0 | 0 | 0 | 0 |
| 8 | 1 | 1 | 1 | 1 |
| 9 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |
| 11 | 2 | 1 | 1 | 1 |
| 12 | 2 | 1 | 1 | 1 |
| 13 | 0 | 0 | 1 | 1 |
| 14 | 2 | 1 | 1 | 1 |
| 15 | 1 | 2 | 1 | 1 |
| 16 | 1 | 1 | 2 | 1 |
| 17 | 2 | 1 | 0 | 0 |
| 18 | 1 | 1 | 0 | 0 |

**Table 5.8.:** CSE pre-processing results for $r_1, r_2, r_3, r_4$

| Filter | 3CS | | 4CS | |
|---|---|---|---|---|
| | $r_1$ | $r_2$ | $r_3$ | $r_4$ |
| S2 | 10 | 9 | 7 | 3 |

**Table 5.9.:** CSE optimized results due to the proposed algorithm

| $\hat{u}_k$ | *cnt* | *pattern* | CSD | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 1 | $r_2$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 |
| 5 | 1 | $r_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 11 | 1 | $-r_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 | -1 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 23 | 1 | $r_2$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | -1 |
| 27 | 1 | $-r_1$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | -1 | 0 | -1 |
| 33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 35 | 1 | $r_2$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | -1 |
| 63 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | -1 |
| 65 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 91 | 2 | $-r_1, r_2$ | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | -1 | 0 | -1 |
| 109 | 2 | $-r_2, r_4$ | 0 | 0 | 0 | 1 | 0 | 0 | -1 | 0 | -1 | 0 | 1 |
| 145 | 1 | $r_3$ | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 151 | 2 | $r_1, -r_3$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | -1 | 0 | 0 | -1 |
| 165 | 2 | $r_1$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 281 | 2 | $r_3, -r_4$ | 0 | 0 | 1 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | 1 |
| 305 | 1 | $r_1$ | 0 | 0 | 1 | 0 | 1 | 0 | -1 | 0 | 0 | 0 | 1 |
| 321 | 1 | $r_1$ | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

**Table 5.10.:** Filter S2 CSE

| $\hat{h}_k$ | $\hat{u}_k$ | PFP+CSE | MA |
|---|---|---|---|
| 20 | $2^2 \times 5$ | $2^{-7}(r_1)$ | 0 |
| 33 | $2^0 \times 33$ | $2^{-6}(2^0 + 2^{-5})$ | 1 |
| 48 | $2^4 \times 3$ | $2^{-5}(r_2)$ | 0 |
| 68 | $2^2 \times 17$ | $2^{-5}(2^0 + 2^{-4})$ | 1 |
| 88 | $2^3 \times 11$ | $2^{-4}(r_2 - 2^{-4})$ | 1 |
| 126 | $2^1 \times 63$ | $2^{-4}(2^0 - 2^{-6})$ | 1 |
| 140 | $2^2 \times 35$ | $2^{-4}(2^0 + 2^{-3}.r_2)$ | 1 |
| 130 | $2^1 \times 65$ | $2^{-4}(2^0 + 2^{-6})$ | 1 |
| 108 | $2^2 \times 27$ | $2^{-4}(2^0 - 2^{-3}.r_1)$ | 1 |
| 184 | $2^3 \times 23$ | $2^{-3}(r_2 - 2^{-5})$ | 1 |
| 302 | $2^1 \times 151$ | $2^{-3}(r_1 - 2^{-4}.r_3)$ | 1 |
| 436 | $2^2 \times 109$ | $2^{-2}(r_4 - 2^{-5}.r_2)$ | 1 |
| 580 | $2^2 \times 145$ | $2^{-2}(r_3 + 2^{-7})$ | 1 |
| 728 | $2^3 \times 91$ | $2^{-1}(r_2 - 2^{-5}.r_1)$ | 1 |
| 1124 | $2^2 \times 281$ | $2^{-1}(r_3 - 2^{-5}.r_4)$ | 1 |
| 1220 | $2^2 \times 305$ | $2^{-1}(r_1 - 2^{-4} + 2^{-8})$ | 2 |
| 1284 | $2^2 \times 321$ | $2^{-1}(r_1 + 2^{-8})$ | 1 |
| 1320 | $2^3 \times 165$ | $2^{-1}(r_1 + 2^{-5}.r_1)$ | 1 |
| Sum | | | 17 |

**Table 5.11.:** POTx algorithm performance evaluation

| Filter | Algorithm | $N$ | $Q$ | SPT | Gain (%) | MA | SA | Total | Best Solution Time | Run Time |
|---|---|---|---|---|---|---|---|---|---|---|
| | RMZ | 59 | 15 | 200 | - | 29 | 58 | 87 | - - | - - |
| A | FIRGAM | 59 | 13 | 145 | 27.5 | 18 | 58 | 76 | 3h 2m | 4h 14m |
| | SHI | 59 | 10 | 137 | 31.5 | 14 | 54 | 68 | - - | 50h 34m |
| | POTx | 59 | 13 | 144 | 28.0 | 16 | 58 | 74 | - 0.28m | - 1.49m |
| | RMZ | 105 | 13 | 232 | - | 24 | 102 | 126 | - - | - |
| B | FIRGAM | 105 | 11 | 169 | 27.2 | 11 | 100 | 111 | - 9m | 24h |
| | POTx | 105 | 11 | 168 | 27.5 | 10 | 99 | 109 | - 0.05m | 1.13m |
| | RMZ | 325 | 15 | 820 | - | 57 | 322 | 379 | - - | - |
| C | FIRGAM | 325 | 13 | 549 | 33.0 | 22 | 306 | 328 | 13h 47m | 24h - |
| | POTx | 325 | 13 | 513 | 37.4 | 24 | 301 | 325 | - 1.98m | - 28.7m |
| | RMZ | 63 | 15 | 197 | - | 30 | 62 | 92 | - - | - |
| L2 | FIRGAM | 63 | 13 | 140 | 28.9 | 18 | 62 | 80 | - 26m | - 54m |
| | SHI | 63 | 10 | 148 | 24.8 | 17 | 56 | 73 | - - | 16h 28m |
| | POTx | 61 | 13 | 147 | 25.3 | 17 | 59 | 76 | - 0.03m | - 0.40m |
| | RMZ | 60 | 16 | 204 | - | 32 | 59 | 91 | - - | - |
| S2 | FIRGAM | 60 | 15 | 160 | 21.6 | 27 | 59 | 86 | - 23m | - 27m |
| | SHI | 60 | 10 | 168 | 17.6 | 17 | 59 | 76 | - - | 16h 42m |
| | POTx | 60 | 15 | 146 | 28.4 | 19 | 58 | 77 | - 0.15m | - 1.98m |

**Table 5.12.:** Number of multiplier adders using the proposed CSE algorithm compared to Spiral and FIRGAM

| Filter | Spiral [88] | FIRGAM [65] | Proposed |
|---|---|---|---|
| A | 16 | 18 | 18 |
| B | 10 | 11 | 10 |
| L2 | 17 | 18 | 19 |
| S2 | 18 | 27 | 21 |

## 5.7. Summary

Novel allocation schemes have been presented in this chapter. The proposed schemes, namely: coefficient deviation, cost, and hybrid showed an adequate performance for coefficients presorting compared to the sensitivity based scheme. However, employing the sensitivity based scheme within the hybrid scheme resulted in an efficient allocation scheme.
Further, a heuristic algorithm named POTx was developed for reducing the number of non-zero terms in filter coefficients. The POTx showed on one hand remarkable savings in the computation time due to: a) a bounded search space for candidates by upper and lower bounds, b) being monotonic, and c) employing break-off and no-change states for terminating the searching process. On the other hand, the considerable gain in reducing the number of non-zero terms is achieved due to: a) presorted coefficients using dedicated allocation schemes, b) multiple optimization iterations, c) unconstrained number of non-zero terms, and d) designing the filter with 3-dB tighter constraints.
The proposed schemes and algorithm were verified using seven benchmark filters. The results revealed a remarkable reduction in the computation time for all the filters. At the same time, a considerable reduction in the number of non-zero terms for five filters out of seven (more than 70%) is achieved.
A straight forward common sub-expression elimination algorithm was developed in order to reduce the number of multiplier adders. The algorithm is based on pattern search and excludes pattern overlapping based on the frequency of occurrence. Employing common sub-expression elimination introduces around 40% reduction in the number of multiplier adders.

# 6. Architectural System Level Power Optimization

## 6.1. Introduction

Power optimization can be achieved on the system level through the manipulation and re-arrangement of blocks or sub-blocks while preserving the same functionality. This is achieved by combining, mixing, or constructing hybrid architectures for reducing power dissipation. This work has been published in [3].

## 6.2. Combined DF and TF Architectures

Polyphase FIR filters are implemented either in direct-form (DF [1]) or transposed-form (TF [2]). Each of these two forms has some drawbacks. From an implementation point of view both structures contain the same number of multipliers and adders, whereas TF has fewer delays. However, the width of the delays for the TF is generally larger, depending on the coefficient quantization bit-width ($Q$) and decimator input bit-width ($W_i$). Furthermore, the fan-out of the input node is high as a single node drives several multipliers. On the other hand, the critical path for the TF FIR filter is only one multiplier and one multi-operand adder compared with one multiplier and $\lceil N/M \rceil$ multi-operand adders for the DF [89], where $N$ and $M$ are the filter order and decimation factor, respectively. Furthermore, TF allows multiplier optimization because of the shared multiplier structure. The work presented in [90] has chosen the DF since the polyphase decomposition highly reduces the operating frequency of the filter, so the critical path is no longer a problem. On the other hand, [91], [92], [93] have chosen the TF to allow shared multipliers for optimization. In [93] the author has chosen a hybrid structure to combine both the TF and DF structures within the same decimation filter.
An in-depth comparison of both structures for polyphase decimators in terms of power consumption is still missing and it is the topic of this chapter. This is accomplished by an analysis of a typical decimator, which is implemented using different structures and both TF and DF topologies. The results of this analysis are then used to develop a more universal criterion for choosing between TF and

[1]Detailed topology is given in 7.6.2
[2]Detailed topology is given in 7.6.2

**Table 6.1.:** Design specifications

| Parameter | Value |
|---|---|
| Sampling frequency $f_s$ | 960 kHz |
| Oversampling ratio OSR | 24 |
| Signal frequency $f_{signal}$ | 5 kHz |
| IBN of modulator output | -83.39 dB |
| IBN of decimator output | -80.67 dB |

DF in polyphase decimators.

### 6.2.1. Analysis of Power Consumption

A decimator for a low-pass $\Sigma\Delta$ modulator with an oversampling ratio of 24 is used as a case study to investigate the effect of different numbers of decimation stages and the use of TF and DF filter structures on power consumption. It is more efficient to implement the sampling rate reduction in a series of decimation filters [29], though different topologies for the multi-stage decimation filters have been developed for high accuracy analysis using the MSD-toolbox. The investigated 2-stage, 3-stage, and 4-stage topologies are shown in Fig. 6.1, all of which have been implemented using TF and DF polyphase filters. The $M$'s are presented inside each block by the downward arrow. The design parameters for the case study is presented in Table 6.1 for a $3^{rd}$ order low-pass $\Sigma\Delta$ modulator. Figure 6.2 shows the total power consumption for the multi-stage decimators (2, 3, and 4 stages) which are presented in Fig. 6.1. The TF appears to be the more efficient filter architecture considering the power consumption. The bar chart in Fig. 6.3 indicates the total power consumption in each decimation stage, while the pie chart presents the power consumption distribution over the multipliers (MUs), multi-operand adders (MOAs), and delays (DLs) in each decimation stage. Analyzing the power distribution of the various decimator topologies reveals that the power consumption in the MOAs and MUs is almost identical in the corresponding DF and TF filters, as given in Tables 6.2 and 6.3. Figure 6.2 and Table 6.4 show the power dissipation distribution over the stages for the different topologies. Analyzing the results presented in this graph and this table reveals: a) The last decimation stage consumes around 50% of the total power in the whole decimator filter, b) MOAs are the dominant source of power consumption in each decimation stage. Analyzing the results presented in Table 6.4 reveals that the DF filters in the first stage actually consume less power than the transposed form. This observation is confirmed analytically in section 6.2.2.

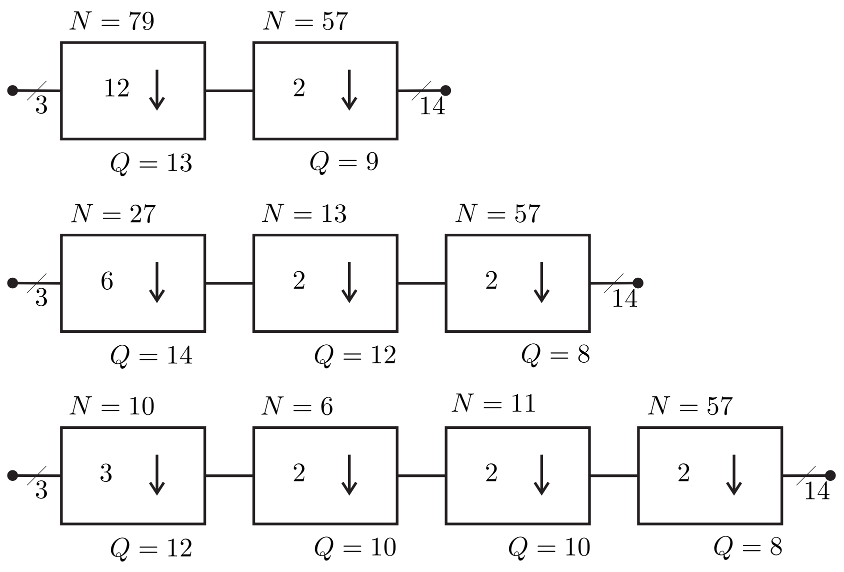


**Figure 6.1.:** A case study for various multi-stage decimator topologies for power consumption of FIR polyphase structures, where $N$ denotes the filter length, $Q$ is the quantization bit-width for filter coefficients, and $M$ is the decimation factor..

**Table 6.2.:** Power consumption [$\mu$W] distribution for 3-stages of the decimation

| | DF | | | TF | | |
|---|---|---|---|---|---|---|
| | $1^{st}$ | $2^{nd}$ | $3^{rd}$ | $1^{st}$ | $2^{nd}$ | $3^{rd}$ |
| MUs | 4.35 | 2.31 | 2.43 | 4.54 | 2.32 | 2.74 |
| MOAs | 4.85 | 3.31 | 7.67 | 4.90 | 3.29 | 7.21 |
| DLs | 1.15 | 2.68 | 14.0 | 0.70 | 1.19 | 7.34 |
| Sum | 10.3 | 8.85 | 24.1 | 10.1 | 6.80 | 17.3 |

### 6.2.2. TF|DF Selection Criterion

As shown in section 6.2.2, there are filters where it is more advantageous to use a DF implementation and others where the TF shows lower power consumption. To avoid time consuming power simulations, an analytic criterion to predict which structure is more advantageous is desirable. In [93] a study of the FIR filter complexity according to the number of DLs and MUs for both TF and DF FIR filters was carried out but a selection criterion was not considered. Further, the quantization bit-width $Q$ and decimator input bit-width $W_i$ was not included for calculating the number of internal DLs bit-width.
As observable in Tables 6.2 and 6.3 and also in the power distribution of the decimator topologies as given in Fig. 6.3, the differentiation between both structures can be obtained by solely regarding DLs. The major difference between DF and TF structures is the number of DLs and the internal bit-width of the DLs. The author has modeled the overall bit-width of all DLs in the filter for the TF and

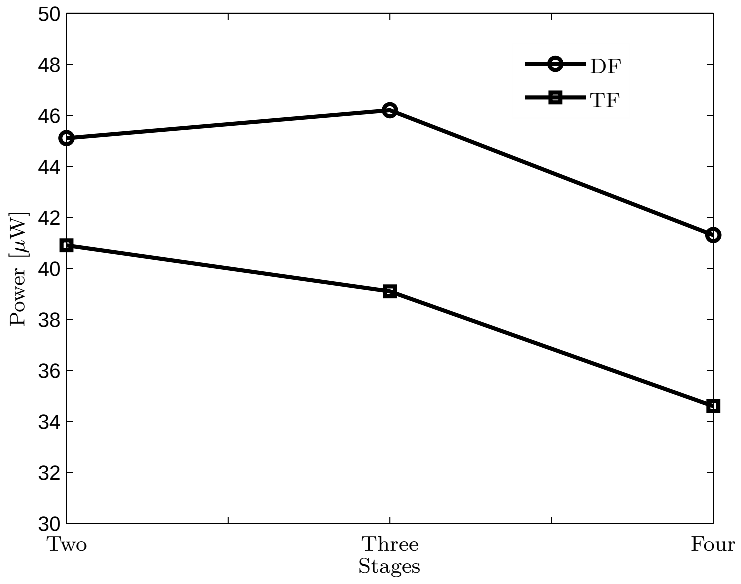


**Figure 6.2.:** Total power consumption in two, three, and four multistage decimation topologies, where DF is Direct-form and TF is Transposed-form.

**Table 6.3.:** Power consumption [$\mu$W] distribution for 4-stages of the decimation

| | DF | | | | TF | | | |
|---|---|---|---|---|---|---|---|---|
| | 1st | 2nd | 3rd | 4th | 1st | 2nd | 3rd | 4th |
| MUs | 0.98 | 0.29 | 2.01 | 3.06 | 1.0 | 0.29 | 2.02 | 3.32 |
| MOAs | 1.87 | 1.31 | 2.55 | 7.68 | 1.8 | 1.27 | 2.55 | 7.25 |
| DLs | 0.54 | 0.92 | 2.00 | 14.0 | 0.83 | 0.73 | 1.56 | 7.35 |
| Sum | 3.39 | 2.52 | 6.56 | 24.74 | 3.63 | 2.29 | 6.13 | 17.92 |

**Table 6.4.:** Power consumption [$\mu$W] distribution in multistage decimation topologies

| | | 1st | 2nd | 3rd | 4th |
|---|---|---|---|---|---|
| 2-Stages | DF | 22.0 | 23.3 | - | - |
| | TF | 21.3 | 19.6 | - | - |
| 3-Stages | DF | 11.2 | 9.32 | 25.3 | - |
| | TF | 11.6 | 8.73 | 18.9 | - |
| 4-Stages | DF | 4.04 | 3.69 | 7.49 | 25.9 |
| | TF | 4.35 | 3.48 | 7.12 | 19.6 |

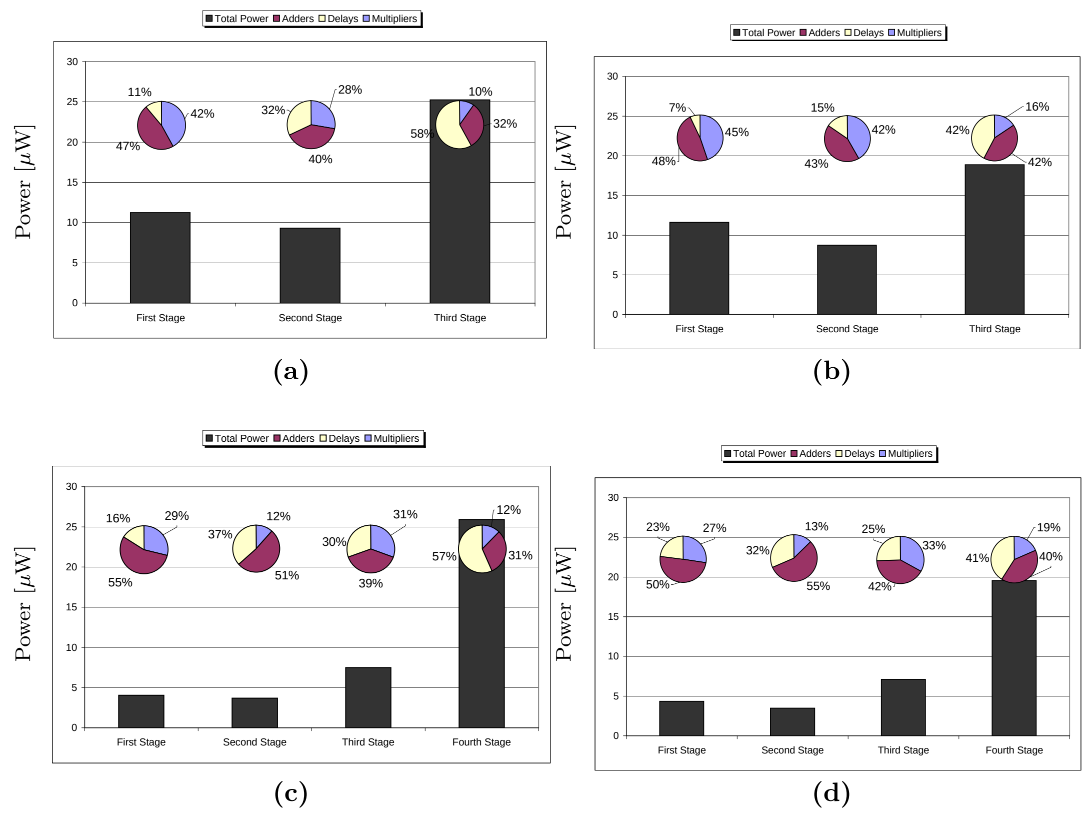


**Figure 6.3.:** Power dissipation distribution in multistage decimation topologies.

DF structures analytically by (6.1) and (6.2), respectively, with decimation factor $M$, filter order $N$, quantization bit-width $Q$, and decimator input bit-width $W_i$.

$$DL_{DF} = (N - M)W_i \tag{6.1}$$

$$DL_{TF} = (W_i + Q)\left(\lceil \frac{N}{M} - 1 \rceil\right) \tag{6.2}$$

In Fig. 6.4 these dependencies are shown for different decimation factors $M$ at $N = 27$ and $N = 77$ with $W_i = 1$, and at $W_i = 1$ and $W_i = 3$ with $N = 77$ (at $Q = 16$). Illustratively, for a filter length ($N = 27$) and input bit-width ($W_i = 1$), it is beneficial to use DF structure with decimation factor up to ($M = 27$) as shown in Fig. 6.4.a. If the filter length increased to ($N = 77$), it is beneficial to use TF structure with decimation factor up from ($M = 19$) as shown in Fig. 6.4.a. However, for larger input bit-width ($W_i = 3$) it is advantageous to employ DF up to decimation factor ($M = 7$) only, as given in Fig. 6.4.b. This graph reveals that the DF is advantageous at low input bit-width, condition typically found at the first decimation stage of a multi-stage decimator (as the filter input bit-width increases at each stage). Furthermore, the graph shows that the DF is especially advantageous for decimation filters with single-bit inputs.
Figure 6.5, illustrates the situation for different filter lengths $N$ at two decimation

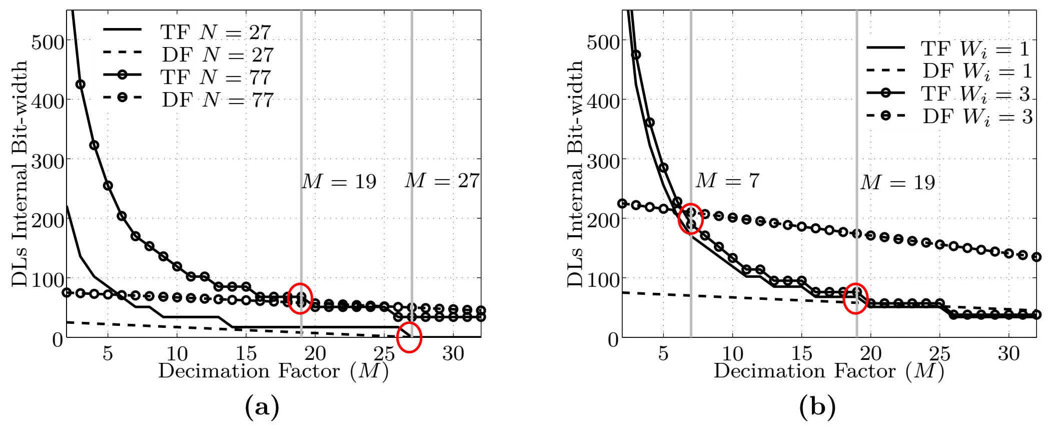


**Figure 6.4.:** Delays width estimation for DF and TF with respect to $M$ at $Q = 16$ for (a) $N = 27, 77$ at $W_i = 1$ (b) $W_i = 1, 3$ at $N = 77$.

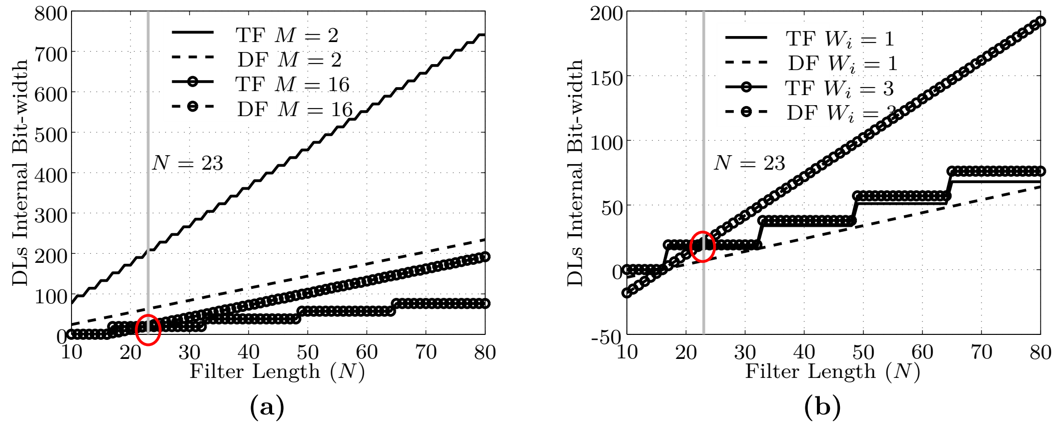


**Figure 6.5.:** Delays width estimation for DF and TF with respect to $N$ at $Q = 16$ (a) $M = 2, 16$ at $W_i = 3$ (b) $W_i = 1, 3$ at $M = 16$.

factors $M = 2$ and $M = 16$ with $W_i = 3$, and with $W_i = 1$ and $W_i = 3$ at $M = 16$ (at $Q = 16$). It is beneficial to use DF structure (despite the filter length) for filter with $W_i = 3$ and $M = 2$ as shown in Fig. 6.5.a. Whereas, if the decimation factor increased to ($M = 16$) it is advantageous to use TF structure for filter length larger than ($N = 23$) as shown in Fig. 6.5.a. However, if the input bit-width reduced to ($W_i = 1$) it is recommended to employ DF as shown in Fig. 6.5.b. For low decimation factors the DF is advantageous independent of filter order, in other cases the situation occurs that the DF is only advantageous at low filter orders due to the step characteristic in the number of DLs for the TF structure.

**Table 6.5.:** Power consumption simulated vs. calculated for DLs in the first decimation stage for the case study topologies

| | Simulated $\mu$W | | Calculated #DLs | | Ratio (DF/TF%) | |
|---|---|---|---|---|---|---|
| | TF | DF | TF | DF | Sim. | Calc. |
| 2-stages | 1.30 | 2.80 | 96 | 201 | 215% | 210% |
| 3-stages | 0.70 | 0.63 | 69 | 63 | 90% | 92% |
| 4-stages | 0.83 | 0.54 | 270 | 162 | 65% | 60% |

### 6.2.3. Performance Evaluation and Results

To validate the proposed criterion, it has been applied to the case study described in section 6.2.1 to compare the simulated power results to the values given by (6.1) and (6.2). Table 6.5 shows the simulated power consumption of the delays of the first stage for the 2-stage, 3-stage, and 4-stage topologies, along with the number of DLs according to (6.1) and (6.2). The third column gives the ratio between the power consumption for the TF and DF and the number of DLs for TF and DF, respectively. These ratios show a good fit with an error well below 5% in all cases, thus showing the applicability of the proposed criterion. Using this criterion, it is thus possible to define an optimum mixed architecture for the different topologies. The criterion has been applied on the case study. Table 6.5 shows how the developed criterion can be used for designing a decimator in mixed architectures. For the 2-stages topology, the TF is recommended in the first stage as it exhibits around 50% less DLs bit-width than DF. For the 4-stages topology on the other hand, it is advantageous to use DF in the first stage and TF in all other stages, as shown by Table 6.5.

## 6.3. Implementing Mb using Sb Decimation Filter

Single-bit (Sb) decimation filters are the most preferred power efficient decimation architecture since they avoid the usage of large and power-consuming multi-bit multipliers [91]. On the other hand, multi-bit (Mb) decimation architectures widely exit for multi-bit $\Sigma\Delta$Ms as they have better SNR and better stability. Therefore, implementing an Mb decimation filter using a Sb decimation filter was investigated in this section in order to take over the Sb advantage on to the Mb architecture. The proposed architecture is given in Fig. 6.6. The proposed architecture was investigated using the case study given in section 6.2.1. Only 2-stage topology was investigated with multiple input bit-widths $W_i = 2, 3, 4, 5$. On the one hand, the maximum reduction in power consumption is around 10% for 2-bit input width, as shown in Fig. 6.7. On the other hand, the power consumption for 4-bit and

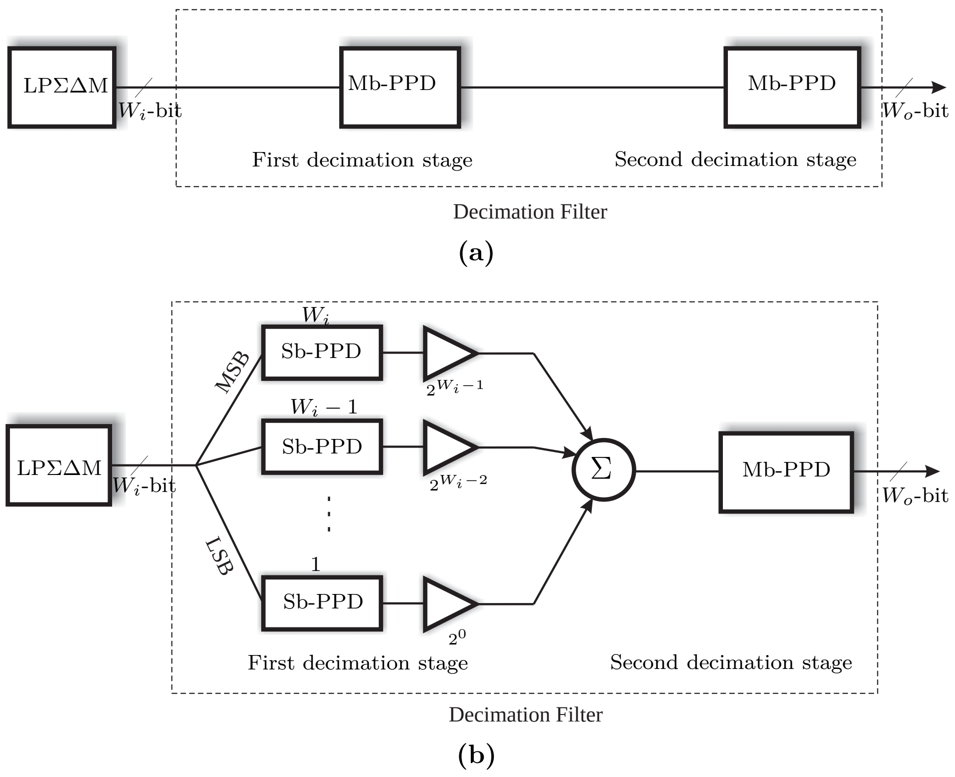


**Figure 6.6.:** Multi-bit (Mb) ΣΔM implementation using single-bit (Sb) architecture.

5-bit input has increased due to the increase in the number of commutators. It should be noted that, the commutator is the only sub-block within the polyphase decimation filter that operates at $f_s$ compared to remain sub-blocks that operates at $f_s/M$.

## 6.4. Summary

This work provides a set of design guidelines for FIR structures (transposed-form (TF) or direct-form (DF)) for a given set of decimation factor $M$ and filter order $N$. A selection criterion has been developed to choose between TF and DF for power efficient architectures, the results of which show less than 5% tolerance compared to gate-level simulations. DF shows a power efficient structure for single-bit decimation filters. The analysis results reveal that DF structures should be preferred at filter stages with single-bit or small multi-bit inputs and/or low decimation factors, whereas TF is advantageous in most other cases.
As the preferred architecture is a single-bit decimation filter, whereas the multi-bit architecture is commonly used as well. Implementing multi-bit decimation filters using single-bit has been proposed, which exhibit up to 10% reduction in power consumption compared to the conventional multi-bit implementation.

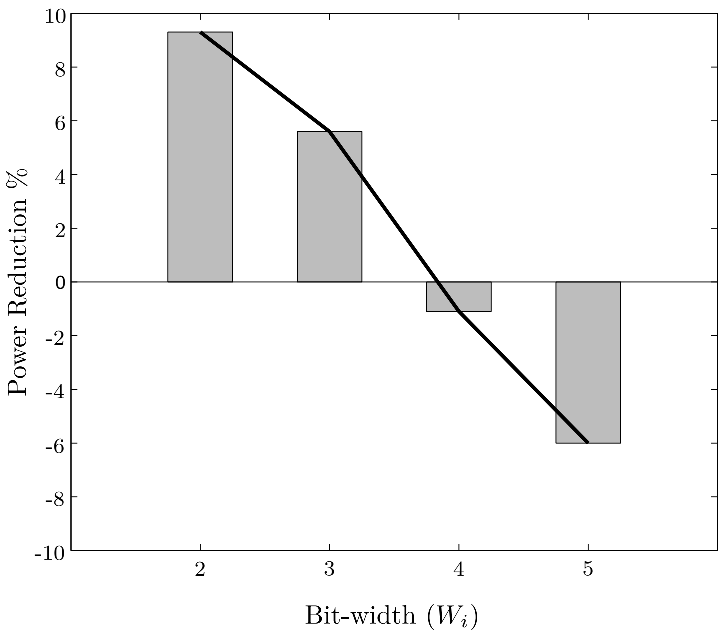


**Figure 6.7.:** Percentage in power reduction in Mb decimation filters employing Sb architecture.

# 7. Design and Implementation Procedure

## 7.1. Introduction

This chapter presents the developed Matlab toolbox and the proposed design procedure for multi-stage FIR decimation filter. The toolbox combines and involves novel methodologies developed in this work as well as several state-of-the-art research efforts. Moreover, this chapter describes the detailed generic VHDL RTL-based soft IP models developed in this work. Several VHDL IP models have been developed to verify the proposed power optimization methodologies. This work considers only the state-of-the-art structures and topologies such as; polyphase decomposition (PPD) and cascaded integrator comb (CIC).
The purpose in developing a consolidate toolbox is to: i) speed up the process of designing a multi-stage decimation filter, ii) consider as much as possible design parameters, and specifications, and iv) provide extended optimization potential.
Similar approaches were presented in literature, proposing toolboxes [94] and IPs [95], [96], [97]. The work in [94] presented a MATLAB toolbox for decimation filters for wireless communication applications only. However, the proposed toolbox is suitable for communication and control applications as well. While the work in [95] presented filter VHDL IP for FPGAs, the developed filter IPs in this work has been tested for FPGAs and ASICs as well. The VHDL models given in [96] has employed common sub-expression elimination algorithm (CSE) for canonic signed digit (CSD) coefficients. The developed VHDL models in this work employ CSE as well for binary and CSD representations.

## 7.2. Multi-Stage Decimation Toolbox

Several publications deal only with the design and optimization issues of decimation filters, whereas, few publications consider the development of toolboxes or automatic design tools for that purpose, such as [94], [98], [99].
The design procedure of a decimation filter involves translating the design specifications to implementation parameters. Furthermore, an extensive analysis and calculations are considered for optimizing the implementation parameters [94].
The aspiration for developing a consolidated design procedure for FIR decimation

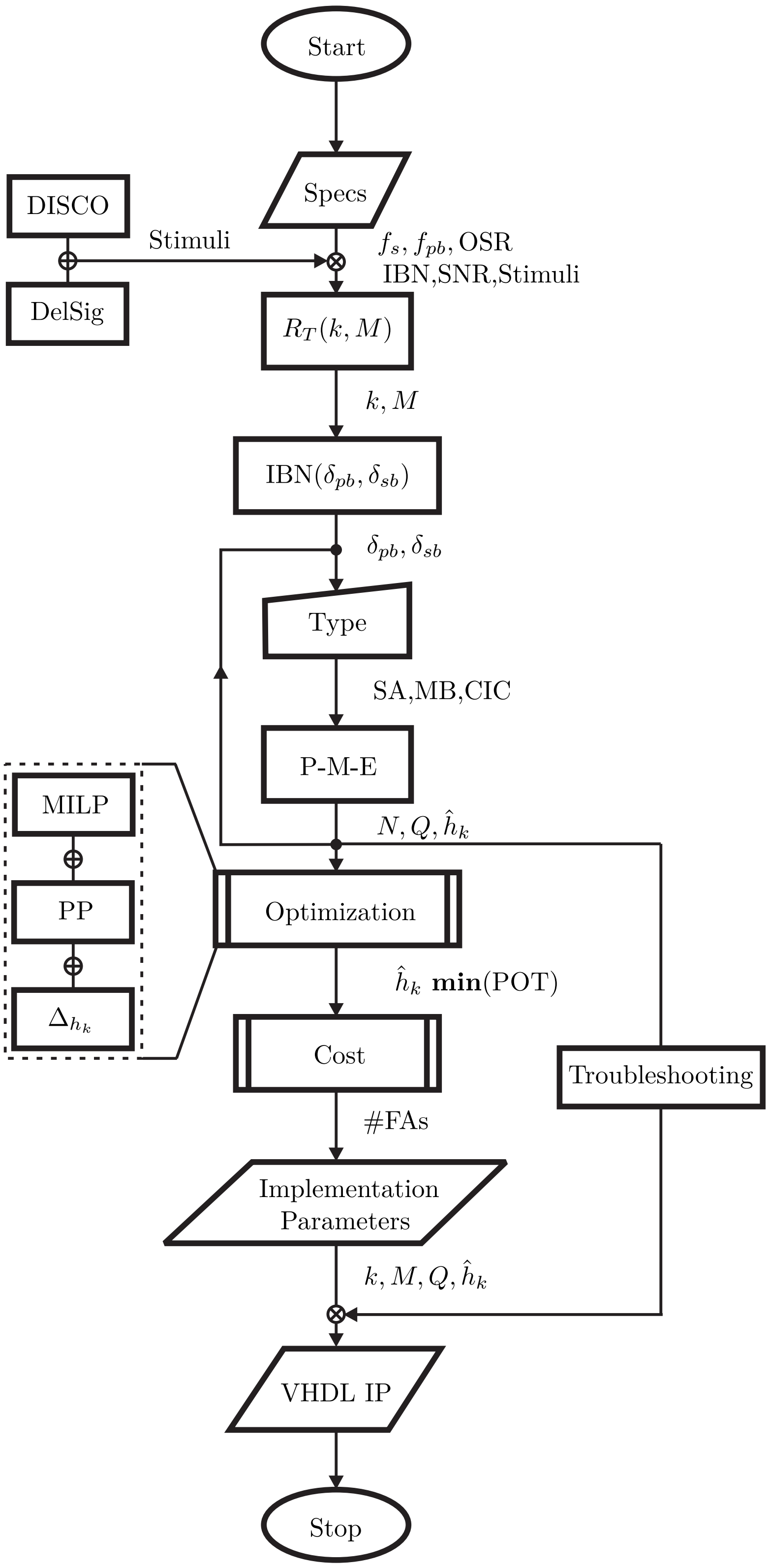


**Figure 7.1.:** MSD-toolbox flowchart.

filters, is to reduce the system design effort and speed up the optimization and the implementation process. This goal is achieved through involving considerable design specifications, bounded design constraints, definite performance metrics, and accurate simulations. These achievments were published in [1].
The Multi-Stage Decimation toolbox (MSD-toolbox) is developed in MATLAB language. The proposed dataflow is given by the flowchart given in Fig. 7.1.
The decimation filter system design starts with definite specifications such as the sampling frequency ($f_s$), oversampling ratio (OSR), passband frequency ($f_{pb}$), signal-to-noise ratio (SNR, regarded also as ADC resolution), in-band noise (IBN), and a stimuli bit-stream from sigma delta modulator. It should be noted that, the transition band-width ($\Delta f = (f_{sb} - f_{pb})/f_{sb}$) could be provided at the input specifications instead of the passband frequency. Compared to the previously developed toolboxes in [94], [98], [99], the IBN and the stimuli bit-stream patterns are the novel constraints considered in the MSD-toolbox, as shown in Fig. 7.1. The stimuli bit-stream permits accurate analysis for the intra decimation stages. Moreover, involving the stimuli affords performing spectral analysis in addition to the frequency domain analysis for the decimation filter. In addition, involving the acceptable tolerance in the IBN maintains additional flexibility for filter coefficient optimization. The stimuli bit-stream can be imported from DelSig-toolbox [5], DISCO-toolbox [19], stored data from practical measurements [100], and SimuLink models [101]. The MSD-toolbox has several routines and sub-programs for calculating implementation parameters and performance evaluation. The sub-programs and routines are based on state-of-the-art algorithms. The essential routines are:

- Calculating $k$ and $M$ ($R_T(k, M)$)
- Calculating $\delta_{pb}$ and $\delta_{sb}$ (IBN($\delta_{pb}, \delta_{sb}$))
- Calculating $h_k$ and $Q$ (P-M-E)
- Coefficient optimization (Optimization)
- Cost estimation (Cost)

The following subsections discuss each routine separately. An illustrative example is presented in section 4.4 for the proposed design procedure and the developed toolbox. Finally, an evaluation for the MSD-toolbox through a consistent comparison with the state-of-the-art toolbox [94] is given in section 6.5.
Figure 7.2 shows the lowpass filter (LPF) frequency response. The FIR filter transfer function is depicted in (7.1) [25].

$$H(z) = \sum_{i=0}^{N-1} h_{k_i} z^{-i} \tag{7.1}$$

where $N$ is the filter length, and $h_{k_i}$ is the filter coefficient set.

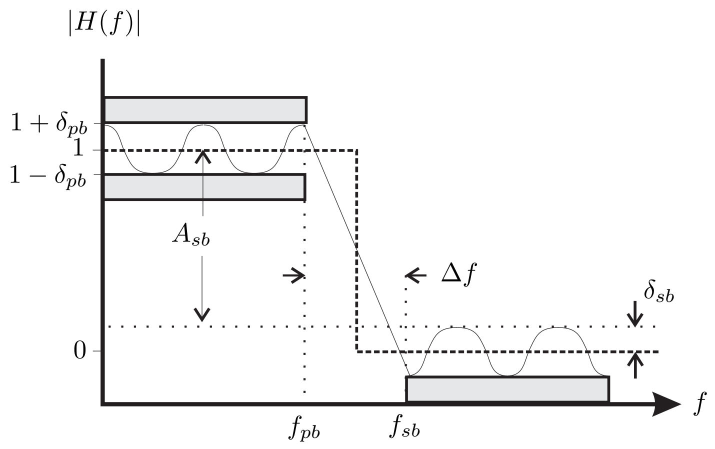


**Figure 7.2.:** Lowpass filter frequency response, where $f_{pb}$ is the passband frequency, $f_{sb}$ is the stopband frequency, $\Delta f$ is the transition bandwidth, $\delta_{pb}$ is the passband ripples, and $\delta_{sb}$ is the stopband ripple

### 7.2.1. $k$ and $M$ Calculations

The optimal number of decimation stages and the decimation factor at each stage is calculated for minimum computational effort ($R_T$), as given in (7.2). The problem is constrained for $k \in \{2, 3, 4\}$ [28] and $M_k = 2$ [28] for even $M$'s.

$$R_T = D_\infty f_s \sum_{i=1}^{k} \frac{M_i}{\left(\prod_{j=1}^{i} M_j\right)\left(1 - \frac{f_{sb}+f_{pb}}{f_s} \prod_{j=1}^{i} M_j\right)} \tag{7.2}$$

where $M$ is the overall decimation factor (equivalent to OSR), $f_s$ is the sampling frequency, $f_{pb}$ is the passband frequency, $f_{sb}$ is the stopband frequency, and $D_\infty$ is a function in passband ($\delta_{pb}$) and stopband ($\delta_{sb}$) ripples, as depicted in (7.3) [28]. $k$ is calculated for minimum $R_T$ at distinct values of $M$.

$$\begin{aligned} D_\infty = {} & \log_{10}\delta_{sb}[0.005309(\log_{10}\delta_{pb})^2 + 0.07114\log_{10}\delta_{pb} - 0.4761] \\ & -[0.00266(\log_{10}\delta_{pb})^2 + 0.5941\log_{10}\delta_{pb} + 0.4278] \end{aligned} \tag{7.3}$$

Detailed analysis for optimizing the number of decimation stages and the decimation factor for each stage is given in [28], [102], [103].

### 7.2.2. $\delta_{pb}$ and $\delta_{sb}$ Calculations

The passband ripples ($\delta_{pb}$) and stopband ripple ($\delta_{sb}$) are calculated using iterative simulations preserving a predefined acceptable tolerance in the in-band noise (IBN). This is done by designing a single stage filter and tuning the $\delta_{pb}$ and $\delta_{sb}$, by defining a certain range or a set of discrete values. The $\delta_{sb}$ remains the same for all the $k$-stages. While, the $\delta_{pb_i} = \delta_{pb}/k$ for stage $i$ [28].

### 7.2.3. $h_k$ and $Q$ Calculations

The exact number of decimation stages and the dedicated decimation factor for each stage were calculated in addition to the given design specifications, which sustain all the necessary parameters required for calculating the filter coefficients ($h_k$) for each decimation filter stage employing the Parks-McClellan Equiripple. Following the filter coefficient ($h_k$) calculation comes the quantization bit-width ($Q$) calculation step in order to compute the scaled filter coefficient set ($\hat{h}_k$).
Initially, a set of parameters is calculated for each stage, such as baseband frequency ($f_B$), passband frequency ($f_{pb}$), and stopband frequency ($f_{sb}$). The $f_B$ is calculated by (7.4).

$$f_B = \frac{f_s}{2\text{OSR}} \tag{7.4}$$

where OSR is the oversampling ratio and $f_s$ is the sampling frequency. The $f_{pb}$ and the $f_{sb}$ are calculated by Eqs (7.5) and (7.6), respectively. However, the intermediate stopband frequency ($f_{sb_i}$) is calculated by (7.7) [28].

$$f_{pb} = f_B(1 - \Delta f) \tag{7.5}$$

$$f_{sb} = f_B \tag{7.6}$$

$$f_{sb_i} = \frac{f_s}{\prod_{i=1}^{k} M_i} - f_B \tag{7.7}$$

FIR filter has distinct implementation types, such as standard FIR (FIR), half-band (HB), and multi-band (MB) types. The FIR filter implementation type affects the filter response and coefficients. The Parks-McClellan Equiripple (P-M-E) algorithm is used to obtain the FIR filter coefficients ($h_k$) for the predefined filter types except the CIC filter type. Subsequently, the required quantization bit-width ($Q$) is calculated iteratively preserving minimum mean error (ME) or minimum increase in the in-band noise (IBN). The FIR equiripple filter analysis can be found in [104].

#### 7.2.3.1. Half-band Filter

Half-band filter is a special class of filter which has been efficiently used for decimation stage [7]. Half-band (HB) filter is characterized such that its passband and stopband ripples are equivalent [7], as depicted in (7.8). Moreover, its passband and stopband frequencies are symmetrical around a quarter of the sampling frequency [7], as given in (7.9). Half-band filters have to have odd filter order [29]. HB filters are recommended to be used at the last decimation stage and for decimation factors of 2 [28].

$$\delta_{pb} = \delta_{sb} \tag{7.8}$$

$$f_{pb} + f_{sb} = f_s/4 \tag{7.9}$$

#### 7.2.3.2. Multi-band Filter

Aliasing is a sampling effect, as shown in Fig. 2.5. The bands of aliasing are well defined. As shown in Fig. 7.3.b, the spectrum of a decimated signal by $M$ has don't care regions ($\phi$) and aliased bands at $2\pi i/M$, where $i = 1, 2 \cdots M$. These alternating bands are defined by (7.10) which as well are considered as constraints for Equiripple filter design. The $\phi$ bands have influence on the filter design, through unconstrained bands [29]. Detailed analysis can be found in [29]. The reduction in filter order increases significantly with increasing the decimation factor, as given in Fig. 7.4. This can be explained from the decrease of the frequency span of the stopbands compared to the frequency span of the don't care bands [29]. Exploiting wider $\phi$ bands leads to a filter having cascaded integrator comb filter characteristic [29]. Moreover, the amplitude response at the $\phi$ bands has to be below a certain level to avoid noise amplification [29]. Figure 7.3.a and Fig. 7.3.d shows that band of interest before and after decimation, respectively. Figure 7.3.b shows the aliased bands into the in-band range. Additionally, the desired filter response with a consecutive doesn't care and stopbands is given in Fig. 7.3.c.

$$f_{sb} = [0, f_{pb}, \frac{ifs}{M_i} \pm f_{pb} \cdots f_s/2], \qquad i \in \{1 \to k\}\} \tag{7.10}$$

where $i, j \in \mathbb{N}$.

#### 7.2.3.3. CIC

One of the most efficient implementations of multi-rate change filter is the cascaded integrator comb (CIC) filters [29] with the transfer function given in (7.11).

$$H(z) = \frac{(1 - z^{-\alpha})^N}{(1 - z^{-1})^N} \tag{7.11}$$

where $N$ is the filter order (number of stages), and $\alpha$ is the product of decimation factor $M$ and differential delay $D$.
CIC filters are mostly used for large decimation factors [29]. There are three main design parameters for adjusting CIC filter response. On the one hand, the decimation factor ($M$) together with the differential delay ($D$) controls the CIC filter zeros, as shown in Figs. 7.5.a and 7.5.b, respectively. Practically, $D \in \{1, 2\}$. On the other hand, the filter order ($N$) affects the depth and the width of the stopband attenuation $A_{sb}$, as given in Fig. 7.5.c. Unluckily, increasing $N$ increases

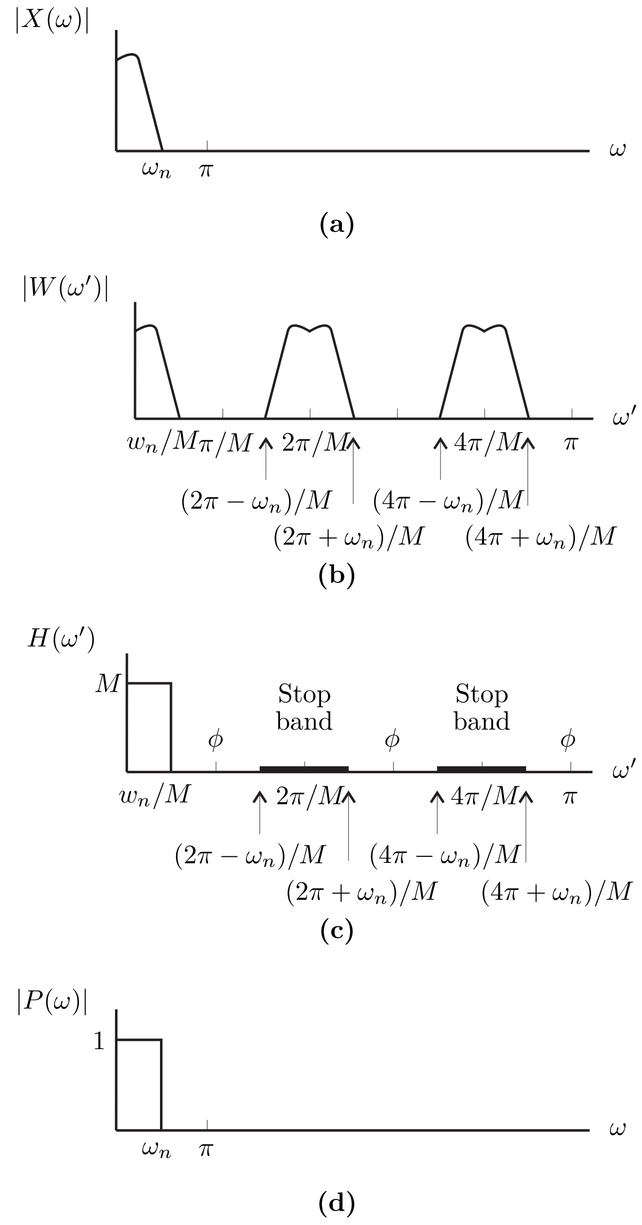


**Figure 7.3.:** Multiple stopband and don't care bands specifications within decimated signal spectral.

the drop in the passband, as exhibited in Fig. 7.5.d. However, the associated drop is compensated by using a compensation FIR filter stage. The amplitude response of an CIC filter is governed by (7.12) [105].

$$P(f) = \left( \frac{\sin \pi D f}{\sin \frac{\pi f}{M}} \right)^{2N} \tag{7.12}$$

where $f$ is the low sampling rate $f_s/M$, $D$ is the differential delay, $M$ is the decimation factor, and $N$ is the filter order. According to the practical recommendation declared in [105], $N \in \{1 \to 7\}$ and $D \in \{1, 2\}$. Since, $M$ and $f$ are fixed values

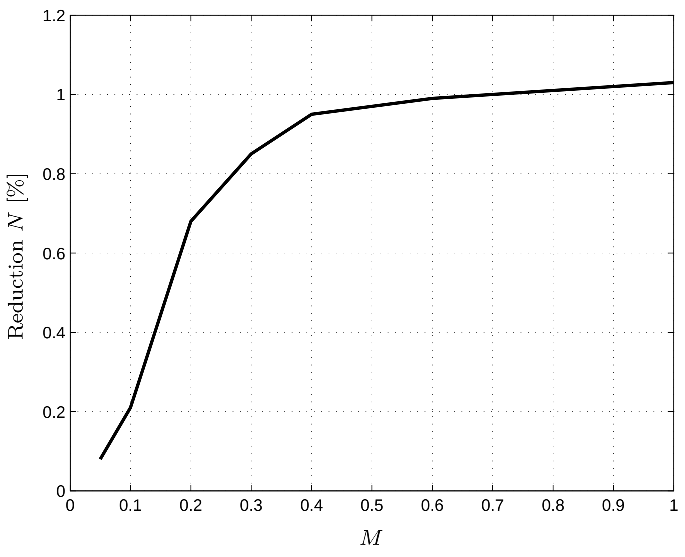


**Figure 7.4.:** Increase reduction percentage in filter order by using multi-band design as a function of $M$ (axis are normalized).

**Table 7.1.:** Compensation filters typical design parameters

| $N$ | $b$ |
|---|---|
| 1 | 2 |
| 2 | 1 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | -1 |
| 7 | -2 |

in (7.12), the same equation has been used to tune $N$ for minimum drop in amplitude response within the passband [105].
The compensation filter transfer function is given by (7.13) [106].

$$G(z^M) = B[1 + Az^{-M} + z^{-2M}] \tag{7.13}$$

where

$$\begin{aligned} B &= -2^{-(b+2)} \\ A &= -[2^{(b+2)} + 2] \end{aligned} \tag{7.14}$$

where b is an integer value, and its corresponding value for a defined $N$ is given in Table 7.1. Figure 7.6.a and b shows the CIC filter response (black solid line), the compensation filter response (dashed line), and the cascaded CIC and compensation filter response (solid Gray line). Further, a few LSBs are truncated at each stage to overcome the aggressive internal bit growth. The number of truncated

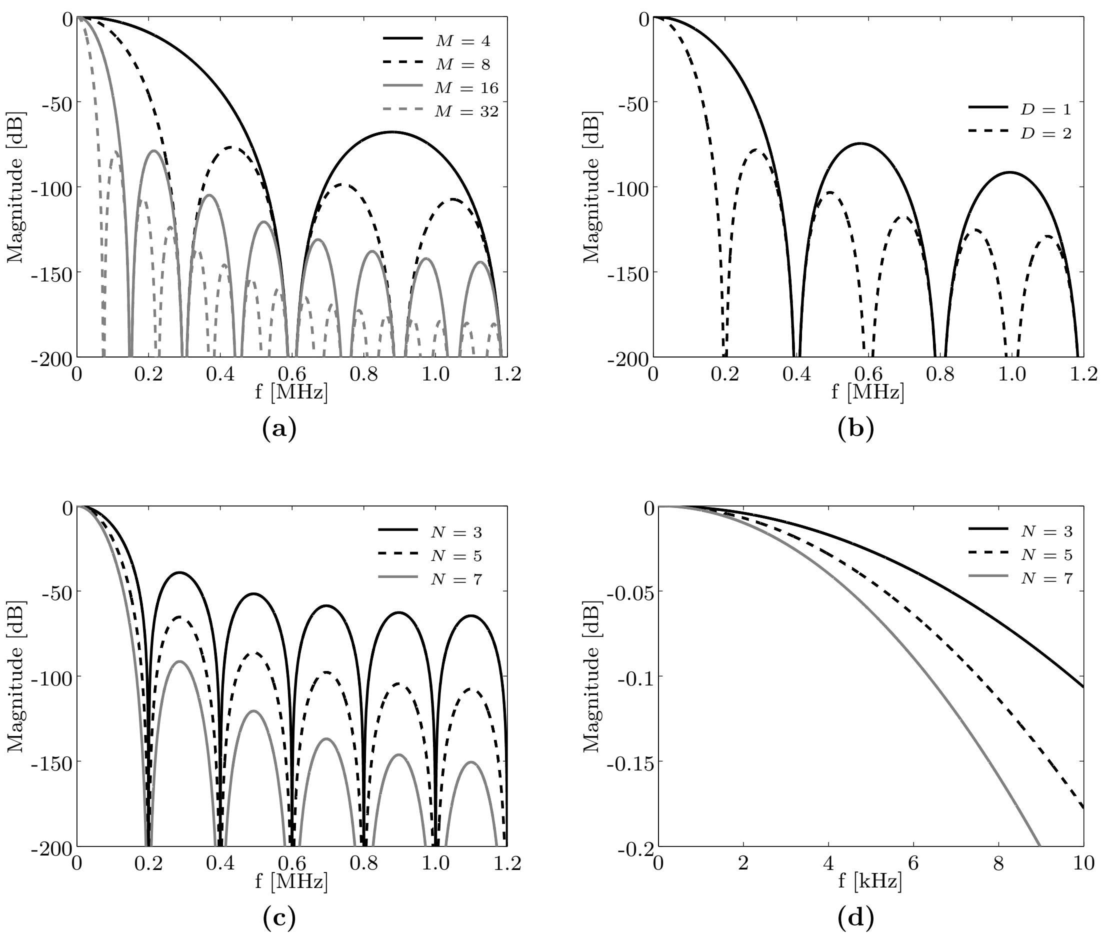


**Figure 7.5.:** CIC design parameters (a) $M$ (b) $D$ (c) $N$ (d) drop on passband for $f_s = 2.4$ MHz, $N = 6$, and $D = 2$, where $N$ is the filter order, $D$ is the differential delay, and $M$ is the decimation factor.

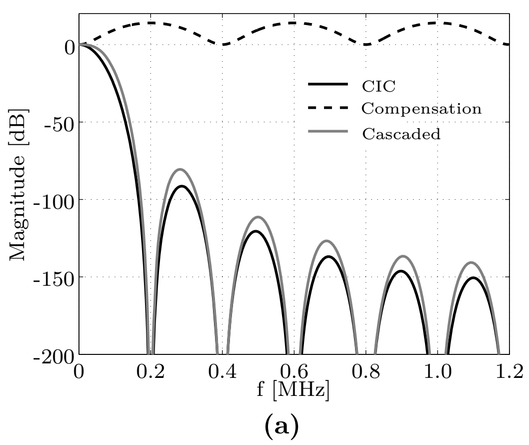
bits is derived from (7.15) [105].

$$B_j = \lfloor -log_2(F_j) + log_2(\sigma_{T_{2N+1}}) + \frac{1}{2}log_2(\frac{6}{N}) \rfloor \tag{7.15}$$

for $j = 1, 2, 3 \cdots, 2N$, where $F_j$ is the variance error gain, $\sigma_{T_{2N+1}}$ is the total variance, $N$ is the filter order. The variance error gain is given by (7.16) [105].

$$F_j{}^2 = \begin{cases} \sum\limits_k h_j^2(k), & j = 1, 2, \cdots, 2N \\ 1, & j = 2N+1 \end{cases} \tag{7.16}$$

where $k = 0, 1, 2 \cdots 2N + 1 - j$ ($j$ represents the stage number), and $h_j(k)$ is the impulse response coefficients shown by (7.17), its derivation can be found



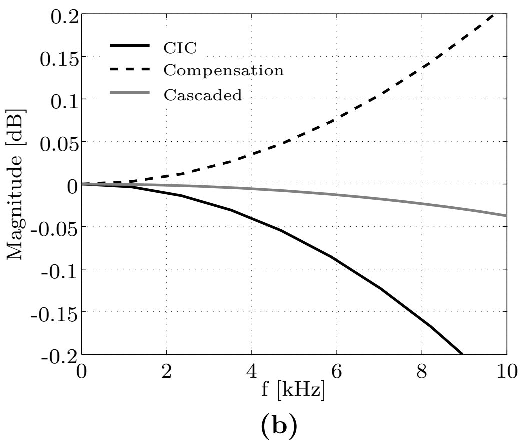


**Figure 7.6.:** CIC compensation filter (a) response (b) zoom-in, for $f_s = 2.4$ MHz, $N = 7$, $D = 2$, and $M = 6$, where $N$ is the filter order, $D$ is the differential delay, and $M$ is the decimation factor.

in [105].

$$h_j(k) = \begin{cases} \sum_{i=0}^{\lfloor k/MD \rfloor} (-1)^i \begin{pmatrix} N \\ i \end{pmatrix} \begin{pmatrix} N-j+k-MDi \\ k-MDi \end{pmatrix}, & j = 1, 2, \cdots, N \\ (-1)^i \begin{pmatrix} 2N+1-j \\ k \end{pmatrix}, & j = N+1, \cdots, 2N \end{cases} \tag{7.17}$$

where $N$ is the filter order, $D$ is the differential delay, and $M$ is the decimation factor. While, the total variance is calculated as given by (7.18) and (7.19).

$$\sigma_{T_j}{}^2 = \sigma_j{}^2 F_j{}^2 \tag{7.18}$$

$$\sigma_j{}^2 = \frac{1}{12} E_j{}^2 \tag{7.19}$$

where $E_j$ is a uniform distribution error at stage $j$, as stated by (7.20).

$$E_j = \begin{cases} 0, & \text{if no truncation} \\ 2^{B_j} \end{cases} \tag{7.20}$$

where $B_j$ is the number of truncated LSB's at stage $j$.

#### 7.2.3.4. IBN Calculation

The in-band noise (IBN) is calculated analytically as depicted by (7.21) [16], or calculated from the spectral density as shown in Fig. 7.7. The analytical calculation of the IBN is an ideal estimation for the expected IBN for a certain ΣΔM, and it can not be used for a decimation filter. However, the calculated IBN from the spectral density is more accurate since it considers more practical and simulation

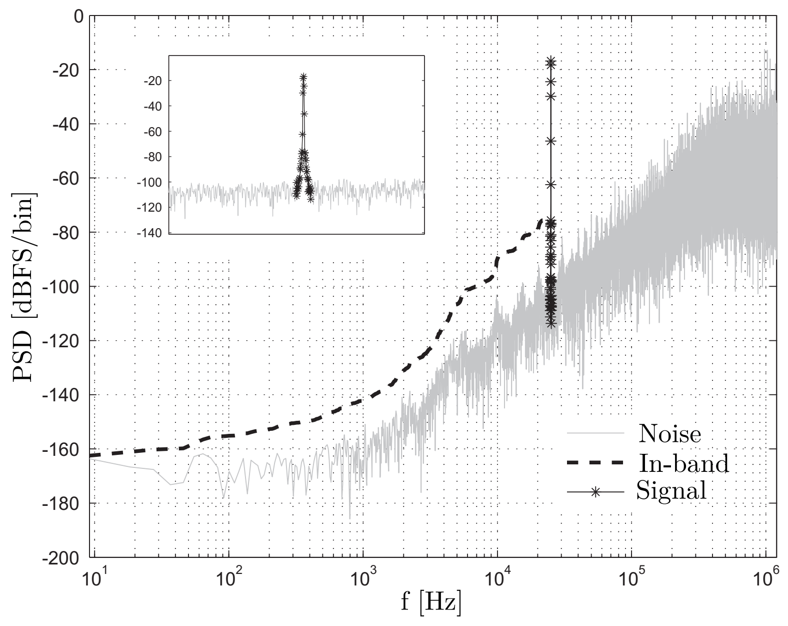


**Figure 7.7.:** IBN calculation from stimuli bit-stream for $f_s = 2.4$ MHz, OSR$= 48$, and $f_{signal} =$ 25 kHz.

effects. This criterion allows the calculation of the IBN for the intra decimation stages in addition to the overall IBN after decimation. Therefore, it is the default setting for the MSD-toolbox.

$$\mathrm{IBN} = \left(\frac{\Delta}{2}\right)^2 \frac{1}{3\pi(2N+1)} \left(\frac{\pi}{\mathrm{OSR}}\right)^{2N+1} \left(k_q \prod_{i=1}^{N} k_i\right)^{-2} \tag{7.21}$$

$N$ is the $\Sigma\Delta$M order, OSR is the oversampling ratio, $k_i$ is the $\Sigma\Delta$M scaling coefficients, $k_q$ is the quantizer gain (used within the $\Sigma\Delta$M), and $\Delta$ is the quantizer LSB which calculated as given in (7.22).

$$\Delta = \frac{\mathrm{FS}}{2^B - 1} \tag{7.22}$$

where $B$ is the quantizer bit-width, and FS is the quantizer full-scale. Moreover, the IBN can be calculated from the spectral density as shown in Fig. 7.7. The criterion for calculating the IBN from the spectral density was proposed in [101] and verified in [16] and [19]. The calculation of the IBN is performed by removing the signal peak from the spectral and then suming all the noise in the in-band range $[0 \rightarrow f_B]$, as shown in Fig. 7.7 by the zoom-in. If the $k_i$ and $k_q$ are not available in the input specifications, the last term in (7.21) is set to one.

### 7.2.4. Coefficient Optimization

FIR filter coefficients are calculated using the Parks-McClellan Equiripple algorithm in an infinite floating point format from Matlab. The coefficients are then quantized and scaled in order to be represented in finite discrete fixed point representation. Practically, each discrete coefficient is translated to a set of binary bits; zeros and ones. The number of none-zero bits in each coefficient exhibits a burden in the hardware implementation, which consequently, consumes more power. Therefore, coefficient optimization tends to reduce the number of none-zero bits at each filter coefficient (if possible). Several methodologies and algorithms have been developed and used for this purpose in literature [40], [45], [49], [65], [63], [69], [70], [72].

### 7.2.5. Cost Estimation

The MSD-toolbox provides a complete flow for the decimation filter design, optimization, and implementation. Since, the toolbox delivers the VHDL description of the designed decimation filter, it has broad awareness of the design hierarchy and bit-level details. The number of full-adders (FAs) involved in the multipliers and multi-operand adders is estimated and plotted in bar charts for rough estimation of the power consumption. The number of FAs is calculated from the output bit-width ($W_o$) of a multiplier or a multi-operand adder. The number of FA depends on the topology of implementation. The multi-operand adder is implemented in balanced tree structure, as depicted in Fig. 7.8.a with a ripple carry adder (RCA) structure for the adder cells. It should be noted that, the RCA is the adequate approach for low power implementation [107]. The constant multiplier is implemented in shift-add structure, as depicted in Fig. 7.8.b with a ripple carry adder (RCA) structure for the adder cells. A detailed discussion on the implementation of constant multiplier and multi-operand adder is given later in this chapter on VHDL IPs sections 8.2.2 and 8.2.4, respectively.
The number of FAs in multi-operand adder tree is determined from the number of adder cells and the output bit-width of each adder cell. As an example, consider a 7-input multi-operand adder, as shown in Fig. 7.8.a on the left. It requires 6 adder cells on 3 hierarchical levels. Conventionally, all adder cells have the same output bit-width ($W_o$), which is equivalent to the maximum bit-width. Nevertheless, for an efficient power optimized implementation the developed VHDL IP models support a variable bit-width for each internal adder cell, within a multi-operand adder or a multiplier block. The number of full-adders (#FAs) is calculated by (7.23).

$$\#\text{FAs} = \sum_{i=1}^{n} W_{o_i} \tag{7.23}$$

where n is the number of adder cells, $W_{o_i}$ is the output bit-width for adder cell $i$.
The #FAs involved in a constant multiplier is calculated by

$$\#\text{FAs} = \sum_{j=1}^{m} (W_i + \mathbf{max}(\ll S_j, \ll S_{j+1}) + 1) + \sum_{i=m+1}^{n} (W_{o_i}) \tag{7.24}$$

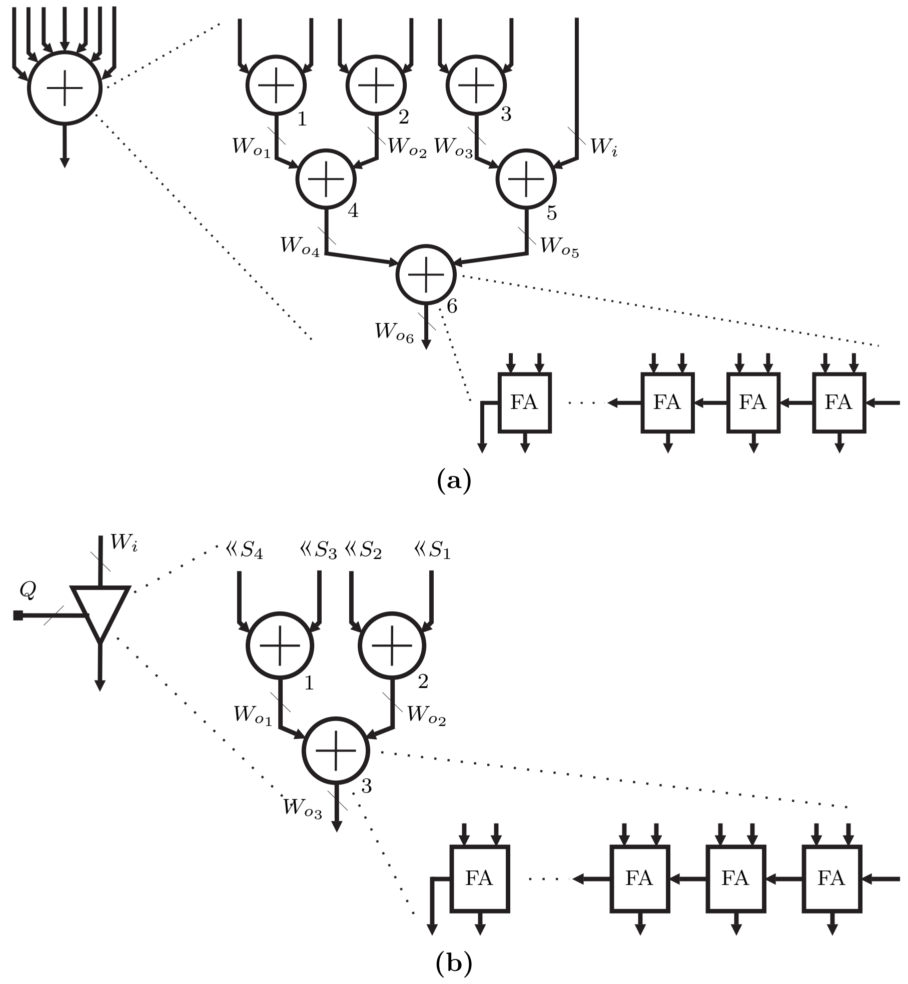


**Figure 7.8.:** Architectural cost estimation for (a) multi-operand adder (b) multiplier, where FA is full-adder, $W_i$ is the input bit-width, $W_o$ is the output bit-width, «$S_j$ is the number of shift left bits.

where $m$ is the number of input adder cells (half number of none-zero bits), $n$ is the number of adder cells, $W_i$ is the multiplier input bit-width, «$S_j$ is the number of shift left bits bit $j$, $W_{o_i}$ is the output bit-width for adder cell $i$, and the 1 is added to avoid overflow. Illustratively, consider a numerical example for adding 7 values using a multi-operand adder. The internal tree arrangement of the adder cells is depicted in Fig. 7.8.a (middle), where each adder cell is implemented in RCA as shown in Fig. 7.8.a (right). The input values are given in the first column of Table 7.2. The output bit-widths for each adder cell at the 3 levels are given in the last three rows of Table 7.2. The expected #FAs is 45. However, for a conventional implementation a 54 FA is needed for the same multi-operand adder. This reveals that about 20% saving in the number of FA for variable bit-width adder cells. It should be noted that, an extra 1-bit is added to avoid addition overflow. Consider a numerical coefficient of 240 for the constant multiplier for illustration, as well. The internal architecture of the shift-and-add multiplier is given in Fig. 7.8.b (middle), with the number of shift left bits ($S_j$) depicted at each

**Table 7.2.:** Multi-operand adder cost estimation

| Value | Binary | $W_o$ 1st Level | $W_o$ 2nd Level | $W_o$ 3rd Level |
|---|---|---|---|---|
| 64 | 1000000 | 8 | 8 | 9 |
| 56 | 111000 | | | |
| 13 | 1101 | 7 | | |
| 96 | 1100000 | | | |
| 25 | 11001 | 5 | 8 | |
| 2 | 10 | | | |
| 240 | 11110000 | | | |

input by $S_1, S_2, S_3, S_4$, respectively. Equation (7.25) gives the binary representation of the constant coefficient 240, where $2^{'s}$ power represents the shift weights and the $2^{'s}$ multiplicands corresponds the binary representation.

$$240 = 2^7 \times 1 + 2^6 \times 1 + 2^5 \times 1 + 2^4 \times 1 + 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 0 \quad (7.25)$$

Equation (7.25) reveals to $S_4 = 7, S_3 = 6, S_2 = 5,$ and $S_1 = 4$. For a 3-bit input multiplier ($W_i = 3$), the calculated number of full-adders is 32 FA by using (7.24), with 11 FA for the first adder cell, 10 FAs for the second adder cell, and 11 FA for the third adder cell. A complete procedure is illustrated through a detailed design example in section 6.4, which demonstrates the advantages of precise estimation for the number of gates involved in multi-operand adder and multiplier implementation.
The dynamic power is correlated to the load driven by a gate, as given in chapter 3. Moreover, the static power is proportional to the area consumed by a gate. Therefore, the estimated number of full-adder gates is used to model the power consumption in FIR filters [107].

## 7.3. Troubleshooting and Verification

Verification has become necessary with the increase of the design complexity. However, defining the source of an error, or debugging a code for a design is becoming as complex as verification with the increase of design integrity and complexity. Therefore, a form of problem identification and problem solving is necessary for the design procedure, which is known as troubleshooting. Troubleshooting defines the source of a problem, while, verification assures the functionality of a design. Recently, hardware verification languages (HVL) are booming in the market of VLSI design. The developed toolbox and the proposed design procedure do not utilize HVL, nevertheless, they provide a bit-level manipulation for the designs in both MATLAB and VHDL. Assertion statement is used in VHDL codes and structure data type is used within the MATLAB functions. Troubleshooting is a form of problem identification and problem solving. The proposed design procedure and the developed toolbox provide a troubleshooting criterion correlating MATLAB and VHDL together. An illustrative demonstration for the proposed criterion, a straightforward example for a CIC filter with order of 3, decimation ratio of 1 (no downsampling), and differential delay of 1 is shown in Fig. 7.9. Figure 7.9.a drives the analytical intra bit-level activity of the designed CIC filter. Symbol (I*i*) is the input for integrator *i*, (ID*i*) is the delayed integrator output. The same is for comb (C*i*) and comb delay (CD*i*). Each row represents the bit activity during one clock cycle. On the other hand, each column represents the activity during consecutive clock cycles. The consecutive clock cycles are denoted by column *t*. The arrows depicted in the Fig. 7.9.a represents the delayed outputs. The input bit-stream stimulus is depicted by the second column (I1). The MSD-toolbox supplies the intra bit-level activity of the CIC filter in a tabular format exported in a text file as given in Fig. 7.9.b. This is the reference pattern for the filter activity. The simulated VHDL model output is given in Fig. 7.9.c. By correlating the MATLAB bit-level activity with the VHDL simulation, it became easier to allocate the source of the error when it occurs.

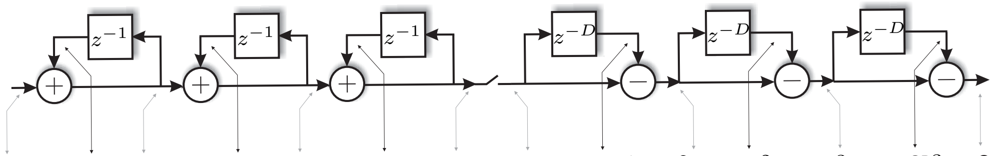
## 7.4. Design Example

An illustrative design example is given in this section. The design specification is given in Table 7.3 for a $2^{nd}$ order $\Sigma\Delta$M with 1-bit quantizer simulated in SIMULINK model from [101] for storing a stimuli file. The whole script took 66.2 seconds on 1-GB RAM and 1.6 GHz processor PC. The result of each of the five routines is depicted by a figure or a table in sequel.

**Table 7.3.:** Design parameters

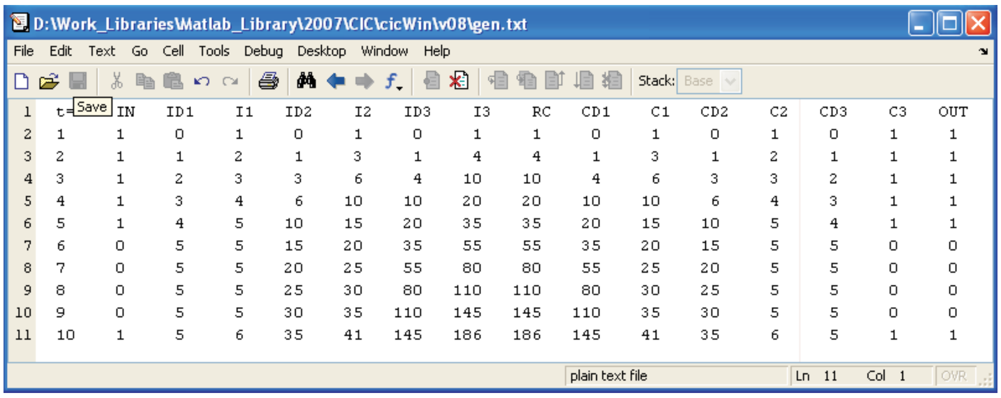
| Parameter | Value |
|---|---|
| $f_s$ | 2.4 MHz |
| OSR | 48 |
| $f_{signal}$ | 5 kHz |
| Stimuli | ✓ |

The result from calculating $k$ and $M$ ($R_T(k, M)$) routine is displayed in Fig. 7.10. The bar chart shows the computational effort for the two, three, and four cascaded decimation stages. For three stages, there two alternatives for the arrangement of the decimation filters, either [12 2 2] or [6 4 2]. The [6 4 2] arrangement exhibits the minimum computational effort. The results of this step is $k = 3$ and $M = [6, 4, 2]$. Calculating $\delta_{pb}$ and $\delta_{sb}$ (IBN($\delta_{pb}, \delta_{sb}$)) the routine exports the results in a form of a table as given in Table 7.4. The last column represents the penalty in the IBN. The last row is chosen manually for defining the filter specifications according to certain acceptable penalty in the IBN.
The multi-band filter architecture has been chosen for Parks-McClellan Equiripple algorithm. Further, the last stage is set to be half-band filter for decimation factor of two. Calculating $h_k$ and $Q$ (P-M-E) exports the filter coefficients ($h_{k_1}, h_{k_2}, h_{k_3}$)



| t | I1 | ID1 | I2 | ID2 | I3 | ID3 | $M$ | C1 | CD1 | C2 | CD2 | C3 | CD3 | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 2 | 1 | 1 | 2 | 1 | 3 | 1 | 4 | 4 | 1 | 3 | 1 | 2 | 1 | 1 |
| 3 | 1 | 2 | 3 | 3 | 6 | 4 | 10 | 10 | 4 | 6 | 3 | 3 | 2 | 1 |
| 4 | 1 | 3 | 4 | 6 | 10 | 10 | 20 | 20 | 10 | 10 | 6 | 4 | 3 | 1 |
| 5 | 1 | 4 | 5 | 10 | 15 | 20 | 35 | 35 | 20 | 15 | 10 | 5 | 4 | 1 |
| 6 | 0 | 5 | 5 | 15 | 20 | 35 | 55 | 55 | 35 | 20 | 15 | 5 | 5 | 0 |
| 7 | 0 | 5 | 5 | 20 | 25 | 55 | 80 | 80 | 55 | 25 | 20 | 5 | 5 | 0 |
| 8 | 0 | 5 | 5 | 25 | 30 | 80 | 110 | 110 | 80 | 30 | 25 | 5 | 5 | 0 |
| 9 | 0 | 5 | 5 | 30 | 35 | 110 | 145 | 145 | 110 | 35 | 30 | 5 | 5 | 0 |
| 10 | 1 | 5 | 6 | 35 | 41 | 145 | 186 | 186 | 145 | 41 | 35 | 6 | 5 | 1 |

**(a)**

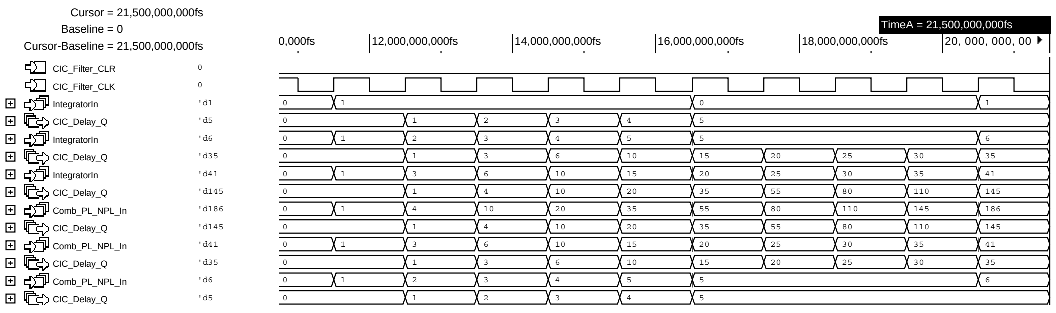


D:\Work_Libraries\Matlab_Library\2007\CIC\cicWin\v08\gen.txt

| t= | IN | ID1 | I1 | ID2 | I2 | ID3 | I3 | RC | CD1 | C1 | CD2 | C2 | CD3 | C3 | OUT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 2 | 1 | 1 | 2 | 1 | 3 | 1 | 4 | 4 | 1 | 3 | 1 | 2 | 1 | 1 | 1 |
| 3 | 1 | 2 | 3 | 3 | 6 | 4 | 10 | 10 | 4 | 6 | 3 | 3 | 2 | 1 | 1 |
| 4 | 1 | 3 | 4 | 6 | 10 | 10 | 20 | 20 | 10 | 10 | 6 | 4 | 3 | 1 | 1 |
| 5 | 1 | 4 | 5 | 10 | 15 | 20 | 35 | 35 | 20 | 15 | 10 | 5 | 4 | 1 | 1 |
| 6 | 0 | 5 | 5 | 15 | 20 | 35 | 55 | 55 | 35 | 20 | 15 | 5 | 5 | 0 | 0 |
| 7 | 0 | 5 | 5 | 20 | 25 | 55 | 80 | 80 | 55 | 25 | 20 | 5 | 5 | 0 | 0 |
| 8 | 0 | 5 | 5 | 25 | 30 | 80 | 110 | 110 | 80 | 30 | 25 | 5 | 5 | 0 | 0 |
| 9 | 0 | 5 | 5 | 30 | 35 | 110 | 145 | 145 | 110 | 35 | 30 | 5 | 5 | 0 | 0 |
| 10 | 1 | 5 | 6 | 35 | 41 | 145 | 186 | 186 | 145 | 41 | 35 | 6 | 5 | 1 | 1 |

plain text file    Ln 11    Col 1    OVR

**(b)**



**(c)**

**Figure 7.9.:** Troubleshooting and verification (a) analytical (b) MATLAB (c) VHDL for a CIC filter with $N = 3$, $M = 1$, and $D = 1$, where I for integrator, ID for integrator delay, C for comb, and CD for comb delay

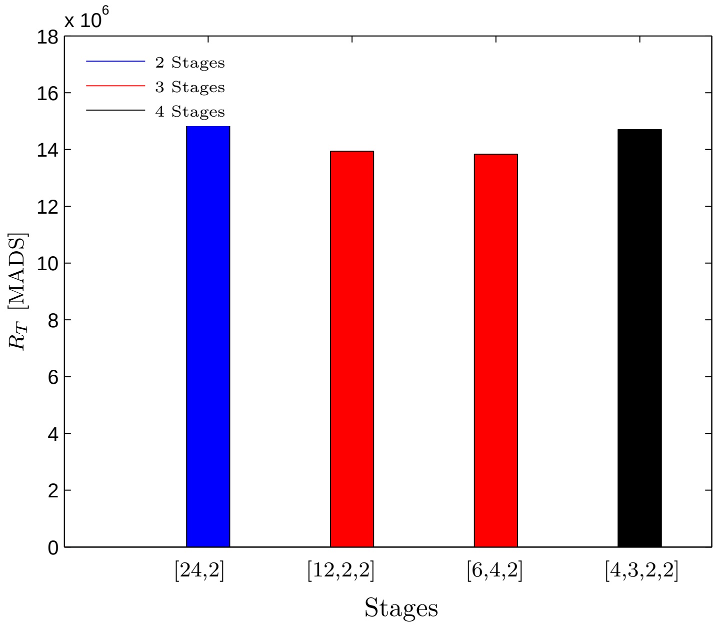


**Figure 7.10.:** Estimated computation effort ($R_T(k, M)$) using various multistage decimation filter.

**Table 7.4.:** Tuning ripples

| rp | rc | IBN ±% |
|---|---|---|
| 0.00005 | 0.00010 | -1.9753 |
| 0.00005 | 0.00001 | -2.0042 |
| 0.00005 | 0.00010 | -1.1514 |
| 0.00001 | 0.00010 | -2.0042 |
| 0.00001 | 0.00005 | -1.3065 |
| 0.00001 | 0.00010 | -2.0662 |
| 0.00010 | 0.00010 | -1.1514 |
| 0.00010 | 0.00001 | -2.0662 |
| 0.00050 | 0.00001 | -1.1514 |

and their responses as shown in Fig. 7.11 for the three cascaded stages, with filter orders of $N_1 = 36$, $N_2 = 39$, and $N_3 = 19$, respectively. Moreover, the IBN after and before decimation is plotted and calculated from the graph, as given in Fig. 7.12. There is less than 2 dB increase in the IBN after using the designed decimation filter. Defining a suitable quantization bit-width ($Q$) for each stage is done iteratively, through defining a certain set of $Q$'s. Table 7.5 shows the results for the examined $Q$'s and the corresponding effect on in-band noise (IBN), mean error (ME), and mean square error (MSE) in filter response. The second row is chosen for the implementation parameters.
Cost estimation (Cost) involves the estimation of the number of full-adders (#FAs) in multiplier and multi-operand adder blocks within the decimation filter. Figure 7.13 shows the estimated #FAs in multipliers and multi-operand adders, re-

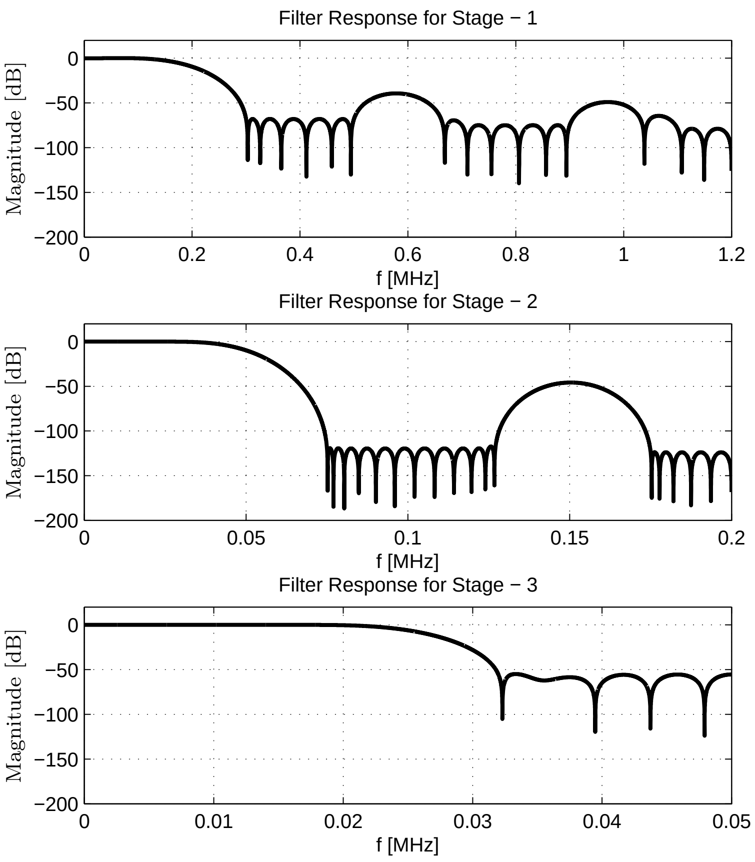


**Figure 7.11.:** Filter responses.

**Table 7.5.:** Quantization bit-width tuning

| $Q_1$ | $Q_2$ | $Q_3$ | IBN | ME | MSE |
|---|---|---|---|---|---|
| 18 | 12 | 10 | -0.2598 | 0.6e-6 | 0.9e-12 |
| 16 | 12 | 10 | -0.2592 | 3.2e-6 | 16e-12 |
| 16 | 10 | 5 | -0.6380 | 3.2e-6 | 16e-12 |
| 14 | 10 | 5 | -0.6912 | 15e-6 | 320e-12 |
| 12 | 9 | 5 | -1.9123 | 45e-6 | 3.4e-9 |

spectively. The number of FAs in multiplier is related to the implementation and the output bit-width. The estimation of the #FAs considers shift-and-add implementation for the constant multipliers. The estimated #FAs per decimation stage is given in Fig 7.13.a to 7.13.c. The number of FAs for multi-operand adder is related to the output bit-width per internal adder (two input adder). So, for a multi-operand adder with 6 inputs, there are 5 internal adders. Figure 7.13.e to Fig. 7.13.g shows the estimated #FAs per decimation stage, where the $x$-axis represents the number of multi-operand adders per stage. As an example, the second

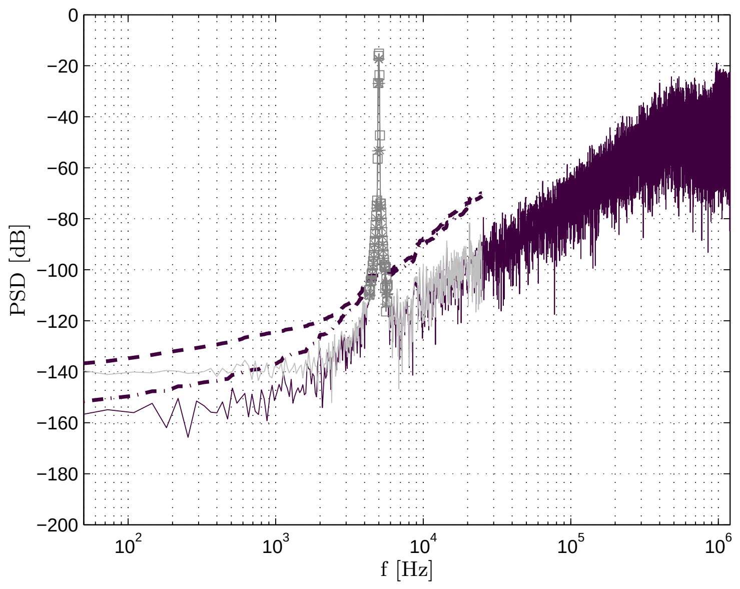


**Figure 7.12.:** PSD before and after decimation.

stage has $N_1 = 39$ and $M_1 = 4$ leads to $\lceil N/M \rceil = 10$ multi-operand adder. Figure 7.13.h reveals the estimated power consumption in the decimation stages, i.e. the second stage is estimated to consumes about 50% of the complete decimation stage compared to 30% by the first stage and 20% by the third stage.

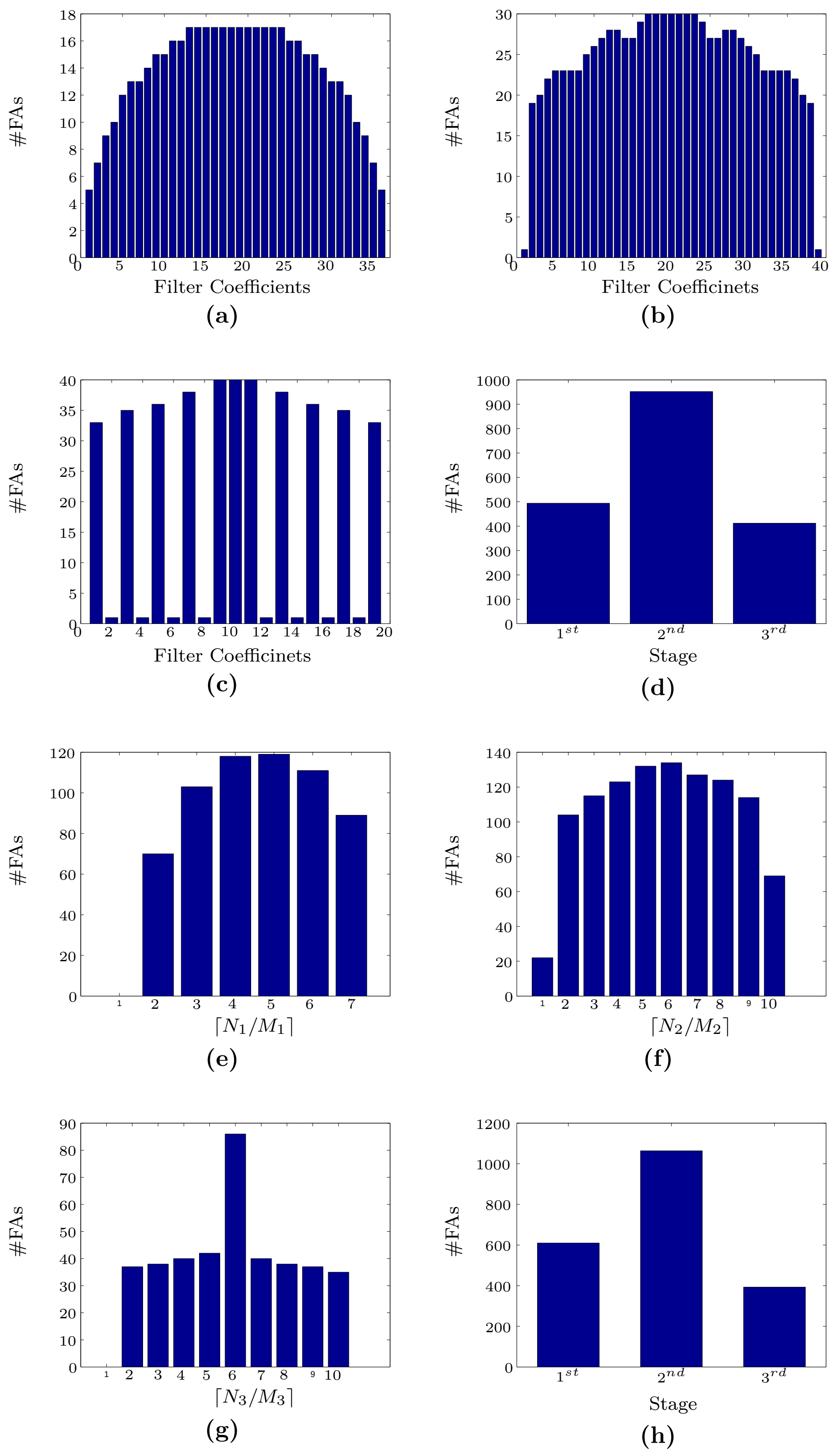


**Figure 7.13.:** Estimated number of FAs for multipliers in (a) $1^{st}$ stage (b) $2^{nd}$ stage (c) $3^{rd}$ stage (d) total, estimated number of FAs for multi-operand adders in (e) $1^{st}$ stage (f) $2^{nd}$ stage (g) $3^{rd}$ stage (h) total.

## 7.5. MSD-toolbox Evaluation

The MSD-toolbox was compared to the state-of-the-art decimation toolbox proposed in [94]. Multi-standard wireless communication specifications have been considered for evaluating the proposed toolbox and to have consistent comparison with the toolbox from [94]. The multi-standard decimation filter specification is given in Table 7.6. Table 7.7 shows the decimation filter implementation parameters generated from MSD-toolbox and the state-of-the-art toolbox [94]. The implementation considers several filter architectures such as: cascaded integrator comb (CIC), multi-band (MB), half-band (HB), and standard FIR filter. It has to be noted that, the compensation filter for CIC is employed as FIR filter and its length is denoted in the table by the plus sign, e.g. in the last row in Table 7.7. The results presented in Table 7.7 reveal the considerable saving in the filter lengths using the developed toolbox for some of the cases. The discrepancy between the filter lengths between the developed and the state-of-the-art toolbox even though employing the same architecture is due to considering the IBN during the analysis for the MSD-toolbox. The penalty in the IBN was limited to half bit, i.e., 3 dB, during the design process. On the one hand, the decimation filter for the WCDMA and WiMAX designed by the state-of-the-art toolbox guaranties better IBN compared to the filter designed by MSD$^2$ with larger filter length. On the other hand, the decimation filter for GSM and WLAN designed by the MSD-toolbox guaranties the same IBN with smaller filter lengths compared to the toolbox presented in [94]. The MSD-toolbox is suitable for generic decimation filters, however, the state-of-the-art toolbox is limited to the wireless communication standards only.

**Table 7.6.:** Decimation filter design parameters for multi-standard specifications

| Standard | OSR | $f_s$ MHz | $f_{pb}$ MHz | $f_{sb}$ MHz | $\delta_{pb}$ | $A_{sb}$ dB |
|---|---|---|---|---|---|---|
| GSM | 128 | 34.667 | 0.08 | 0.1 | 0.0058 | 65 |
| WCDMA | 16 | 61.44 | 2 | 2.5 | 0.028 | 55 |
| WLANa | 8 | 96 | 8 | 10 | 0.028 | 44 |
| WiMAX | 8 | 133.632 | 8 | 10 | 0.028 | 39 |

**Table 7.7.:** Decimation filter implementation for multi-standard specifications

| Standard | Filter Structure | | | Decimation Factor | | | Filter Length | | |
|---|---|---|---|---|---|---|---|---|---|
| | [94] | MSD[1] | MSD[2] | [94] | MSD[1] | MSD[2] | [94] | MSD[1] | MSD[2] |
| | CIC | MB | CIC | 32 | 16 | 16 | 3 | 60 | 3 |
| GSM | HB | MB | MB | 2 | 4 | 4 | 11 | 24 | 23 |
| | FIR | HB | HB | 2 | 2 | 2 | 101 | 19 | 19+7 |
| Total | | | | | | | 115 | 103 | 52 |
| | CIC | MB | CIC | 4 | 8 | 8 | 4 | 55 | 3 |
| WCDMA | HB | HB | FIR | 2 | 2 | 2 | 19 | 19 | 35+7 |
| | FIR | - | - | 2 | - | - | 48 | - | |
| Total | | | | | | | 71 | 74 | 45 |
| | CIC | MB | CIC | 4 | 4 | 4 | 9 | 25 | 3 |
| WLANa | FIR | HB | FIR | 2 | 2 | 2 | 32 | 19 | 24+7 |
| Total | | | | | | | 41 | 44 | 34 |
| | CIC | MB | CIC | 4 | 4 | 4 | 4 | 45 | 3 |
| WiMAX | FIR | HB | FIR | 2 | 2 | 2 | 36 | 19 | 31+7 |
| Total | | | | | | | 41 | 44 | 41 |

MSD[1]: without CIC
MSD[2]: forcing CIC

## 7.6. VHDL IPs

In addition to the system level analysis carried out using the developed MATLAB MSD-toolbox, several VHDL IPs have been developed, synthesized, verified, and implemented through this work. The developed IPs are generic, structurally modeled, and accessible at all hierarchical levels.

### 7.6.1. Tools Chain

Figure 7.14.a shows the tools chain used in this research work including the manipulated files and parameters during each step in the design procedure, as shown in Fig. 7.14.b.
The system design of the decimation filter design exploits MATLAB for calculating the implementation parameters from the design specifications preserving the design constraints. The decimation filter design specifications are correlated with the $\Sigma\Delta$ modulator specifications. The necessary design specifications are the sampling frequency ($f_s$), the oversampling ratio (OSR), the resolution (SNR), and the in-band noise (IBN). From the preliminary design specifications the MATLAB MSD-toolbox exports the decimation filter implementation parameters, such as; number of decimation stages ($k$), decimation factor at each stage ($M_i$), filter order ($N$), coefficient quantization bit-width ($Q$), scaled discrete filter coefficients ($\hat{h}_k$).
Next, comes the RTL modeling for the FIR decimation filter. A generic VHDL model has been developed using CADENCE RTL design EDA tools. The models developed in this dissertation are all modeled structurally to assure efficient and hierarchical implementation. CADENCE NClaunch is used for RTL design, modeling, and elaboration.
Afterwards, the modeled design is verified employing stimuli bit-stream and test-bench using CADENCE SimVision. The verification process is an iterative feedback process. Often, the VHDL model might need slight modifications after few simulation runs to achieve the desired functionality. This first verification step is known by functional verification. It is an ideal simulation without considering gate or wire delays.
The developed VHDL models are then synthesized to certain CMOS ASIC technology or certain FPGA. This work involves the 130nm ASIC CMOS technology offered by UMC FARADY incorporation and Spartan-3E XILINX FPGA. This work uses SYNOPSYS synthesizer DesignCompiler for ASIC synthesis and XILINX ISE for FPGA synthesis. The design has to be constrained in time, area, and load during synthesis. A Tcl scripts is used for design constraints by defining the clock frequency, input and output delays, wireload mode and model, and clock skew. The synthesizer exports a mapped VHDL netlist, SDF file including all the gate delays, and SDC file including all the time constraints.
A conclusive verification is required after synthesis to verify the functionality of the design after considering the gate and wire delays by importing the (netlist+SDF+SDC).

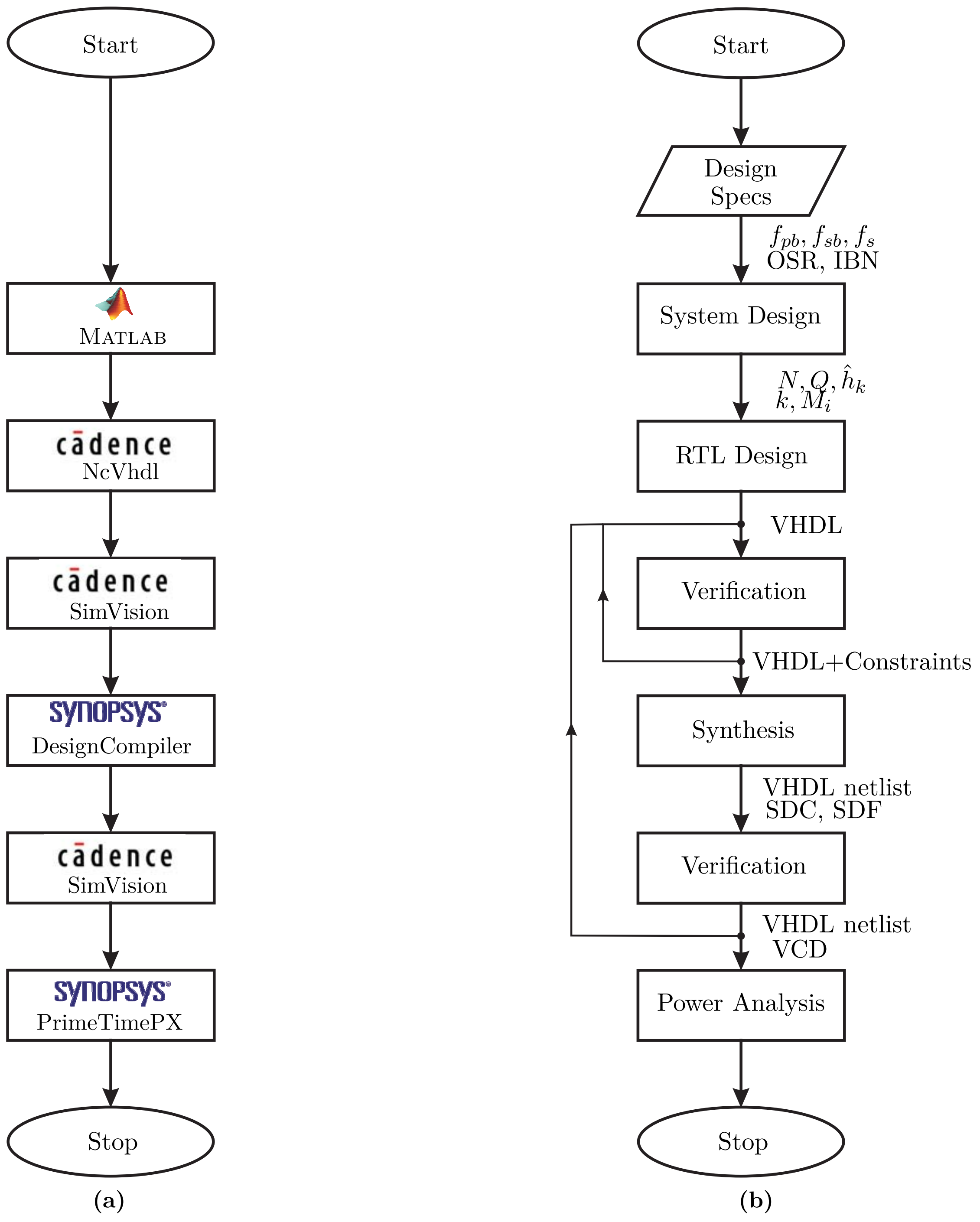
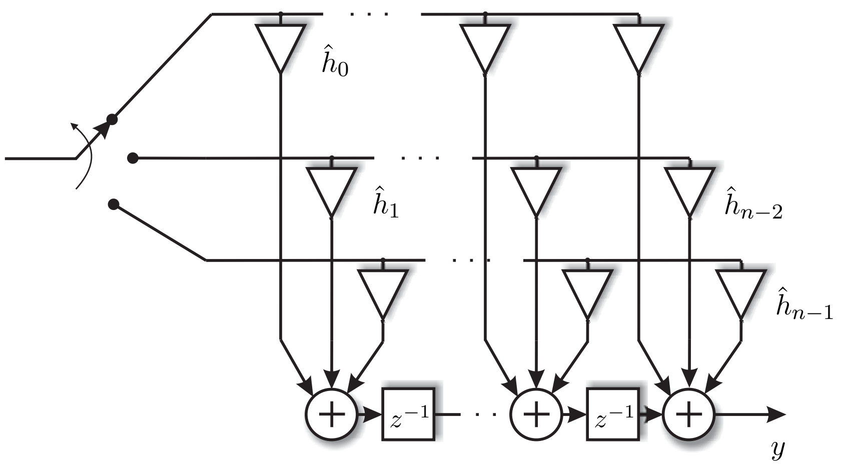


**Figure 7.14.:** Tools (a) chain (b) procedure.

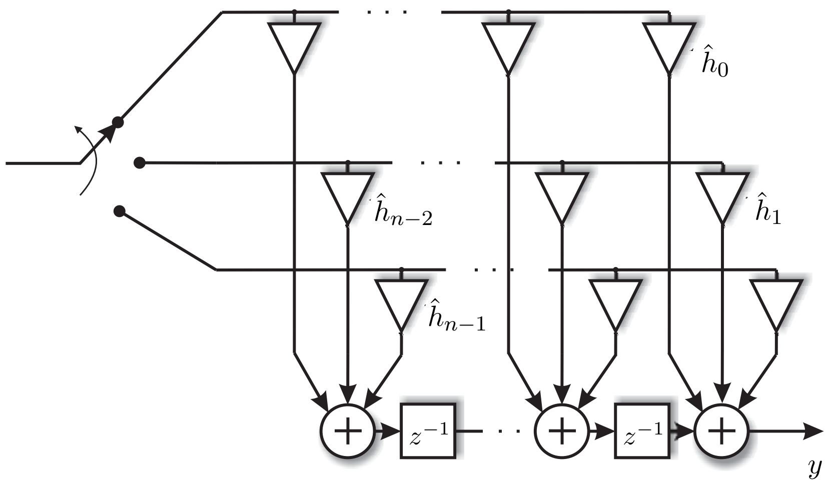
This is known as the timing verification. This verification step is also iterative feedback process. Cadence SimVision is used for verification and it exports a VCD file which is used for precise power analysis later on.
An accurate power analysis is performed using Synopsys PrimeTime-PX by importing (netlist+VCD).
A set of mathematical functions have been developed by VHDL via packages and procedures to perform certain functions which are not available in VHDL default packages. Those functions are: ceiling ($\lceil \quad \rceil$), floor ($\lfloor \quad \rfloor$), sum of elements ($\sum$), product of elements ($\prod$), maximum value of a vector (**max**), minimum value of a vector (**min**), binary logarithmic of a value ($\mathbf{log_2}$), flip vector elements in left-/right direction (**fliplr**), read a row from matrix (**readRow**), read a column from matrix (**readColumn**), convert 2D matrix to 1D vector (**2D-to-1D**), get index of a specified object in an array or matrix (**getIndex**), determine whether the integer is a power-of-two or not (**PoT**).
The mathematical operations are performed employing 2's complement representation. The mathematical operations imply natural and integer numbers to avoid using the VHDL "math_real" package, since not all synthesizers can synthesize it.
The "textio" package is integrated with the testbenches for efficient manipulation of the stimuli and exported data for precise analysis. The developed codes imply assertion statements for debugging and verification. A standard VHDL modeling for several FIR decimation filters can be found in [108].

### 7.6.2. Polyphase Decimation Filter

The polyphase decomposition is a fundamental structure for multirate signal processing applications [25], [26], [109]. It exhibits an efficient implementation of decimation filters because it reduces the computational load by factor of $M$ [25], [26], where $M$ is the decimation factor. Polyphase decomposition is very efficient for parallel processing realization [110]. Practically the polyphase structure is modeled using the commutator model [26], [91]. There are two major polyphase structures known as Type-I and Type-II structures. Type-I uses a counterclockwise commutator and it is conventionally used as analysis filter [25]. Type-II uses a clockwise commutator and it is conventionally used as synthesis filter [25]. Decimation filters are implemented in Type-I structures. A detailed analysis on polyphase fundamentals is given in [25], [26], [109]. Therefore, a polyphase structure has been chosen as a basic filter for this work.
Polyphase filters are implemented in direct-form topology (DF) and transposed-form topology (TF), as shown in Fig. 7.15.a and Fig. 7.15.b, respectively. The critical path in a DF topology has (1 multiplier + $\lceil N/M \rceil$ multi-operand adder). On the other hand, the critical path for a TF has (1 multiplier + 1 multi-operand adder). Therefore, the TF topology has less latency compared to DF topology [111]. Moreover, the TF topology offers a common space for all the filter coefficients multipliers, leading to an inter and intra multiplier optimization, such as common



**(a)**



**(b)**

**Figure 7.15.:** Polyphase decimation filters in (a) direct-form topology (PPD-DF) and (b) transposed-form topology (PPD-TF).

sub-expression elimination [111].
A generic VHDL IP is developed for a PPD filter for both DF and TF topologies. The individual filter components are described in the following sub sections. The main PPD filter components are a constant multiplier, a delay register, and a multi-operand adder. The polyphase decimation filter transfer function is given by (7.26) [26].

$$H(z) = \sum_{j=0}^{M-1} E_j(z^M)z^{-j} \tag{7.26}$$

where

$$E_j(z) = \sum_{i=\infty}^{-\infty} h[Mi + j]z^{-i}$$

where $N$ is the filter order, $M$ is the decimation factor, and $E$ is the polyphase decomposition.

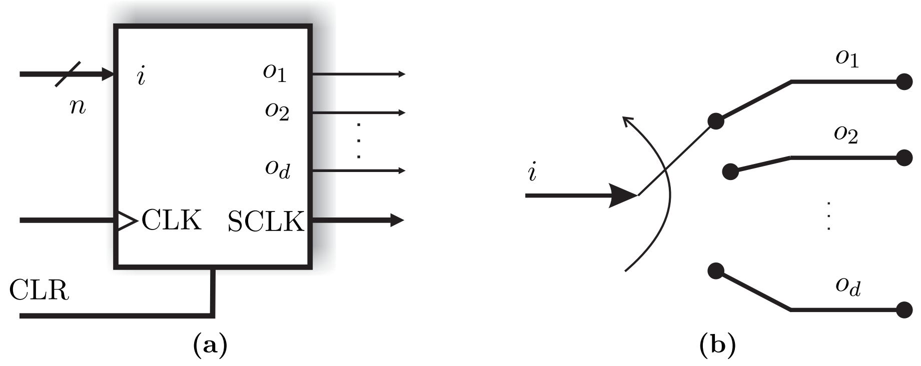


**Figure 7.16.:** Commutator (a) schematic (b) symbol.

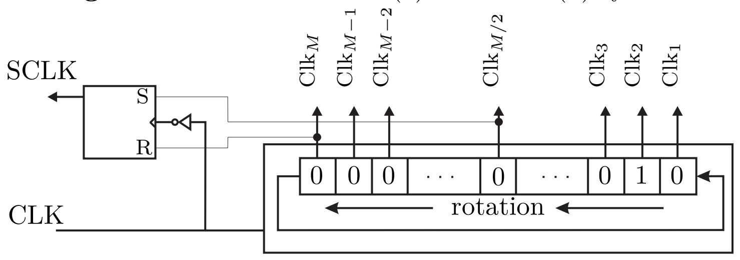


**Figure 7.17.:** Commutator clock generator.

#### 7.6.2.1. Commutator

The commutator is the block responsible for distributing the input data to the polyphase decimator filter branches. The input to the commutator, which is at the same time the output from the ΣΔM or output from a previous decimation stage, is at high sampling rate $f_s$ and the output rate at $f_s/M$, where $M$ is the decimation factor and the number of polyphase decimator branches as well. The commutator schematic and symbol are shown in Fig. 7.16.a and Fig. 7.16.b, respectively. The commutator consists of two main blocks [91], [100] the clock generator and the double clocked flip-flop as given in Fig. 7.17 and Fig. 7.18.b, respectively. The clock generator generates $M+1$ clocks. A slow clock (SCLK) equivalent to $f_s/M$, and $M$ clocks ($\mathrm{CLK}_i$), as exhibited in Fig. 7.18.a. The clocks are, input clock, which is generated from a clock generator block and responsible for storing the data in each flip-flop for each equivalent branch in the polyphase decimator. The output clock is the slow clock (SCLK) generated from the clock generator block, too. It is equivalent to $f_s/M$ where $M$ is the decimation factor for the current decimation stage. In the case of cascaded decimation structures this output clock will be used as the input sampling clock for the following stage. This previous clock will be used to output the stored data from each flip-flop. The double clocked flip-flop is a delay flip-flop, stores the data referred to the input clock positive edge, and transfer data to output referred to output clock (SCLK) positive edge. The commutator input ($i$) has a bit-width ($W_i$), as shown in Fig. 7.18.b.

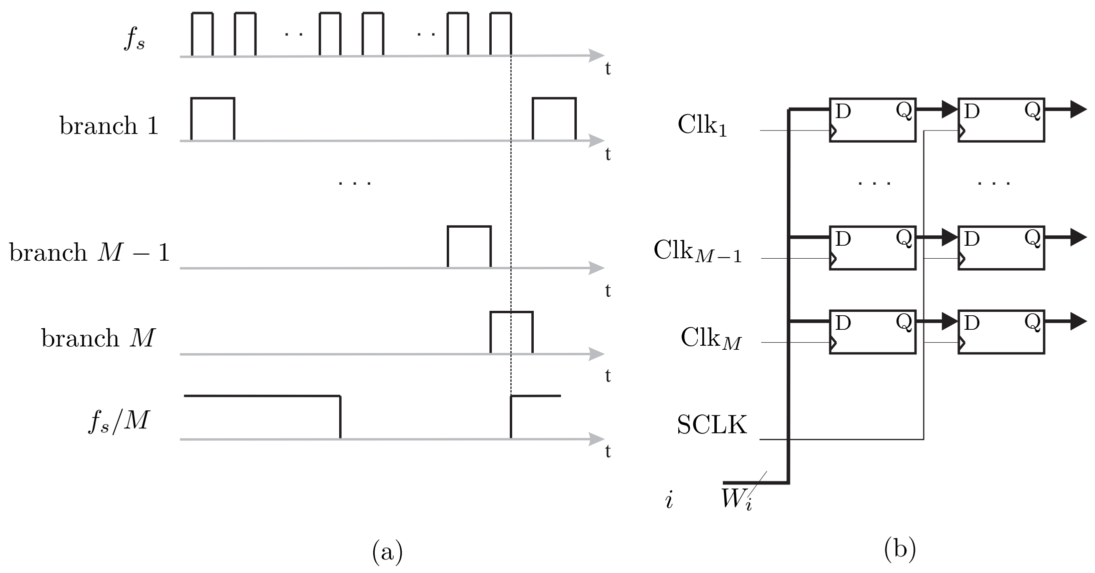


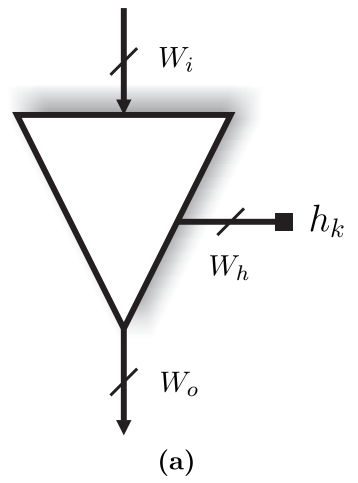
**Figure 7.18.:** Commutator (a) timing diagram (b) double clocked flip-flop.

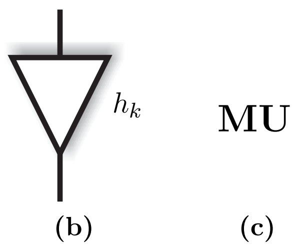
### 7.6.2.2. Signed Constant Multiplier

A constant multiplier is implemented behaviorally by using the (*) symbol, to assess the synthesizer smart datapath optimization. Further, it is implemented in shift-and-add topology employing the encoding scheme proposed in this work which is presented in chapter 8. The multiplier schematic, symbol, abbreviation, and parameters are depicted in Fig. 7.19.a to Fig. 7.19.d. Where, the SU parameter is used to identify whether the input data is a signed or unsigned bit stream data. If it is unsigned this reveals to a data range $\{0, 2^{W_i}-1\}$. On the other hand, if it is signed, the data ranges $\{-2^{W_i-1}, 2^{W_i-1}-1\}$. The $W_i$ represents the ΣΔM output or a previous filter output (in case of cascaded decimation filters). The $W_h$ is equivalent to the quantization bit-width ($Q$). The output bit-width is calculated from the input and coefficient bit-width $W_o = W_i + W_h$. Conventionally, the multiplier design parameters are fixed through the whole filter.

### 7.6.2.3. Delay Register

A straight forward implementation for a positive edge trigger delay flip-flop (DFF) with active high clear signal (CLR) is developed as shown in Fig. 7.20.a. The delay schematic, symbol, abbreviation, and parameters are given in Fig. 7.20.a to Fig. 7.20.d. The DFF has a single design parameter define that the input and output bit-width of the delay register elements. Conventionally, the delay register design parameter is fixed through the whole filter, i.e., all the delay registers have the same $W$. However, the developed IP provides arbitrary register bit-widths within the same filter according to the preceding multi-operand adder.

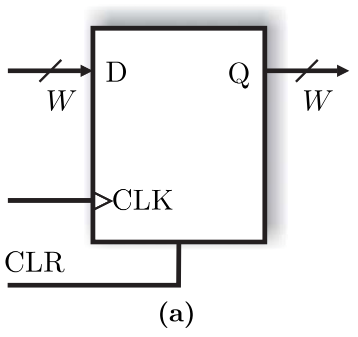


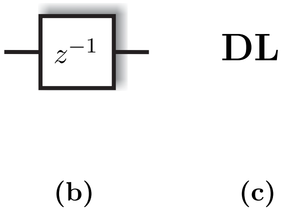


| | |
|---|---|
| $W_i$ | input bit-width |
| $W_h$ | coefficient bit-width |
| $W_o$ | output bit-width |
| $\hat{h}_k$ | scaled coefficient |
| SU | Signed \| Unsigned |

(d)

**Figure 7.19.:** Multiplier element - (a) schematic, (b) symbol, (c) abbreviation, (d) parameters.





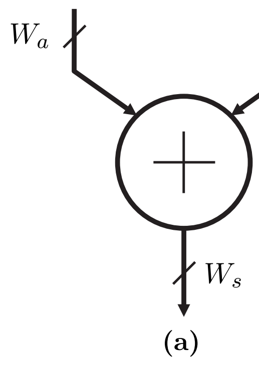
| | |
|---|---|
| $W$ | input and output bit-width |

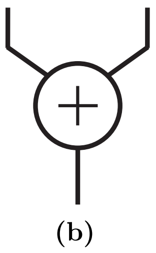
(d)

**Figure 7.20.:** Delay element - (a) schematic, (b) symbol, (c) abbreviation, (d) parameters.

#### 7.6.2.4. Multi-operand Adder

Multi-operand adder is an arithmetic component used for multi-input adders. The standard adder component which is known as the full-adder (FA) (also known as 3-2 compressor) can only add two inputs, despite the carry input. Therefore, an adder cell element is developed with certain design parameters to afford further flexibility in multi-operand adder tree structures later on. The developed adder cell (AD) schematic, symbol, abbreviation, and parameters are shown in Fig. 7.21.a to Fig. 7.21.d. Conventionally, for a multi-bit parallel adder both adder inputs have the same bit-width ($W_a = W_b$) and the output bit-width $W_s = W_a + 1$. Each adder cell has certain input bit-width for both inputs ($W_a, W_b$) as shown in Fig. 7.21.a. The developed adder cell employs signed addition. The adder is implemented in ripple carry adder (RCA) topology for assuring low power implementation [107]. A multi-operand adder can be implemented with unbalanced or balanced tree





**AD**

**(c)**

| | |
|---|---|
| $W_a$ | input $A$ bit-width |
| $W_b$ | input $B$ bit-width |
| $W_s$ | output $S$ bit-width |

**(d)**

**Figure 7.21.:** Adder element - (a) schematic, (b) symbol, (c) abbreviation, (d) parameters.

structures [107], as shown in Fig. 7.22. The total number of adders used in both structures is the same but the balanced tree structure has less latency [107]. The multi-operand adder schematic, symbol, abbreviation, and parameters are given in Fig. 7.23.a to Fig. 7.23.d. The number of multi-operand adder inputs is derived from the decimation factor ($M_i$). The balanced multi-operand adder tree is the only structure involved in this work. For an optimized implementation of a balanced multi-operand adder structure, it has been modeled structurally. For the internal implementation a set of parameters are calculated in advance, such as; total number of adders (TNoA), number of depth Levels (Levels), number of adders per level (NoA), levels being subsequent to skipped levels (SL), and levels provide internal structural connection for skipped levels (LoSL). Those set of design parameters are calculated from the number of inputs for the MA. The NoA and SL

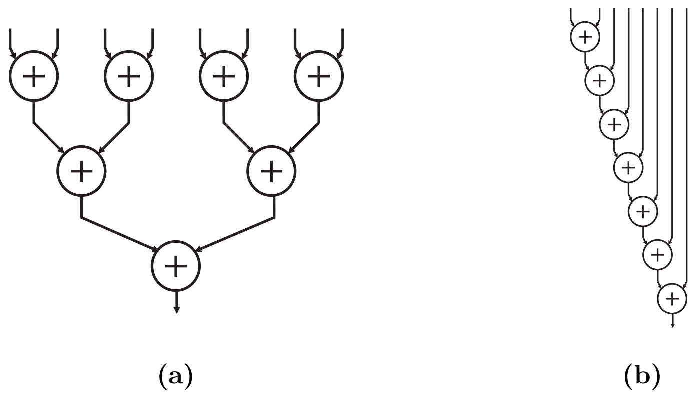
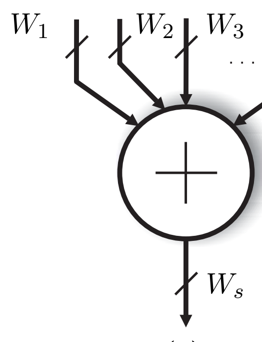
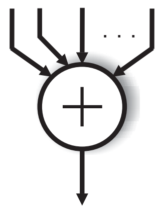


**Figure 7.22.:** Multi-operand adder structures for 8 inputs (a) balanced and (b) unbalanced.



(a)



(b)

**MOA**

(c)

| | |
|---|---|
| $W_1$ | input 1 bit-width |
| $W_2$ | input 2 bit-width |
| $W_d$ | input $d$ bit-width |
| $W_s$ | output $S$ bit-width |
| $d$ | number of inputs |

(d)

**Figure 7.23.:** Multi-operand adder (a) schematic, (b) symbol, (c) abbreviation, (d) parameters.

are calculated according to the flowcharts given in Figs 7.24 and 7.25, respectively. The proposed algorithm for calculating the NoA, as shown in Fig.7.24, has three routines (**R-I,R-II,R-III**) and a single loop. R-I, determines whether the number of adders at the depicted level is odd or even and set or reset the skipped level flag, respectively. R-II, increments the skipped level flag or maintains it according to odd or even number of adders at the depicted level, respectively. R-III, used only at the first level (input level) of the multi-operand adder, and it sets or resets a flag for odd or even number of inputs, respectively. The number of inputs is defined by ($d$). A flag (Extra) is reset (Extra=0) for even number of inputs and is set (Extra=1) for odd number of inputs. Y and N represent the decision yes or no respectively. E and O represent the decision for even or odd respectively. An illustrative example is presented in Fig. 7.26. For an eleven input MA, a total of 10 adders are needed at four levels. The number of adders per level is depicted by NoA in Fig. 7.26. There are two levels imply skipped level, the second and the last levels, as given by SL in Fig. 7.26. The first skipped level has an internal structural connection from the multi-operand adder direct inputs, so that it has ($-1$) flag in the LoSL matrix.

| | |
|---|---|
| total number of adders (TNoA) | $= m - 1$ |
| number of depth levels (Levels) | $= \lceil \log_2(m) \rceil$ |
| number of adders per level (NoA) | $= [5, 3, 1, 1]$ |
| subsequent levels for skipped levels (SL) | $= [0, 1, 0, 1]$ |
| levels provide internal connection of skipped levels (LoSL) | $= [(0, 0), (-1, 2), (0, 0), (3, 4)]$ |

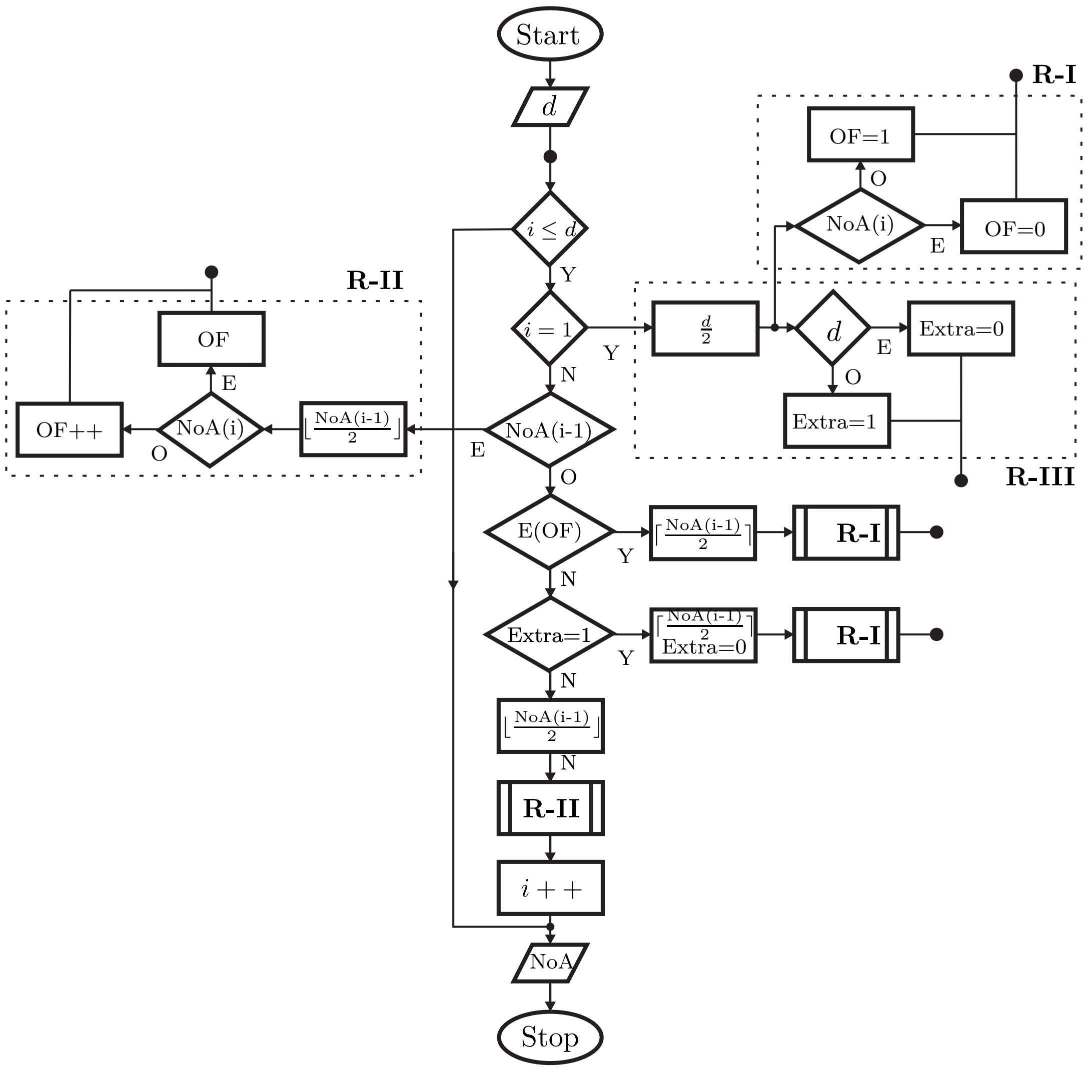


**Figure 7.24.:** Determine the number of adders per level.

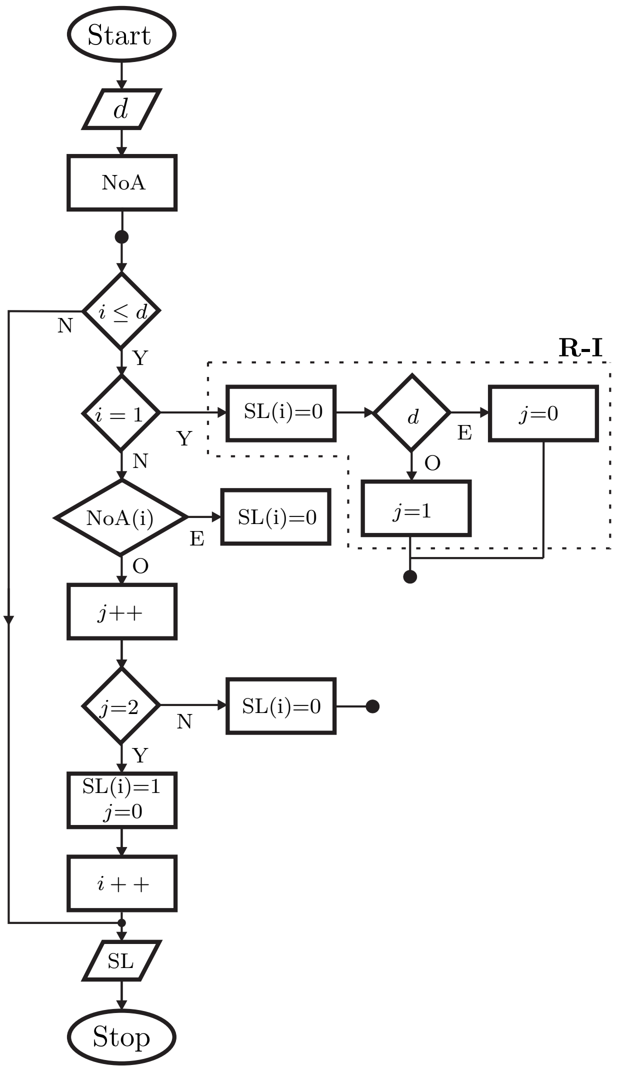


**Figure 7.25.:** Determine the internal skipped levels.

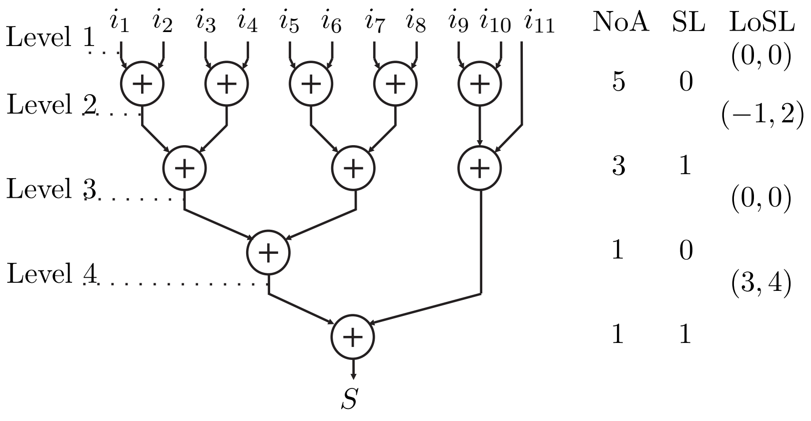


**Figure 7.26.:** Balanced multi-operand adder structure for 11 inputs.

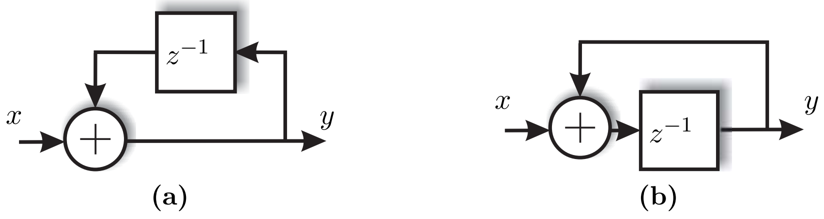


**Figure 7.27.:** Integrator stage (a) basic (b) pipeline.

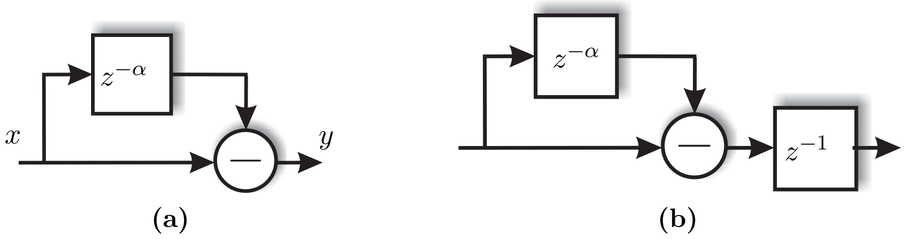


**Figure 7.28.:** Comb stage (a) basic (b) pipeline.

### 7.6.3. Cascaded Integrator Comb Filter

Decimation filters have to offer strong stopband attenuation factor to prevent aliasing and imaging into the in-band, as shown in Fig. 2.5. Cascaded Integrator Comb (CIC) filter, is a class of linear phase FIR digital filters that provides deep stopband attenuation at the aliased signal bands [105]. Furthermore, CIC filters are power and area efficient filters because they require no multiplier and require no storage [105]. The CIC filter consists of two basic building blocks, an integrator and a comb [29]. It should be noted, that CIC filter needs a correction or compensation stage to compensate for the drop in the magnitude within the passband.
The integrator difference equation is given by (7.27). Equation (7.28) [105] shows the transfer function of that integrator by taking the $z$-transform of (7.27). An integrator stage is implemented as exhibited in Fig. 7.27.

$$y[n] = x[n] + y[n-1] \tag{7.27}$$

$$H(z) = \frac{1}{1 - z^{-1}} \tag{7.28}$$

The comb difference equation is given by (7.29). Equation (7.30) [105] shows the transfer function of a comb by the $z$-transformation of (7.29). A comb stage is implemented as exhibited in Fig. 7.28.

$$y[n] = x[n] - x[n-\alpha] \tag{7.29}$$

$$H(z) = 1 - z^{-\alpha} \tag{7.30}$$

where $\alpha = D \times M$.
The integrator is unstable itself [29], it has infinite dc gain. The comb is the

inverse of the integrator [29]. The cascade of integrator and comb stages forms a FIR filter, named CIC. The CIC transfer function is given by (7.11). This is a recursive implementation of the CIC filter. Hence, (7.11) can be modified as shown by (7.31) for non-recursive implementation [112]. Several alternative implementation of CIC filter is given in [112]. However, this work considers the recursive implementation only, because of its efficiency with high over sampling ratios [110]. Moreover, the recursive CIC filters can be implemented in different topologies, such as conventional (standard), pipe-lined, pipe-lined integrator, and pipe-lined comb as given in Fig. 7.29.a to Fig. 7.29.d, respectively. The developed IP provides the advantage of setting each individual stage, rather integrator or comb, to pipe-lined or not. Thus, a hybrid CIC architecture is proposed in this work for low power implementation [113], as shown in Fig. 7.29.e.

$$H(z) = \left( \sum_{i=0}^{\alpha-1} z^{-i} \right)^{N} \tag{7.31}$$

#### 7.6.3.1. Integrator

The integrator stage is implemented structurally using the predefined DL and AD elements, shown in Figs. 7.20 and 7.21, respectively, to construct the basic and the pipelined integrator architecture, shown in Fig. 7.27.a and Fig. 7.27.b respectively. The pipeline integrator has no additional pipeline registers [114], which increases the clock rate for the CIC filter [114]. The integrator schematic, symbol, abbreviation, and design parameters are given in Fig. 7.30.a to Fig. 7.30.d.

#### 7.6.3.2. Comb

The comb stage is implemented structurally using the predefined DL and AD elements, shown in Figs. 7.20 and 7.21, respectively, to construct the basic and the pipelined comb architectures, shown in Fig. 7.28.a and Fig. 7.28.b respectively. The subtraction process is implemented by the inversion of one of the inputs of the AD and setting the carry-in to one. The pipeline comb has one additional pipeline register [114], in contrary with its counter part the integrator stage. However, it increases the clock rate for the CIC filter [114]. The comb schematic, symbol, abbreviation, and design parameters are given in Fig. 7.31.a to Fig. 7.31.d.

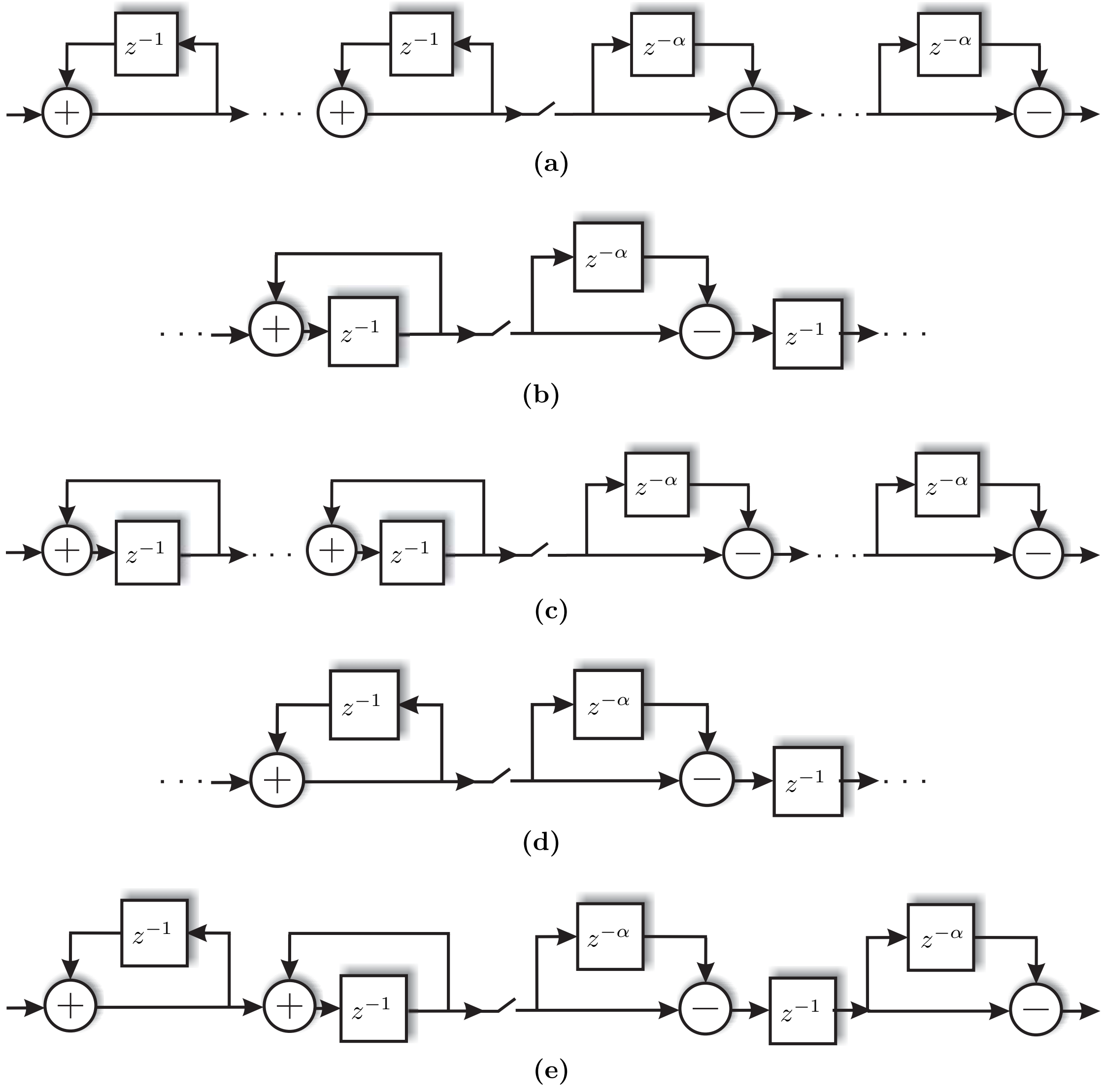
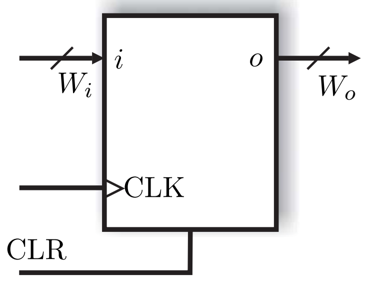
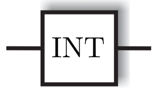


**Figure 7.29.:** Cascaded integrator comb configurations (a) conventional (b) pipelined (c) with pipelined integrator (d) with pipelined comb (e) hybrid.



**(a)**



**(b)**

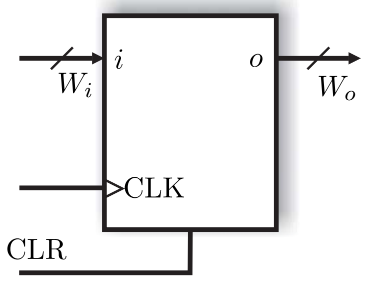
**INT**

**(c)**

| | |
|---|---|
| $W_i$ | input bit-width |
| $W_o$ | output bit-width |
| PL | pipeline |

**(d)**

**Figure 7.30.:** Integrator stage - (a) schematic, (b) symbol, (c) abbreviation, (d) parameters.



**(a)**

**(b)**

**COMB**

**(c)**

| | |
|---|---|
| $W_i$ | input bit-width |
| $W_o$ | output bit-width |
| PL | pipeline |
| D | differential delay |

**(d)**

**Figure 7.31.:** Comb stage - (a) schematic, (b) symbol, (c) abbreviation, (d) parameters.

## 7.7. Summary

A consolidated design procedure and single design platform for multi-stage decimation filters were developed in this work. The MSD-toolbox involves the methodologies developed through this work besides the state-of-the-art research methodologies. The toolbox provides a fast design process for FIR decimation filters. The MSD calculates the optimal number of decimation stages and the decimation factor at each stage sustaining minimum filter computational effort. Further, it calculates the frequency specifications for the various decimation stages, such as passband and stopband frequencies, passband ripples and stopband attenuation. Moreover, the MSD acquires the filter coefficients using Parks-McClellan Equiripple algorithm for standard, or halfband, or multi-band filters. Afterwards, the appropriate quantization bit-width is calculated for each decimation stage sustaining minimum mean error in the filter response and minimum penalty in the in-band noise. Finally, the MSD-toolbox integrates several optimization methodologies for optimizing the fixed point filter coefficients, such as mixed integer linear programming, polynomial programming, and iterative optimization.
The MSD-toolbox brings a VHDL model for the designed and optimized filter coefficients in polyphase as well as cascaded integer comb filters. Generic VHDL models were developed for polyphase and cascaded integrator comb decimation filters as RTL-based soft IPs. The developed IPs have structural hierarchy. The developed VHDL IPs have the advantage of flexible accessibility for each implementation parameter of each individual block.
The toolbox supplies a concrete bit-level activity files for troubleshooting and verification.

# 8. RTL Power Optimization

## 8.1. Introduction

This chapter emphasizes on power optimization criteria on the RTL for constant multipliers and consequently on the multiple constant multiplications (MCM) problem for digital filters. Constant multipliers are conventionally known as shift-and-add or multiplierless architectures. In order to reduce the power dissipation in constant multiplier, this can be done on three different perspectives, which are: 1) reduce the number of FAs used within the constant multiplier according to reducing the internal bit-width, 2) reduce the number of logic operations (LO), i.e. the overall number of adders used in the constant multiplier through CSE, and 3) reduce the logic depth (LD) due to employing balanced tree architectures and CSD representation. This work proposes a novel criterion to reduce the internal bit-width within the constant multiplier, consequently the number of FAs. Further, the author combines several approaches to achieve efficient CSE. Moreover, the CSE is used to drive or steer the construction of a constant multiplier.

## 8.2. Problem Notations

The common notations used within this chapter are defined in this section to offer a consistent representation throughout the work.
Consider the integer coefficient in decimal representation
$\hat{h}_k = 49984_{10}$
with its binary representation
$\hat{h}_k = 1100001101000000_2$ [MSB ... LSB]
where its binary-to-decimal representation is
$\hat{h}_k = 2^6 + 2^8 + 2^9 + 2^{14} + 2^{15}$
Multiplication is a shift left operation. Therefore, from the previous format, the coefficient can be represented in a format of shifts, as shown
$\hat{h}_k = x \ll 6 + x \ll 8 + x \ll 9 + x \ll 14 + x \ll 15$
where $\ll$ is logical shift left logical operation, i.e. the number of zeros added to the right of the input of a multiplier $x$. There is a simplified notation used to represent the shifts, as follow
$S\{1 : m\} = 6 + 8 + 9 + 14 + 15$
where $m$ is the number of non-zero terms in the binary representation of the

coefficient. The set S represents the number of shifts at each non-zero term in the binary representation. The power of two factors represents the shift left (number of zeros added to the right of the input). Figure 8.1 shows the construction of a shift-and-add constant multiplier and the used notations is depicted on the figure, where

- Ax Adder number x
- [Wx] Number of full-adders (NFAs), which is equivalent to the output bit-width of adder Ax
- Sx Shift left bits
- Lx Level x

The number of full-adders (NFAs) is calculated at the first level as given in (8.1) and at the internal levels as in (8.2) follow

$$\text{NFAs} = \mathbf{max}(Si, Si + 1) + W_i + 1 \tag{8.1}$$

$$\text{NFAs} = \mathbf{max}(W_i, W_{i+1}) + 1 \tag{8.2}$$

where $W_i$ is the constant multiplier input bit-width, and the extra one to avoid an overflow problem. It has to be noted that, in case of signed multiplication the MSB adder will have an extra bit for sign. Further, the depth of the multiplier is calculated by (8.3)

$$d = \lceil log_2(m) \rceil \tag{8.3}$$

where $m = Cost(\hat{h}_k)$, and the total number of adders #A= $m - 1$.

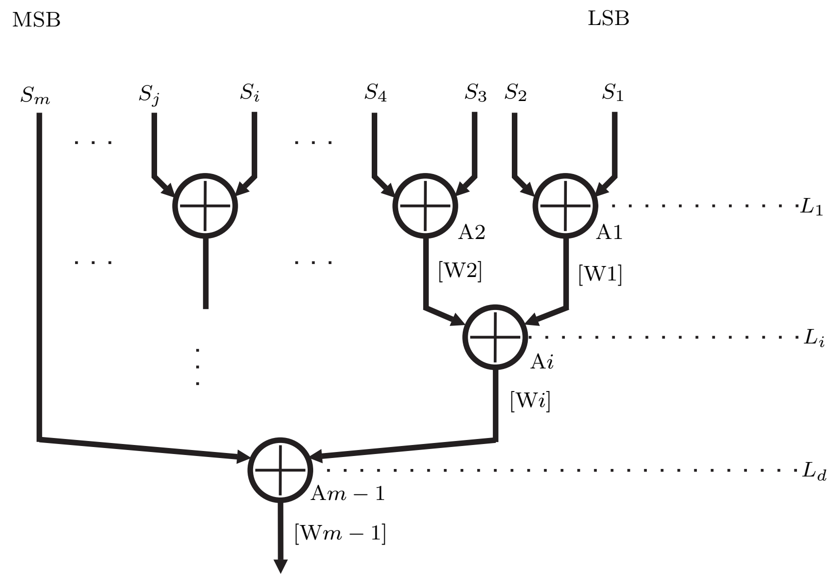


**Figure 8.1.:** Problem notations.

## 8.3. Discrepancy

Before presenting the proposed criterion, a quantitative study has been carried out which ended with a useful observation which can be used for efficient construction for the constant multiplier. Does constructing the shift-and-add from LSB $\rightarrow$ MSB or MSB $\rightarrow$ LSB influence the internal bit-width? Figure 8.2 shows the effect of the direction of constructing the shift-and-add multiplier on the number of FAs. The shift-and-add multiplier constructed using the all LSB $\rightarrow$ MSB criterion exhibits the lowest NFAs, as shown in Fig. 8.2.a. However, the shift-and-add multiplier constructed using the all MSB $\rightarrow$ LSB criterion exhibits the highest NFAs, as shown in Fig. 8.2.c. On the other hand, when combining both LSB $\rightarrow$ MSB and MSB $\rightarrow$ LSB, or vice versa as well, the NFAs differ according to the direction of construction, as shown in Fig. 8.2.b and Fig. 8.2.d. Therefore, the author defined parameters called LSB discrepancy ($\partial_L$) and MSB discrepancy ($\partial_M$) to identify the direction of constructing the shift-and-add. The $\partial_L$ is calculated according to the last two LSBs and the $\partial_M$ is calculated according to the last two MSBs, as depicted in (8.4)

$$\begin{aligned}\partial_L &= S_2 - S_1 \\ \partial_M &= S_m - S_{m-1}\end{aligned} \tag{8.4}$$

Therefore, the shift-and-add multiplier construction starts from the least discrepancy, i.e. if $\partial_L < \partial_M$ the construction take place from LSB $\rightarrow$ MSB and vice verse.

## 8.4. Proposed Nested Constant Multiplier

### 8.4.1. Theory

Nested multiplication, also known as Horner's rule, is an efficient mathematical method for rounding and chopping polynomials [115]. The nested multiplication is defined as follow [115], [116]:

$$\begin{aligned}p(x) &= a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} + a_nx^n \\ p(x) &= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x(a_n))\cdots))\end{aligned} \tag{8.5}$$

It can be expressed by the pseudo code [115], [116]:

```
px=a(n)
for k = n-1 downto 0
```

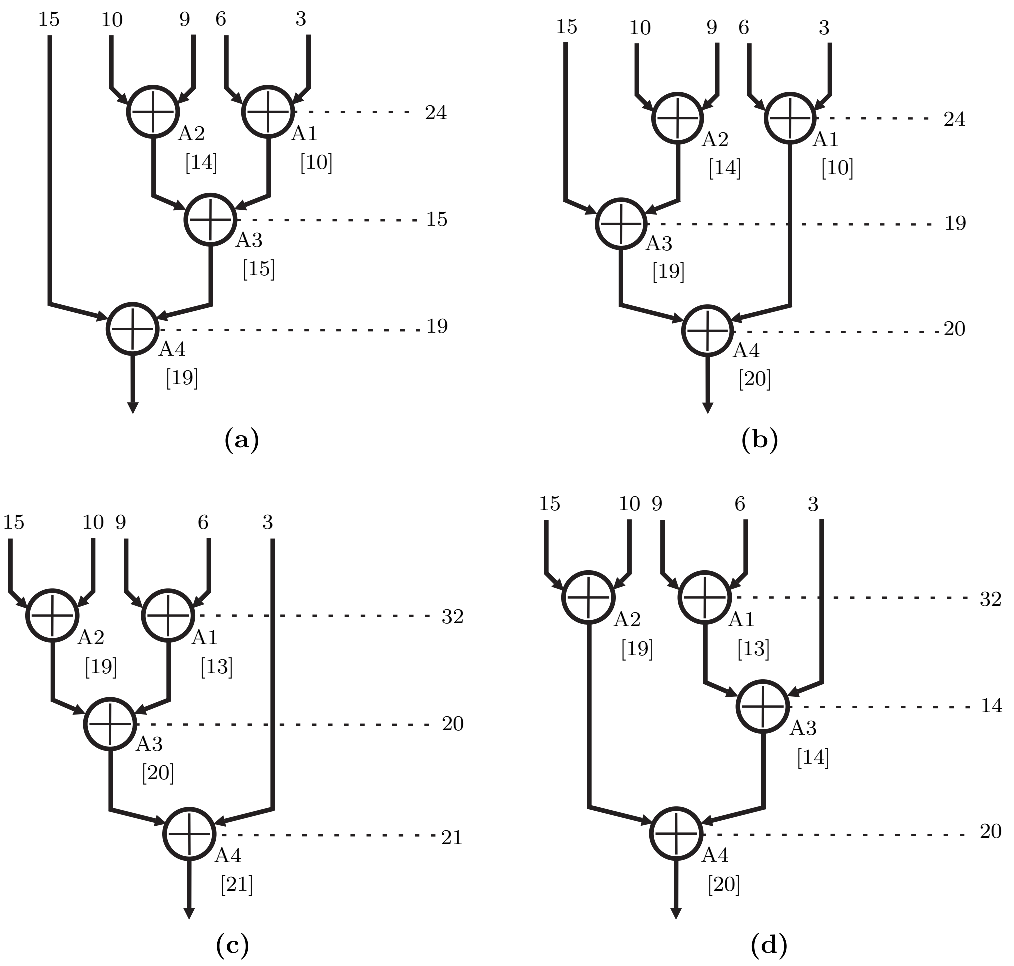


**Figure 8.2.:** Effect of (LSB → MSB) and (MSB → LSB) on constructing a constant multiplier with $W_i = 3$ (a) all LSB→MSB [58 FA] (b) LSB→MSB+MSB→MSB [63 FA] (c) all MSB→MSB [73 FA] (d) MSB→MSB+LSB→MSB [66 FA].

```
    px = a(k)+px × x
end
```

### 8.4.2. Implementation

The author was inspired by the nested multiplication criterion to implement it in hardware for within the shift-and-add constant multipliers. The advantage behind the nested multiplication is reducing the bit-width at adder inputs, consequently the NFAs.
The nested multiplication can be transferred to hardware form as shown in Fig. 8.3. The criterion is applicable only for nodes which share the same adder cell. The first step is to relocate the least number of shift bits at the output of the adder cell. Then reduce the input shifts at both of the adder cell. Numerically, this can

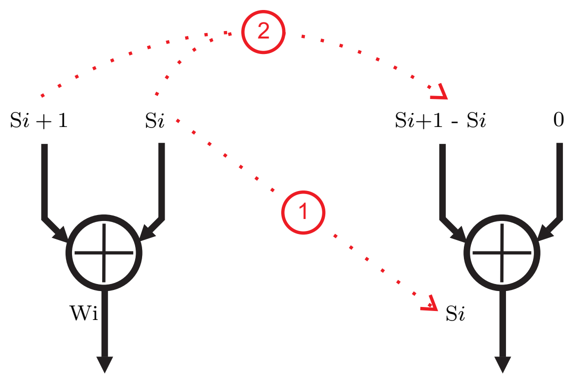


**Figure 8.3.:** Illustrating the proposed nested multiplier.

be expressed as follow for the integer coefficient 49984
$\hat{h}_k = 49984_{10}$
$\hat{h}_k = 1100001101000000_2$ [MSB … LSB]
$\hat{h}_k = 2^6 + 2^8 + 2^9 + 2^{14} + 2^{15}$
$S\{1:m\} = 6 + 8 + 9 + 14 + 15$

$$\begin{array}{ll} S_{L1} = 6 + 8(0+1) + 14(0+1) & A_{L1} = \{(0+1), (0+1)\} \\ S_{L2} = 6(0+2) + 14 & A_{L2} = \{(0+2)\} \\ S_{L3} = 6(0+8) & A_{L3} = \{(0+8)\} \end{array}$$

The numbers between the brackets () represents the number of shift bits at each adder cell input, i.e. each () is equivalent to a single adder cell. The number outside the brackets, e.g., 8() represents the shift bits at the output of the adder cell. The $S_L$ vectors represents the nested multiplication criterion at each level of the shift-and-add multiplier. The $A_L$ vectors hold the adder cell at each level. The advantage of the proposed criterion can be observed at each adder cell, e.g. for the adder cell (8+9) the NFAs is 9+$Wi$+1, while for 8(0+1) the NFAs is 1+$W_i$+1, which offers a reduction in the NFAs by $100 \times 8/(9 + W_i + 1)$, which is inversely proportional with input bit-width.
A level-by-level illustrative example for the coefficient 34376 with input bit-width $W_i$ 3-bits is given in Fig. 8.4. $\hat{h}_k = 34376_{10}$
$\hat{h}_k = 1000011001001000_2$ [MSB … LSB]
$\hat{h}_k = 2^3 + 2^6 + 2^9 + 2^{10} + 2^{15}$
$S\{1:m\} = 3 + 6 + 9 + 10 + 15$

The proposed criterion is compared to the conventional and the state-of-the-art methods as shown in Fig. 8.5. The proposed criterion shows reduction in the NFAs by 32.75% compared to the conventional architecture, and reduction by 17% compared to the pseudo floating point (PFP) method [82].

In order to evaluate the performance of the NMU criterion, a set of simulation sce-

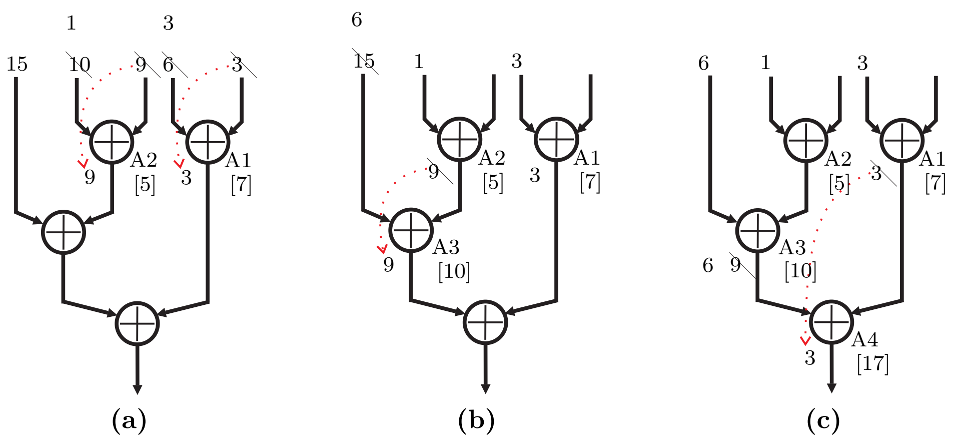


**Figure 8.4.:** Proposed nested multiplier construction at (a) Level-1 (b) Level-2 (c) Level-3.

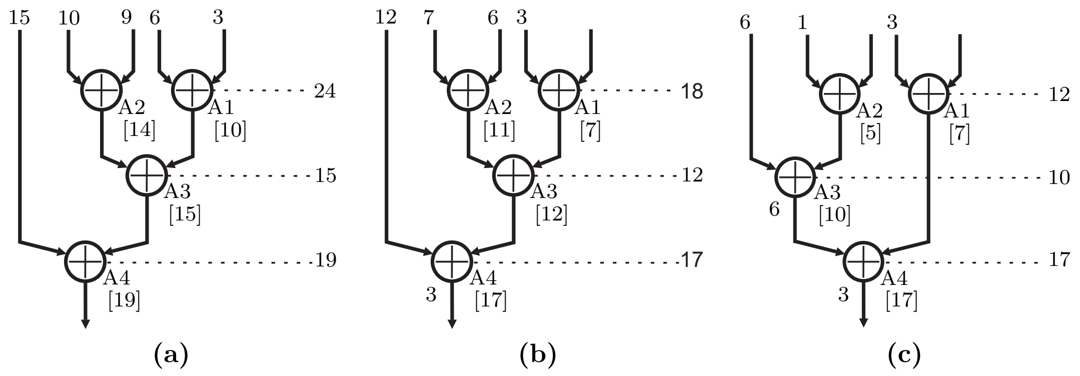


**Figure 8.5.:** Number of multiplier adders using (a) conventional [58 FA] (b) PFP [47 FA] (c) proposed [39 FA].

narios were carried out for various input bit-widths ($W_i$) and various quantization bit-widths ($Q$) considering binary and CSD representations. Figure 8.6 shows the total number of full-adders (NFAs) used to implement coefficient from 9 to 1024 using the PFP and the NMU criteria for binary and CSD as well. The result shows a considerable reduction in NFAs due to using the proposed criterion compared to the state-of-the-art algorithm. The percent of reduction in CSD constant multipliers is less than their binary counterparts, as exhibited in Fig. 8.6.b, due to the fact that CSD has less number of non-zero terms.
On the one hand, the NFAs increase rapidly with the increase of input bit-width, as can be seen from the slope of the lines given in Fig. 8.7.a. On the other hand, the NFAs increase in constant rate with the increase of the quantization bit-width, as can be seen at the vertical values for the same $W_i$ as shown in Fig. 8.7.a. However, for CSD multipliers the increase in NFAs is not as excessive as in binary multipliers, as can be seen in Fig. 8.7.b.
The percent of reduction in the NFAs is reduced with the increase of $W_i$, however, it is increased with the increase of $Q$, as depicted in Fig. 8.8.a and Fig. 8.8.b.

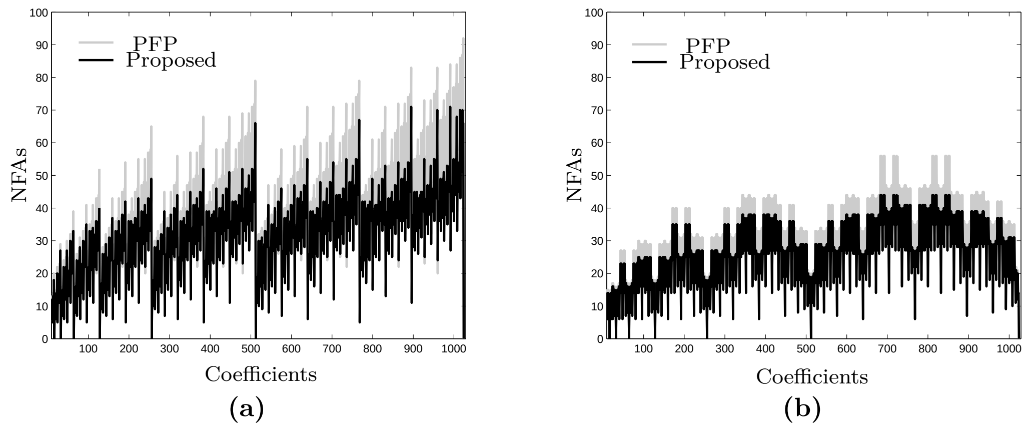


**Figure 8.6.:** NFAs using PFP versus proposed NMU for (a) binary (b) CSD coefficients.

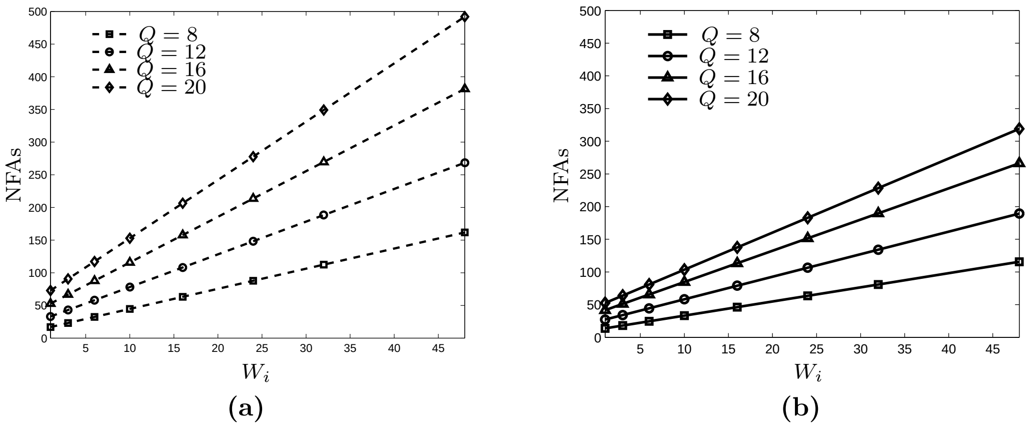


**Figure 8.7.:** NFAs employing the NMU criterion for (a) binary and (b) CSD representations.

Figure 8.9 shows the difference between the binary and CSD employing NMU.

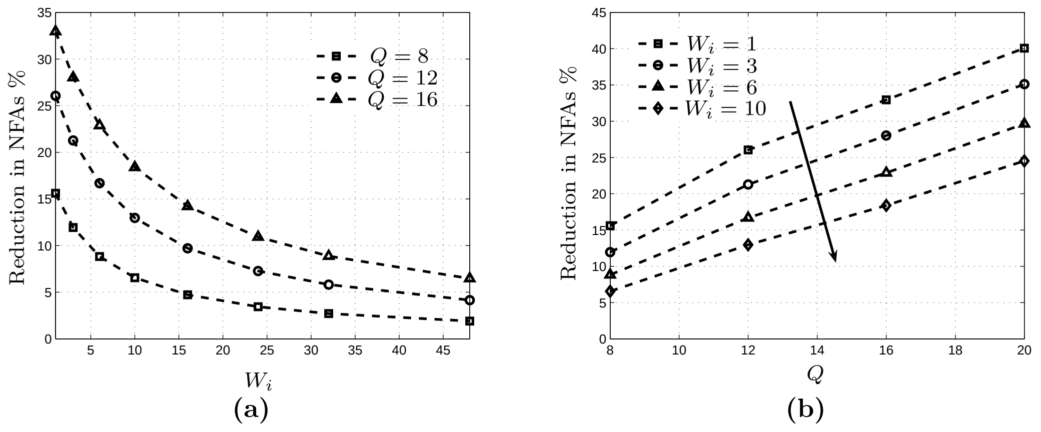


**Figure 8.8.:** Percent of reduction in NFAs using the proposed criterion according to variable (a) input bit-width (b) quantization bit-width.

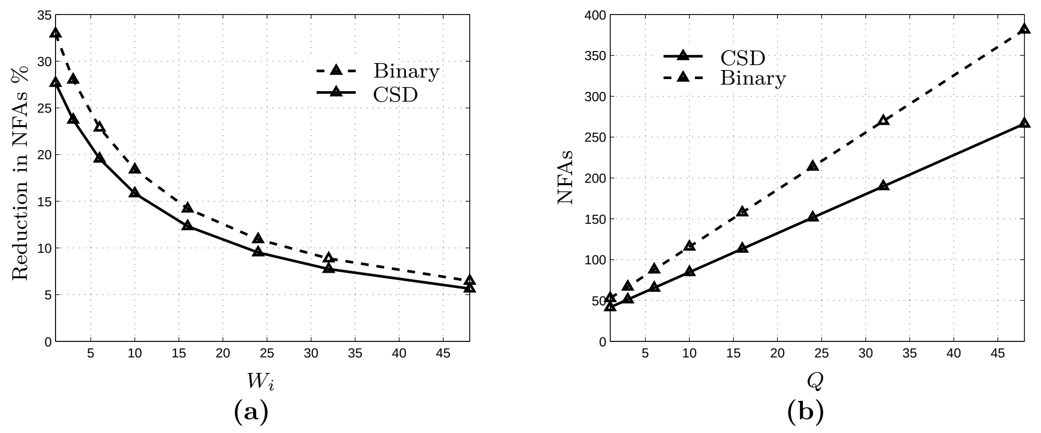


**Figure 8.9.:** Binary versus CSD employing proposed criterion for $Q = 16$ (a) percent of reduction in NFAs (b) NFAs.

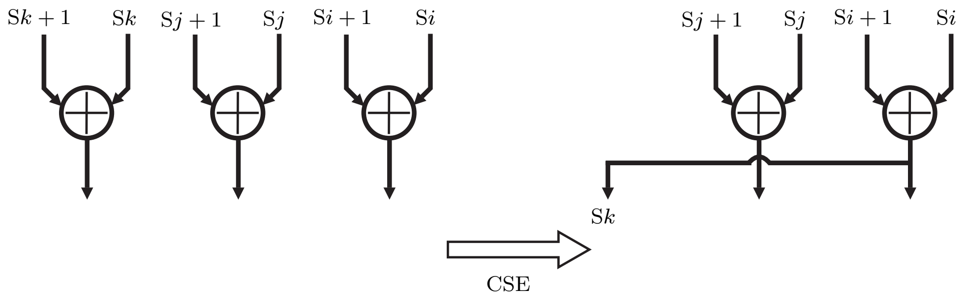


**Figure 8.10.:** CSE effect on multiplier adders.

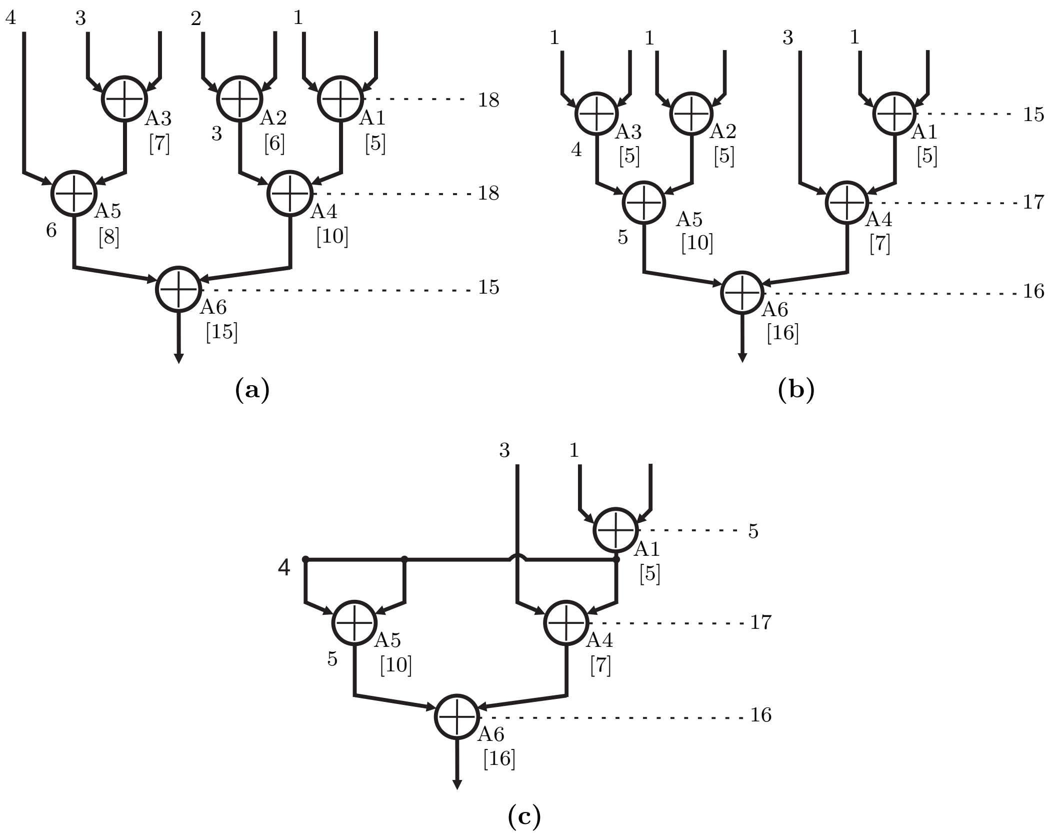


**Figure 8.11.:** NFAs in (a) NMU [51 FAs] (b) NMU guided by CSE [48 FAs] (c) after CSE [38 FAs].

## 8.4.3. Nested Multiplication Driven by CSE

The common sub-expression elimination given in section 5.9 is integrated within the construction of shift-and-add multipliers to guide the construction process. CSE is effective at first level only, since the internal adders are uncorrelated (do not have the same input). The basic idea of CSE is given in Fig. 8.10. The effect of employing CSE to steer the construction of shift-and-add multipliers is shown in Fig. 8.11.

## 8.5. RTL Modeling

The shift-and-add implies two basic components: hardwired shifts and adder blocks. To facilitate the manipulation of the shift-and-add components, each component has been modeled as record with several variables in order to manipulate each detail within the constant multiplier. This modeling criterion supports integrating the proposed NMU algorithm for constructing shift-and-add multipliers. The hardwired shift is entitled as 'NodeCell' and the adder block is entitled as 'AdderCell'. The attributes of each cell is given in Fig. 8.12 and described in the following in detail.

The **'NodeCell'**:
**'Level'**- record defines the level of the node within the multiplier construction, e.g. for binary coefficient with cost of 5 non-zero terms, there are 3 levels of adders in the MA tree.
**'Flag'**- assert a feedback between nonconsecutive levels.
When 'Flag' record is set to 1, the 'Index' defines the index of the previous node used for feedback for odd number of nodes.
**'InWidth'** - multiplier input bit-width.
**'FillerRight'** - hard wired shifts corresponds to multiplication.
**'Sign'** - defines whether the left filler 'FillerLeft' are 1's (2's complement binary) or 0's (binary).

The **'AdderCell'**:
**'Flag' and 'Index'** - records are used for CSE, where 'Flag' indicates that there is a CSE at this adder and 'Index' defines the index of the adder used to feed through the CSE.
**'Type'** - record set the adder type used in implementing the constant multiplier tree, whether RCA or CSKA.
**'PosNeg'** - record defines whether the constant coefficient is positive or negative, i.e. to deal with binary or 2's complement binary.
**'SignedIn'** - defines the sign of the input bit stream whether positive or negative.
**'InWidthA', 'InWidthB', and 'OutWidth'** - defines the input bit-widths for both adder inputs and the adder output as well, respectively.

## 8.6. Summary

A novel encoding criterion for the reduction of the number of full-adders was presented in this chapter entitled as nested multiplication. The proposed criterion is valid for binary and CSD representations. The nested multiplication achieved around 25% reduction in the number of full-adders compared to the state-of-the-art criterion, the pseudo floating point. The percent of reduction in the NFAs decreases with the increase of the input bit-width. On the other hand, it increases

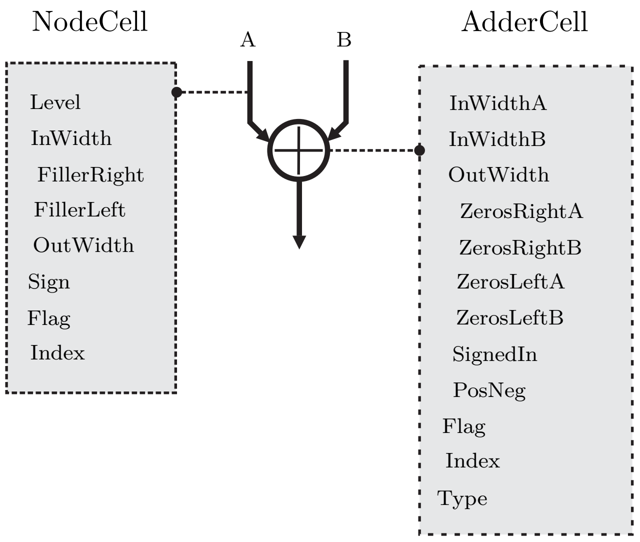


**Figure 8.12.:** RTL coding scheme for proposed NMU.

with the increase of the quantization bit-width. Further, an efficient RTL coding and implementation was delivered for the proposed criterion.

# 9. Digital Front End

## 9.1. Introduction

This chapter presents a novel power optimized implementation of a digital-front-end (DFE) for FM radio receivers. In order to epitomize the methodologies and algorithms proposed and developed in this work, a booming practical application for all-digital receivers has been chosen for implementation. The replacement of signal processing analog components with their digital counterparts has been motivated by their improved flexibility, reliability, and robustness. Therefore, digital-front-ends have been introduced in the radio receivers. This chapter presents a power efficient implementation of a narrow-band tunable digital-front-end for FM radio receivers. This work has been published in [4].

## 9.2. Design Specifications

Table 9.1 displays the FM radio receiver specifications. The sampling frequency is 390 MHz with a bandwidth of 20.8 MHz centered on a quarter of the sampling frequency 97.5 MHz. Each radio channel is 200 kHz. The receiver is implemented using 130nm silicon CMOS technology. A bit-stream stimuli of a $2^{20}$ length were integrated in the design procedure. The power spectral density (PSD) of the bandpass sigma delta modulator output is plotted, as shown in Fig. 9.1.

**Table 9.1.:** Receiver specifications.

| Parameter | Value |
|---:|:---|
| $f_s$ | 390 MHz |
| $f_c$ | 97.5±10.4 MHz |
| OSR | 975 |
| CMOS | 130nm Faraday UMC |
| $\delta f$ | 200 kHz |
| BW | 20.8 MHz |
| Stimuli | ✓ |

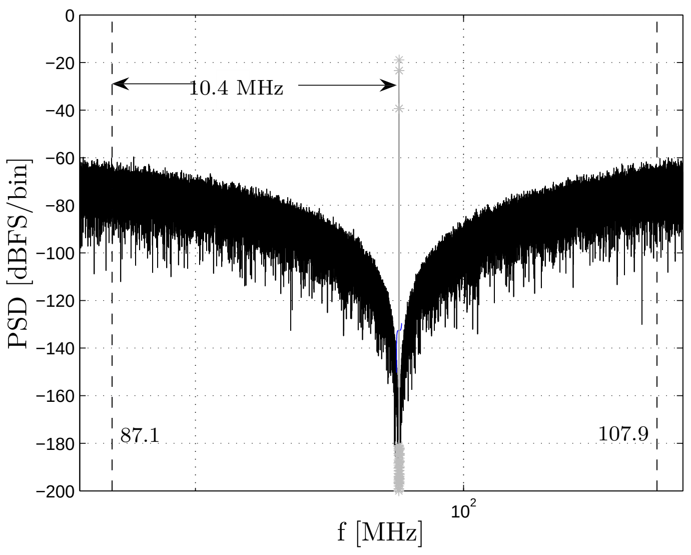
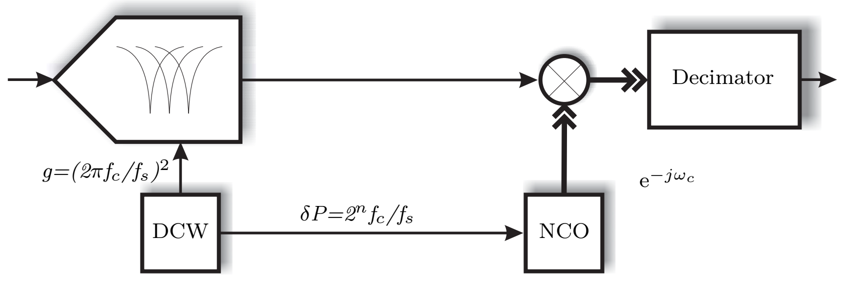


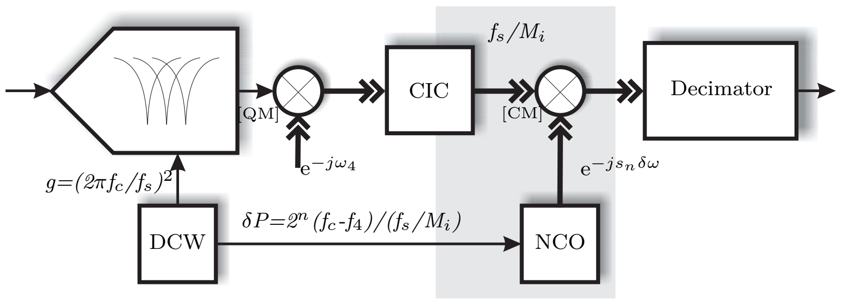
**Figure 9.1.:** PSD of a 4th order tunable BPΣΔM.

## 9.3. Proposed DFE

The FM radio receiver design considers a narrow-band tunable bandpass sigma delta modulator (BPΣΔM) placed early in the signal path, using a DFE for down conversion, channel selection, and sample-rate conversion.
A tunable 4th order BPΣΔM with a sampling frequency ($f_s$) of 390 MHz and a bandwidth (BW) of 20.8 MHz ranging from 87.1 MHz to 107.9 MHz centered around center frequency ($f_c$) 97.5 MHz for a channel width of ($\delta f$) 200 kHz. The modulator is implemented by continuous-time (CT) topology using CMOS 130nm technology. It provides an output resolution of more than 12-bit involving a single bit quantizer.
The BPΣΔM center frequency is tuned in integer steps by a control signal from the digital control word (DCW) through the local feedback coefficient $g$, as depicted in Fig. 9.2.a. The resonator local feedback coefficient $g$ controls the poles of the modulator transfer function and thus the modulator center frequency $f_c$. Further details about CT BPΣΔM are given in [117]. The modulator is followed by a digital-front-end (DFE) stage to translate the signal to baseband, reduce its sampling rate, and suppress the quantization and folded noise besides unwanted channels [118]. The conventional DFE and the proposed architecture are described in the following in detail.
The conventional DFE architecture shown in Fig. 9.2.a, consists of a digital numerically controlled oscillator (NCO) tuned to the channel of interest by a control signal from a digital control word (DCW), driving a digital mixer. The sample rate at the output of the mixer is equal to the high sample rate used within the ADC. The DCW determines the center frequency of the modulator as well as the digital-front-end through $g$ and $\delta P$, respectively. It translates $g$ to its correspond-



**(a)**



**(b)**

**Figure 9.2.:** DFE architectures (a) conventional (b) proposed, where $g$ is the BPΣΔM local feedback coefficient and $\delta P$ is the NCO frequency controlled word.

ing $\delta P$ so that the $f_c$ is identical for both, the BPΣΔM and the DFE. Thus, $g$ is the BPΣΔM local feedback coefficient. While, $\delta P$ represents the frequency controlled word (FCW) for the NCO. Detailed explanation for the FCW comes later in the section 5. $f_c$ is determined by $f_c$=$f_4 \pm s_n \delta f$. $f_4$ is a quarter of the sampling frequency, $\delta f$ is the 200 kHz frequency resolution and $s_n$ is an integer number for frequency selection ($s_n \in$ [-52:52]).
The conventional implementation consists of power hungry components running at high sampling rate. So, it is possible to tradeoff between an increased design complexity and a reduced sample rate for some blocks (and therefore reduced energy consumption), leading to the proposed architecture depicted in Fig. 9.2.b. The proposed down conversion is implemented using two mixers, a quadrature and a complex mixer. The quadrature mixer has a fixed oscillation frequency at a quarter of the sampling frequency ($f_4$=$f_s$/4). It translates the ΣΔM's output down to a complex intermediate frequency (IF). The complex mixer has a tunable oscillation frequency set by a NCO for channel selection, with a frequency resolution $\delta f$ of 200 kHz chosen in accordance with the used BPΣΔM. The complex mixer translates the complex IF signal band to the complex baseband for decimation (the double arrow in Fig. 9.2 represents the complex $I/Q$ signals). However, it is not sufficient to split the digital mixer to a quadrature mixer [QM] and a complex mixer [CM], and retain both of them running at the same high $f_s$. Therefore, a reallocation of one of the decimation stages in between the two mixers is proposed in order to trade-off CM complexity with lower $f_s$.
The decimation stage is designed using the MSD-toolbox given in chapter 7, opti-

mized by the algorithm given in chapter 5, and implemented using the VHDL IPs presented in chapter 7. The proposed design and implementation given in chapter 7 have been elaborated through this application. The decimation is achieved by a four-stage topology. The first stage, a cascaded integrator comb (CIC) decimator, is placed between the two down mixers. This allows the sampling frequency for the complex mixer and the NCO to be reduced to $f_s/M_i$, where $M_i$ is the decimation factor of the CIC stage. The Gray shaded area in Fig. 9.2.b is thus running at $f_s/M_i$ instead of $f_s$. All blocks are explained individually in more detail in the following sections in correlation with the previous chapters.

## 9.4. Digital Down Converter

The Digital Down Converter (DDC) is used to transform the RF signal into baseband signal. The conventional structure of a down mixer with tunable center frequency $f_c$ is shown in Fig. 9.3.a. Mixing the input signal $\cos\omega_c$ by the local frequency $e^{j\omega_c}$, translates the input signal to the baseband (exactly around dc) since they are identical. The NCO generating the local mixing frequency ($e^{j\omega_c}$) is usually implemented using ROM look-up tables (LUT), and a high access frequency to this structure is costly in terms of energy. Therefore, the approach followed here is to split $f_c$ (which is the center frequency and the local mixing frequency at the same time) into two components, a fixed and a tunable one according to

$$f_c = f_4 \pm s_n \delta f \tag{9.1}$$

where $f_c$ is the local center frequency, $f_4$ is quarter the sampling frequency, $\delta f$ is the oscillator tuning step size, and $s_n$ is an integer representing the channel number $\in \{-52 : 52\}$. Consequently, mixing the input signal $\cos\omega_c$ by $f_4$, translates the input signal to an IF location at $\omega_c - \omega_4$, as depicted in Fig. 9.3.b. Subsequently, complex mixing with the same tuning range but at a baseband instated of RF is proposed. In other words, the NCO needs to generate frequencies ranging from 200 kHz up to 10.4 MHz. While on the other hand, the conventional implementation needs a tuning range from 97.5 MHz up to 107.9 MHz. The conventional implementation requires a bulky ROM size for the NCO, more details and analysis for NCO is presented later on in section 9.5.
The corresponding mixer architecture for the proposed DDC is shown in Fig. 9.3.b. The sine and cosine sequences for an oscillation frequency of $f_4$ have a very simple structure: each term is either 0 or $\pm 1$. The multiplicands are thus 0 or $\pm 1$, which can be implemented using simple boolean operations, as shown in Fig 9.4. Therefore, it has a very power efficient implementation [119], [120]. A quadrature mixer is followed by a complex mixer as shown in more detail in Fig. 9.5.a and Fig. 9.5.b. The quadrature mixer is fed with an input signal oscillating at a quarter of the sampling frequency $f_4$ and runs at $f_s = 390$ MHz. The complex mixer is fed by the NCO which has a tunable oscillation frequency and runs at $f_s/M_i$, where $M_i$ is the decimation factor of the decimation stage between the two mixers.

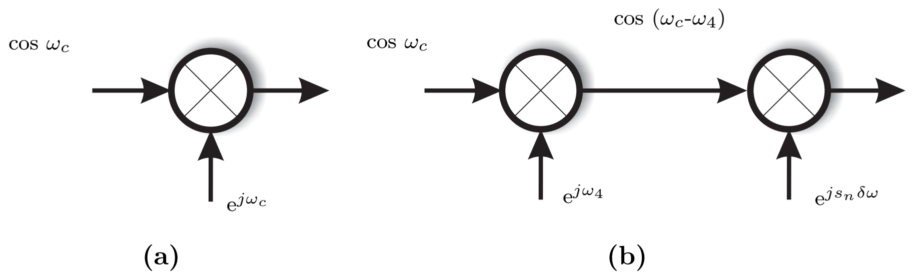


**Figure 9.3.:** Digital down conversion (a) conventional (b) proposed.

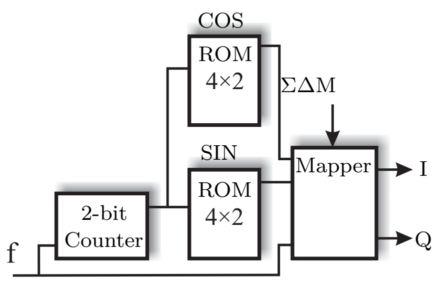


**Figure 9.4.:** Power efficient quadrature mixer for $f_4$.

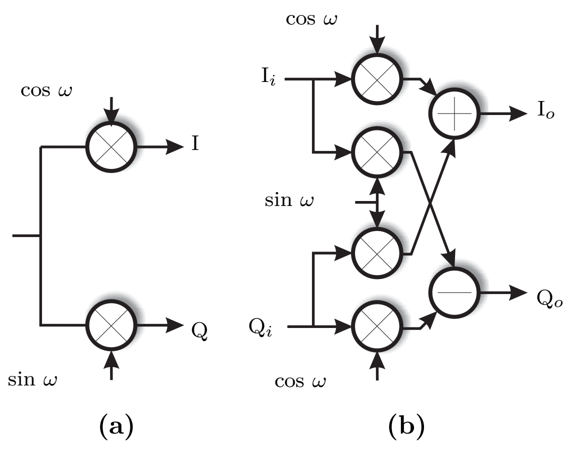


**Figure 9.5.:** Mixer architecture for (a) Quadrature and (b) Complex.

Between the two mixers a CIC decimation filter is placed which is explained in more detail in section 9.6.1.

## 9.5. Numerical Controlled Oscillator

Frequency translation is a common process in signal processing. Since signal processing is done in baseband, a translation between different frequency bands is of

great demand in communication systems for translation from very high frequencies to much lower frequencies. Frequency translation can be achieved in analog domain by phase locked loop (PLL) or in digital domain by numerically controlled oscillator (NCO). A direct digital synthesizer (DDS) consists of a NCO with digital-to-analog converter (DAC) and anti-aliasing filter (AAF). The DDS is out of the scope of this work. ROM-based NCO dissipates more power compared to the PLL but it has a very high tuning resolution, is extremely fast with continuous phase, and matched quadrature signals (for quadrature synthesizers) [121].
A detailed review for the NCO fundamentals can be found in [122], [123]. Besides, elementary definitions of performance quantities can be found in [124]. Moreover, a concise review is published in [122].
NCOs are categorized to ROM-based and ROM-less architectures [125]. Figure 9.6 summarizes the NCO categories. The ROM-less architecture is classified to CORDIC and polynomial approximation (high order $\geq 4$). The ROM-based NCO architecture can be classified according to their phase to single (SIN or COS) or quadrature (SIN and COS) phase. Practically, a quadrature phase NCO is more beneficial. Furthermore, a ROM-based NCO architecture can be classified according to the ROM complexity to symmetry (employing half, quarter and one eighth symmetry property of the sine or cosine), angular decomposition, sine amplitude compression and low order polynomial approximation. A recent related review is published in [125].
ROM is used for phase-to-amplitude conversion. ROM-less NCO architectures has an advantage of high performance and low power consumption, although, it deteriorates the tuning latency [125]. Therefore, ROM-less approach is not suitable for high frequency applications. Hence, this work considers ROM-based NCO.

## 9.5.1. Sinusoidal Symmetry NCO Topologies

An NCO is used to generate sinusoidal waveforms which are multiplied by the input complex data to implement the frequency mixing function [126]. The NCO tuning equation is given by (9.2).

$$f_c = \mathrm{FCW}\frac{f_s}{2^n} = \mathrm{FCW} \times f_{min}, \tag{9.2}$$

where $f_c$ is the desired frequency, FCW is the frequency control word, $f_s$ is the reference sampling frequency, $n$ is the PA bit-width, and $f_{min} = f_s/2^n$ is the frequency resolution of the NCO. The maximum generated frequency from a NCO is defined by the Nyquist criterion constraint as depicted in (9.3).

$$f_{max} = \frac{f_s}{2} \tag{9.3}$$

In the following two sub-sections two ROM-based NCOs using quarter and one eighth symmetry are presented and analyzed in detail.

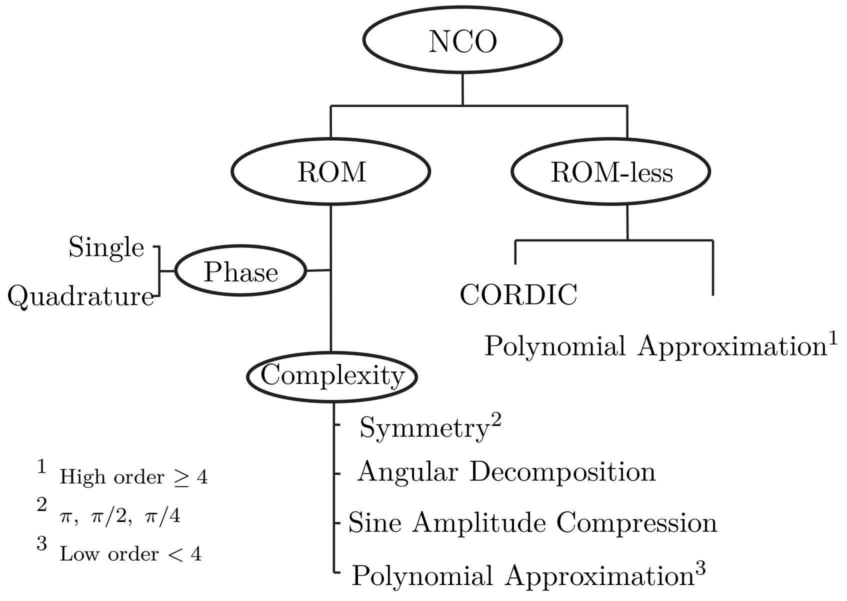


**Figure 9.6.:** NCO classifications.

### 9.5.1.1. $\pi/2$ Symmetry

A straight forward implementation of NCO with reduced ROM size can be achieved by employing the quarter symmetry ($\pi/2$) of the sinusoidal waveform. A $\pi/2$ NCO consists of phase accumulator (PA), single phase-to-amplitude converters ($\pi/2$ SIN ROM), and two complementer blocks, as shown in Fig. 9.7. The size of the ROM is $2^k \times m$. A $\pi/2$ NCO is outlined in [122] and implemented in VHDL [127] as

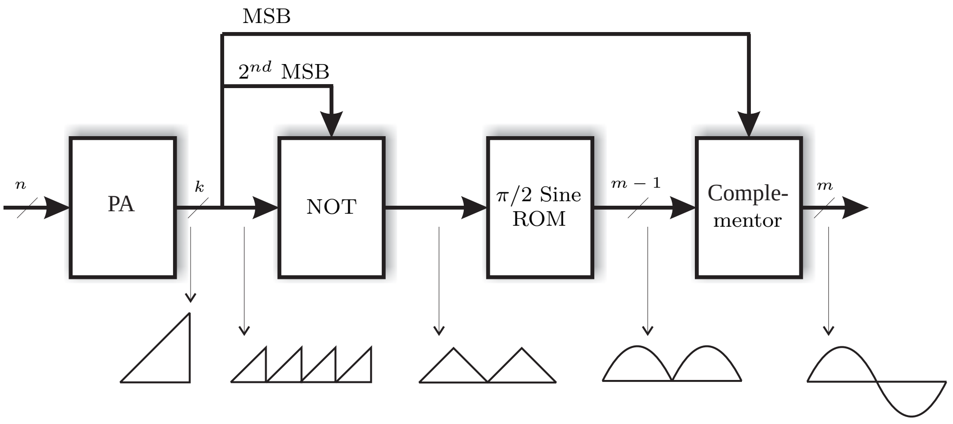


**Figure 9.7.:** Block diagram of the NCO implementation employing $\pi/2$ symmetry, where $n$ is the number of phase accumulator bits, $k$ is the ROM address bit-width and $m$ is the word-length of the ROM.

displayed in Fig. 9.7. This architecture has been used in both the conventional and the proposed DFE architectures for the preliminary implementation. The

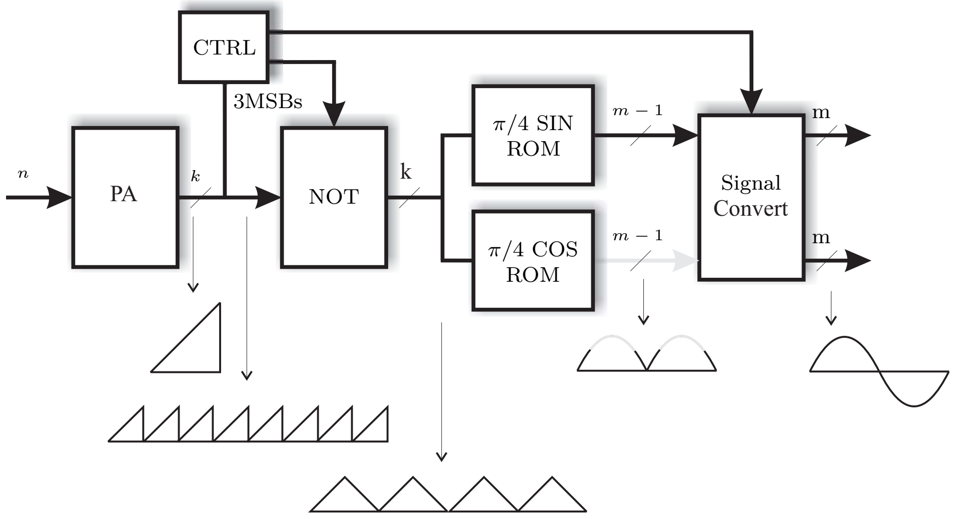


**Figure 9.8.:** Block diagram of the NCO implementation employing $\pi/4$ symmetry.

proposed DFE has achieved about 50% saving in power consumption using pi/2 NCO as given shown later in section 9.5.5. Nevertheless, more emphasis on NCO block in particular has been put on minimizing its power dissipation, which inspires the proposed work and given in the following sub-section.

#### 9.5.1.2. $\pi/4$ Symmetry

A typical implementation for quadrature NCO would be by employing the one-eighth symmetry ($\pi/4$) because it stores samples for both the sine and cosine waveforms. The block diagram of the quadrature NCO is given in Fig. 9.8, where $n$ is the number of phase accumulator bits, $k$ is the ROM address bit-width and $m$ is the wordlength of the ROM. The size of the ROM is $2^k \times m$. This architecture was proposed in [128]. The quadrature NCO consists of phase accumulator (PA), complementer (NOT), two phase-to-amplitude converters ($\pi/4$ SIN ROM and $\pi/4$ COS ROM), signal converter and control unit (CTRL). The signal converter consists of two multiplexers and two complementer circuits. The control unit uses the three MSBs (from PA output) to reconstruct the complete sine and cosine sampled waveforms from their one-eighth waveforms by detecting the sign, the quadrant and whether the amplitude is increasing or decreasing. The phase-to-amplitude converter is a ROM storing fixed samples for one-eighth sine and cosine waveforms. The waveform is accomplished by feeding the PA output to a phase-to-amplitude converter ROM. Consequently, the ROM provides the corresponding waveform amplitude value.
Since, the sine and cosine samples are stored with limited bit-width besides the big phase step for generating large frequencies, the enhancement is required to improve the system performance. Otherwise, a bulk ROM is used leading to a power hungry NCO architecture.

## 9.5.2. Performance Enhancement Criteria

ROM-based NCOs offer a very fine tuning resolution and are extremely fast while providing continuous phase, as opposed to ROM-less architectures and matched quadrature signals (for quadrature synthesizers) [121], [129]. On the other hand, they also consume more power and may exhibit large spurious tones [129]. The latter may arise due to a truncation in the phase accumulator, a compression of the ROM and a finite precision of the sinusoidal samples [129]. Therefore, the design of a high performance and power optimized NCO for a personal communication system poses a challenging task. Several methodologies for improving the performance of ROM-based NCOs were presented in literature, e.g., amplitude dithering [123], phase dithering [129], [130], [131] and piece-wise linear approximation [121], [132]. Other criteria target reducing the ROM size, e.g., sinusoidal symmetry [125], [133] and ROM compression [129].

### 9.5.2.1. Dithering

Dithering is the process of distributing or randomizing the noise content over the signal spectrum by multiplying the noisy signal by a random noise as Linear Feedback Shift Register (LFSR). It could be performed on phase, amplitude and approximation.

### 9.5.2.2. Linear Approximation

A linear approximation (LAP) methodology was proposed in [121], [134] for reducing spurious signal content for $\pi/2$ reduced size ROM NCO. Linear approximation enhances the performance by 12 dBc per address bit-width $k$ [121]. Hence, SFDR $= 12 \times (k + s)$. The linear approximation function is given by (9.4) and is implemented as shown in Fig. 9.9. Linear approximation provides 12 dB per bit [121], instead of 6 dB for conventional architectures employing symmetry only. The proposed NCO architecture is shown in Fig. 9.10 employing $\pi/4$ symmetry and piece-wise linear approximation.

$$f(x) = f_1 + \frac{f_2 - f_1}{x_2 - x_1}(x - x_1) \tag{9.4}$$

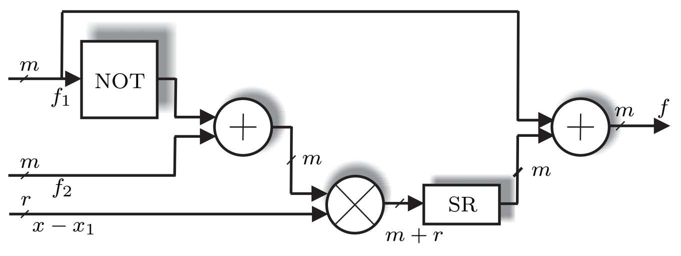


**Figure 9.9.:** Piecewise linear approximation implementation.

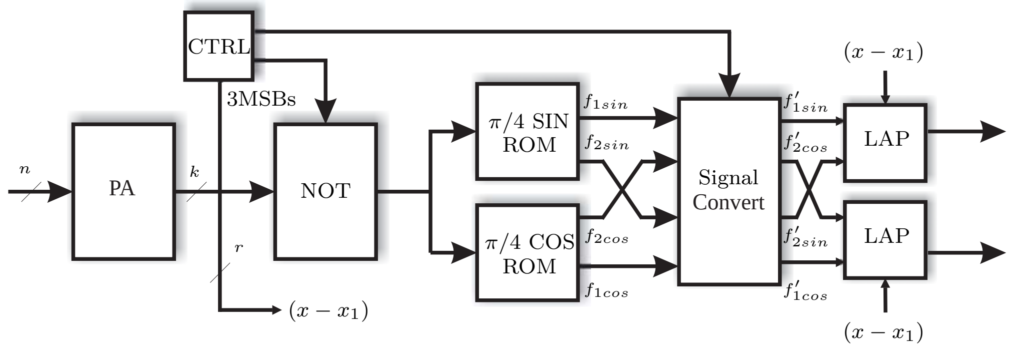


**Figure 9.10.:** Proposed quadrature NCO employing $\pi/4$ symmetry and linear approximation.

where

| | |
|---|---|
| $f_1$ : | exact (current) ROM value |
| $f_2$ : | adjacent (next) ROM value |
| $x$ : | actual phase value (between $x_1$ and $x_2$) |
| $x_1$ : | phase value corresponding to $f_1$ |
| $x_2$ : | phase value corresponding to $f_2$ |
| $x_2 - x_1$: | the difference in the $x$-component of the two points $((x_1, f_1)$ and $(x_2, f_2))$ (constant value $= 2^{(n-k-3)}$) |
| $x - x_1$ : | difference of the actual phase value to the phase value of the point $(x_1, f_1)$ |

## 9.5.3. Modeling and Analysis

ROM-based NCOs are already well addressed in literature. However, deriving the optimal parameter settings is not. Although performance enhancement criteria do exist, their combination and the evaluation of the overall performance do not. Hence the abstract high-level model for ROM-based NCOs shown in Fig. 9.11 was developed to speed up the design process, boost the design flexibility, evaluate optimal design parameters and improve the verification process by means of serving as a reference model. Symmetry improves the performance of an NCO by 6 dBc per $s$ symmetry bit, which is uncorrelated to either dithering or approximation.

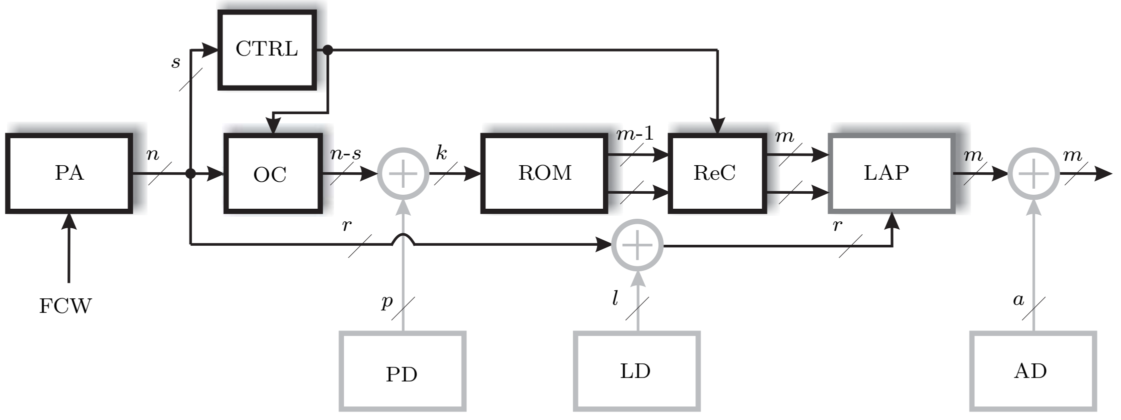


**Figure 9.11.:** ROM-based generic reference model.

Thus, it is not further considered. A comprehensive analysis for dithering and approximation was performed and will be presented in the following. Although the SystemC model was developed as a time loosely model, i.e., no inherent gate delays are defined, the model was implemented as a synchronous system, i.e., the clock is used as a reference for updating and/or transferring the data among model components. Consequently, the run-time of the simulation process is considerably fast while the results are quite accurate. Furthermore, being able to modify the design parameters at the top-level improves the design flexibility in that multiple design aspects can be handled.
The model merges various performance enhancement criteria, e.g., symmetry, dithering and linear approximation. It is used to evaluate the possible performance boost of NCOs using linear approximation which is not yet considered in literature.
The generic reference model for a quadrature ROM-based NCO shown in Fig. 9.11 was developed using Matlab and SystemC. The corresponding parameters are summarized in Table 9.2. The model employs state-of-the-art methodologies for improving the performance of an NCO, i.e., sinusoidal symmetry, phase dithering (PD), approximation dithering (LD), amplitude dithering (AD) and piece-wise linear approximation (LAP). An NCO employing $\pi/4$ symmetry, i.e., $s = 3$, serves as an illustrative example in the following analysis. Generally, symmetry improves the performance of an NCO by 6 dBc per symmetry bit $s$, independent of whether linear approximation or any dithering technique is applied. Moreover, the NCO provides a quadrature output, thus improving the performance by 6 dBc for the same parameter settings (excluding any other performance enhancement criteria). Simulations were performed using a reference clock frequency $f_s = 390$ MHz and generated center frequencies within the range $f_c \in [1 - 48]$ MHz in steps of 3 MHz. These design specifications were chosen in accordance with the proposed DFE. The performance is evaluated by means of the spurious-free dynamic range (SFDR) [125].
Based on the derived model, the performance boost of NCOs using a LAP is analyzed in the following which is not yet addressed in the literature. Subsequently, dithering techniques are applied in order to evaluate whether a further

**Table 9.2.:** Design parameters

| Parameter | bit-width | |
|---|---|---|
| $a$ | amplitude dithering | |
| $k$ | ROM address | |
| $l$ | approximation dithering | |
| $m$ | ROM word-length | |
| $n$ | phase accumulator | |
| $p$ | phase dithering | |
| $r$ | approximation | |
| s | No symmetry | 0 |
| | $\pi$ symmetry | 1 |
| | $\pi/2$ symmetry | 2 |
| | $\pi/4$ symmetry | 3 |

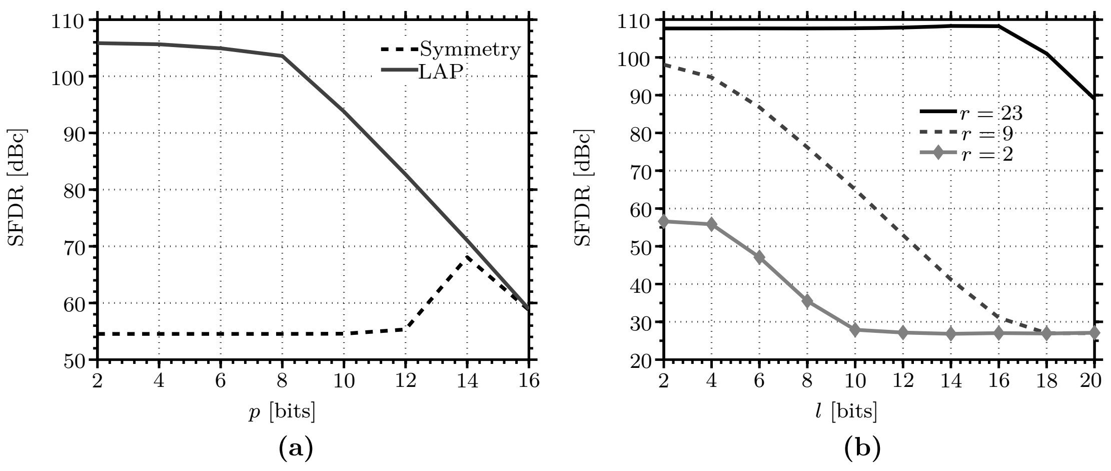


**Figure 9.12.:** Effect of (a) phase dithering $r = n - (k + s)$ (b) approximation dithering on performance of the NCO at $f_s = 390$ MHz, $n = 32$, $s = 3$, $k = 6$, $m = 16$, and NFFT = 4096.

performance boost becomes possible. However, the model may also be used for evaluating optimum parameter sets for classical architectures, i.e., without LAP while applying any dither techniques. Since such results are readily made available in the literature, the focus is on LAP architectures.

#### 9.5.3.1. Dithering

Phase dithering whitens the spectrum of the synthesized signal while increasing the noise floor [129], [131]. Thus, it improves the SFDR by 12 dBc while the recommended PD bit-width is calculated by

$$p = (n - s) - k \tag{9.5}$$

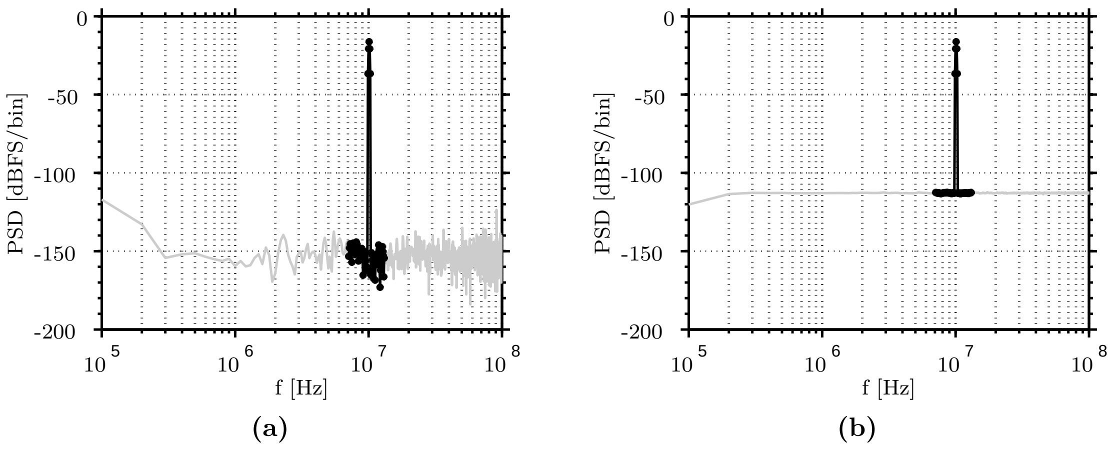


**Figure 9.13.:** Effect of (a) linear approximation ($r = 9$) without dithering (b) linear approximation with approximation dithering ($r = 9$, $l = 1$) at $f_s = 390$ MHz, $n = 32$, $s = 3$, $k = 6$, $m = 16$ and NFFT= 4096.

Increasing the PD bit-width, considering LAP, deteriorate the NCO performance, as shown in Fig. 9.12.a. In other words, using LAP exclusively outperforms the NCO performance without and/or with PD. Hence PD is not required with LAP. Although the least significant phase bits are used within the approximation, dithering does not reveal any improvement in the performance. Figure 9.12.b shows the effect of the approximation dithering at different $r$ bits. Reducing $r$ and increasing $l$ deteriorate the performance. The effect of approximation dithering (LD) on the performance of the NCO can be seen in Fig. 9.13.b. It is obvious that dithering spreads the noise but increases the noise floor, consequently, reduces the SFDR.
The ROM word-length is increased to $m + a$ instead of $m$ when using amplitude dithering (AD). This in consequence increases the hardware complexity and consumes more power. Thus AD was not considered in the following system analysis.
In conclusion, using piece-wise linear approximation excludes the usage of dithering (phase, approximation and amplitude). Further, the excessive increase of approximation bit-width increases the complexity versus insignificant improvement in the performance.

#### 9.5.3.2. Linear Approximation

Linear approximation enhances the performance by 12 dBc per address bit-width $k$ [121]. Hence, SFDR $= 12 \times (k + s)$. Although this formula is true, it is not absolute valid for any range of $k$. The proceeding simulations and analysis reveal bounding constraints to achieve the desired SFDR performance.
Figure 9.14.a shows a linear improvement in the performance with respect to the address bit-width $k$ up to a certain address bit-width value, as marked by the circle. The approximation bit-width was set to $r = n - (k + s)$ which is the maximum allowed $r$, i.e., the remaining bits after truncating the $s$ and $k$ bits. Figure 9.14.c

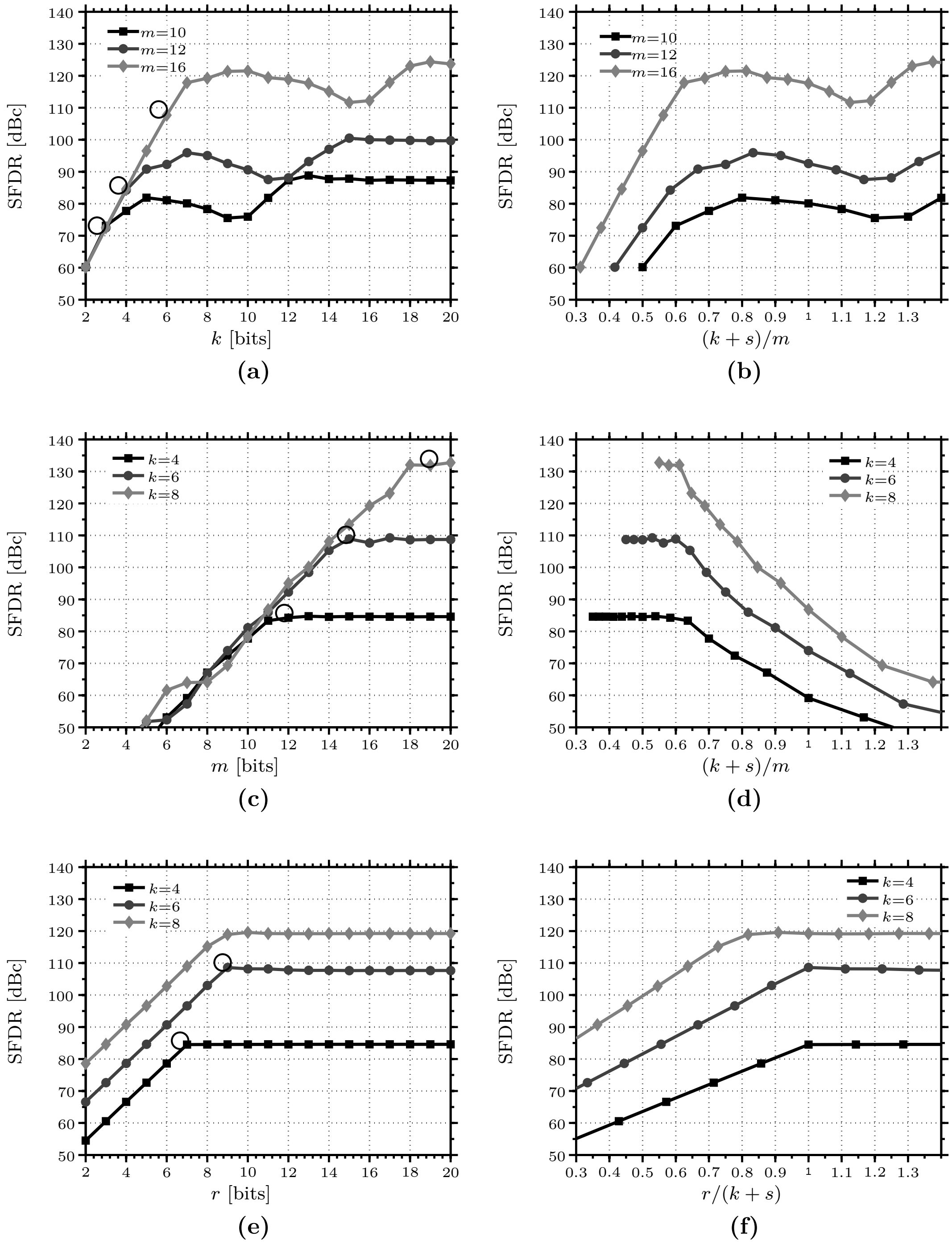


**Figure 9.14.:** System performance relative to (a) ROM address bit-width $k$ (b) $(k+s)/m$ sweeping $k$ (c) resolution (d) $(k+s)/m$ sweeping $m$ (e) approximation bit-width (f) $r/(k+s)$ at $f_s = 390$ MHz, $n = 32$, $s = 3$, and NFFT $= 4096$.

shows an improvement in the performance with respect to the ROM word-length $m$ up to a certain resolution bit-width, with the approximation bit-width again set to $r = n - (k + s)$. Figure 9.14.e shows an improvement in the performance with respect to the approximation bit-width $r$ up to a certain approximation bit-width value (at $m = 16$). It should be noted that at $k = 8$, $r_{min}$ does not follow the same criterion as for $k = 4$ or $k = 6$. The preceding simulation results show that increasing the design parameters does not enhance the performance necessarily. Thus, the design parameters should be within a certain range to accomplish the expected performance improvement.
Figure 9.14.b (at $k = 4, 6$) show that SFDR$= 6\,(k + s) + 6r$ since $r < (k + s)$. Hence, the performance depends on $k, s$, and $r$. Further, SFDR$= 12\,(k + s)$ provided that $(k + s)\,/m \leq 0.6$ and $r \geq (k + s)$. In order to correlate the design parameters and evaluate their effect on the NCO performance in more detail, the SFDR was plotted relative to the ratios $(k + s)/m$ and $r/(k + s)$ as shown in Fig. 9.14.(b,d,f), respectively.
The derived simulations reveal that no further performance boost of the SFDR is to be expected while increasing $k$, $m$ or $r$ randomly. Nonetheless, the proposed bounds should be considered for high performance and low complexity NCOs.

$$\text{SFDR} \approx \quad 12\,(k+s) \iff \frac{k+s}{m} \leq 0.6 \;\wedge\; r \geq k+s, \tag{9.6}$$

$$\text{SFDR} \approx \quad 6\,(k+s)+6r \iff \frac{k+s}{m} \leq 0.6 \;\wedge\; r < k+s. \tag{9.7}$$

### 9.5.4. Proposed Design Scheme

The SFDR characteristic is illustrated in Fig. 9.15.a for various values of $m$ and $k$. It should be noted that, these SFDR curves represent the cross sections indicated by the dashed lines in Fig. 9.15.b.
The major outcome of the simulation results is, however, that the values of $k$, $m$ and $r$ can be minimized according to 9.9- 9.10

$$k_{min} = \lfloor\frac{\text{SFDR}}{12}\rceil - s, \tag{9.8}$$

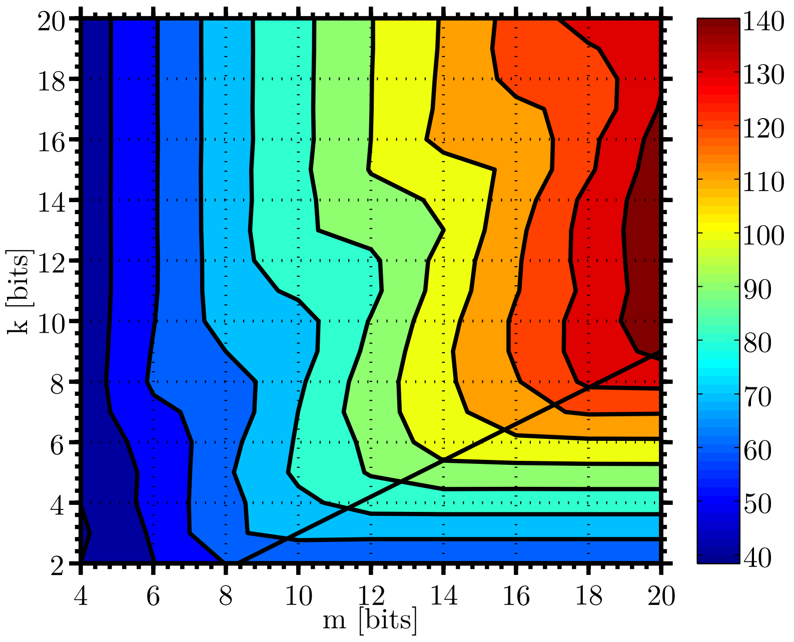
$$m_{min} = \lceil\frac{k_{min}+s}{0.6}\rceil, \tag{9.9}$$

$$r_{min} = k_{min} + s, \tag{9.10}$$

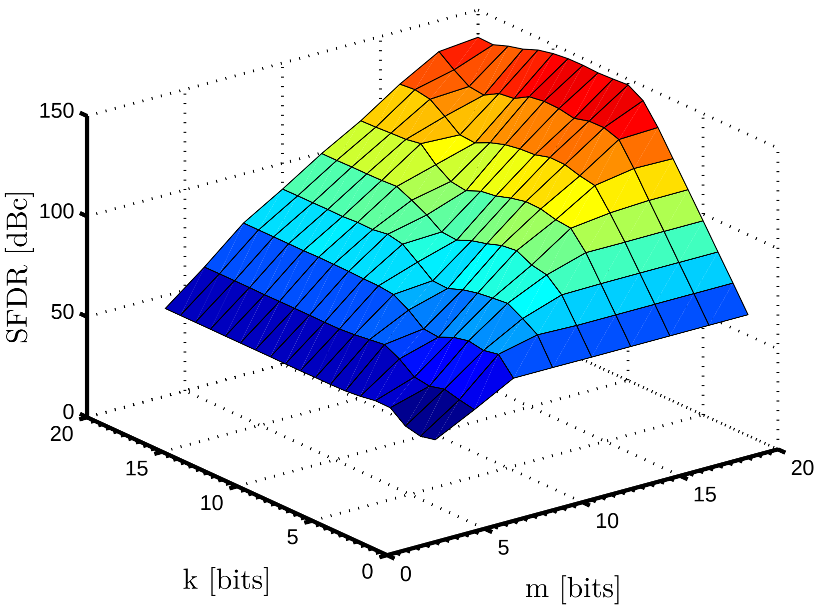
if-and-only-if

$$\frac{k+s}{m} \leq 0.6, \tag{9.11}$$

$$r \geq k+s. \tag{9.12}$$



(a)



(b)

**Figure 9.15.:** Mean value of the SFDR (a) surface plot (b) contour plot (simulation settings: $n = 32$, $s = 3$,$f_s = 390$ MHz, $f_c$ swept from 1 MHz to 48 MHz in steps of 3 MHz).

**Table 9.3.:** NCO design specifications.

| Parameter | Value |
|---|---|
| $f_s$ | 390 MHz |
| $f_c$ | 87.1 MHz |
| SFDR | 80 dB |
| SNR | 72 dB |

**Table 9.4.:** NCO Design parameters.

| | $\pi/2$ | $\pi/4$ | $\pi/4$+LAP |
|---|---|---|---|
| $n$ | 20 | 16 | 16 |
| $k$ | 10 | 10 | 4 |
| $m$ | 12 | 12 | 12 |
| $r$ | – | – | 6 |

### 9.5.5. Power Analysis

Precise power simulation and analysis were carried out for three NCO architectures. The first architecture is employing quarter symmetry and denoted by $\pi/2$ NCO. The second architecture is employing the one-eighth symmetry and denoted by $\pi/4$ NCO. The third and last architecture is employing the one-eighth symmetry as well as linear approximation, as shown in Fig. 9.10, and denoted by $\pi/4$+LAP. The power simulations have been accomplished using Synopsys PrimeTime PX as a power compiler and Cadence SimVision as an event driven simulator. The bit-level switching activity has been dumped into VCD file for accurate power simulation. The $\pi/4$ and $\pi/4$+LAP NCO architectures were modeled in a generic structural VHDL. Whereas, the $\pi/2$ NCO was imported from [127]. The NCO design specification is given in Table 9.3. The NCO implementation parameters are given in Table 9.4 for the three architectures. The $\pi/2$ NCO requires larger phase accumulator bit-width to sustain the same signal quality defined by the design specifications for the signal-to-noise ratio (SNR) and the spurious-free dynamic range (SFDR). The power analysis reveals a 60% reduction in power consumption by the proposed $\pi/4$+LAP NCO architecture compared to the conventional $\pi/4$ NCO. In addition to 50% reduction in power consumption compared to the conventional $\pi/2$ NCO, as exhibited in Fig. 9.16.

## 9.6. Decimation Filters

The decimator has to downsample the signal by an overall factor of 975. It is more efficient to implement the sample rate reduction in a series of decimation filter stages than using a single decimation stage [120]. The decimation filter stage topology has been designed and optimized using the MSD-toolbox, presented in chapter 7. The MSD-toolbox optimizes the number of decimation stages, the decimation factor at each stage, and the filter coefficients. The resulting filter uses a 4-stage topology with decimation factors of $[15, 13, 5, 2]$ respectively. A minimum number of coefficients are used and the coefficients are represented in power-of-two format where possible. In the BP$\Sigma\Delta$M the decimation is achieved by a combined factor of 2×OSR because of both real and imaginary (complex) channels [119]. Therefore, the decimation factor is defined by 1950 (2×OSR). Due to the high

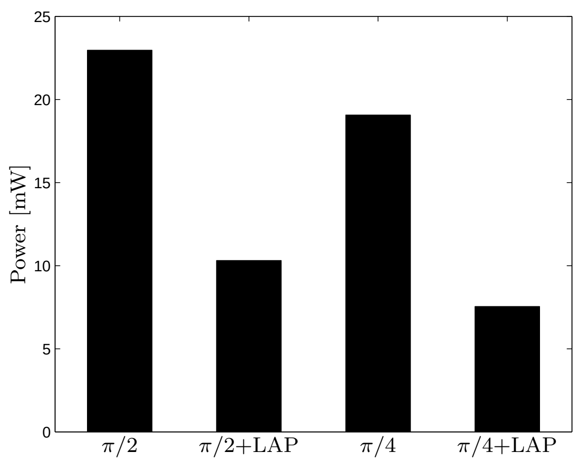


**Figure 9.16.:** Total power consumption in NCO architectures for $f_s = 390$ MHz, $f_c = 78.1$ MHz, $m = 12$, $k = 10$, $n = 20$ for $\pi/2$ NCO and $n = 16$ for $\pi/4$ NCO.

input sampling frequency, a CIC filter stage is used for the first decimation stage whereas the following stages are realized as polyphase decimators.

### 9.6.1. CIC Decimation Filter

Cascaded integrator-comb (CIC) filters have been used as a power efficient decimation filter especially for high sampling frequencies because of the absence of multipliers and the high stop band attenuation [105]. Detailed analysis of CIC filter and its compensation filter is given in subsections 7.2.3.3 and 7.7.3.
The decimation factor of the CIC filter needs to be chosen carefully to keep the effort (signal attenuation and compensation) for this filter reasonable. As the pass band edge gets closer to the null, the CIC filter attenuates more and the compensation filter needs to provide more correction. Since the passband ($f_{pb}$) has to be at 10.4 MHz (52×200 kHz) and the nulls of the CIC filter are located at multiples of $f$=1/$M$, with M the decimation factor. Therefore, the decimation factor has to be small in order to have a low order correction filter. With a decimation factor of 5, the first null is located at $f_s/5 = 78$ MHz. Therefore, the specifications for the sine-based correction filter are relaxed with an order of 11-taps. However, for a decimation factor of 15 the first CIC null will be located at $f_s/15 = 26$ MHz. It would require a sine-based correction filter with an order of 31-taps. Despite the higher order correction (3 times higher), there is still a 4-dB loss in the signal peak power, shown between the arrows in Fig. 9.17. The design choice was thus to implement a 5-stage CIC filter with a decimation factor of 5 and an 11-taps FIR correction filter for this stage.

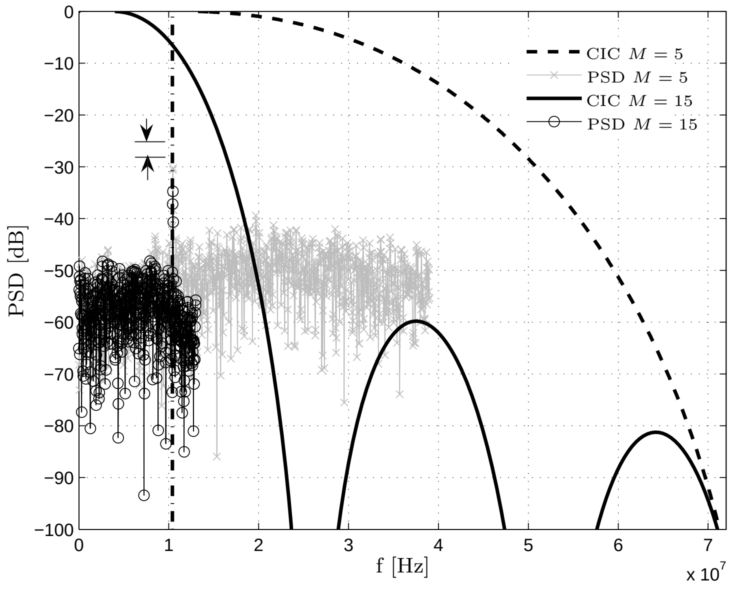


**Figure 9.17.:** CIC filter and decimation factor effect on intermediate response.

## 9.6.2. FIR Polyphase Decimation Filter

The remaining three filter stages are implemented as polyphase filters [91]. The polyphase decomposition offers significant savings in the computation and hardware effort compared to conventional structures. This is because of the fact that the (implicit) decimation takes place before the low-pass filtering in polyphase structures [91]. The design details are presented in Fig. 9.18, where $N$ is the filter order, $Q$ is the quantization bit-width of the filter coefficients, and $M$ is the decimation factor. PPD refers to polyphase decimation and HB refers to half-band. As usual, the correction filter is placed at the end of the signal path as shown in

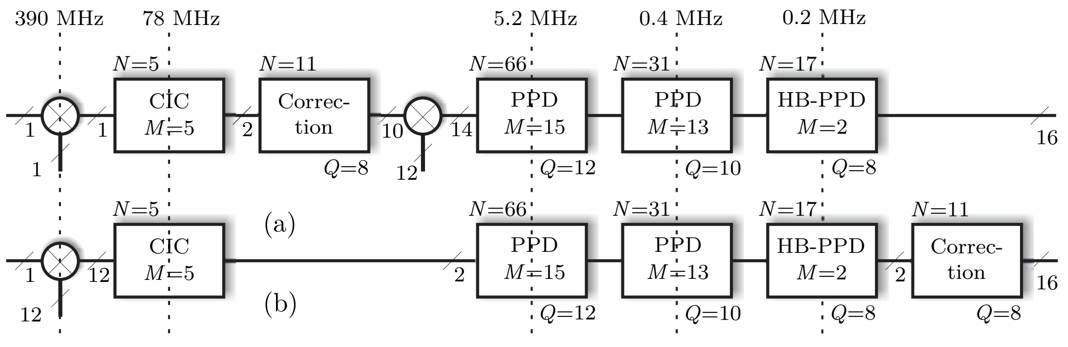


**Figure 9.18.:** Detailed design (a) proposed (b) conventional.

Fig. 9.18.b by the conventional DFE. However, this is not possible for the proposed DFE as the signal of interest is at the IF range before mixing to the baseband. So, the correction filter is placed just after the CIC filter as shown in Fig. 9.18.a.

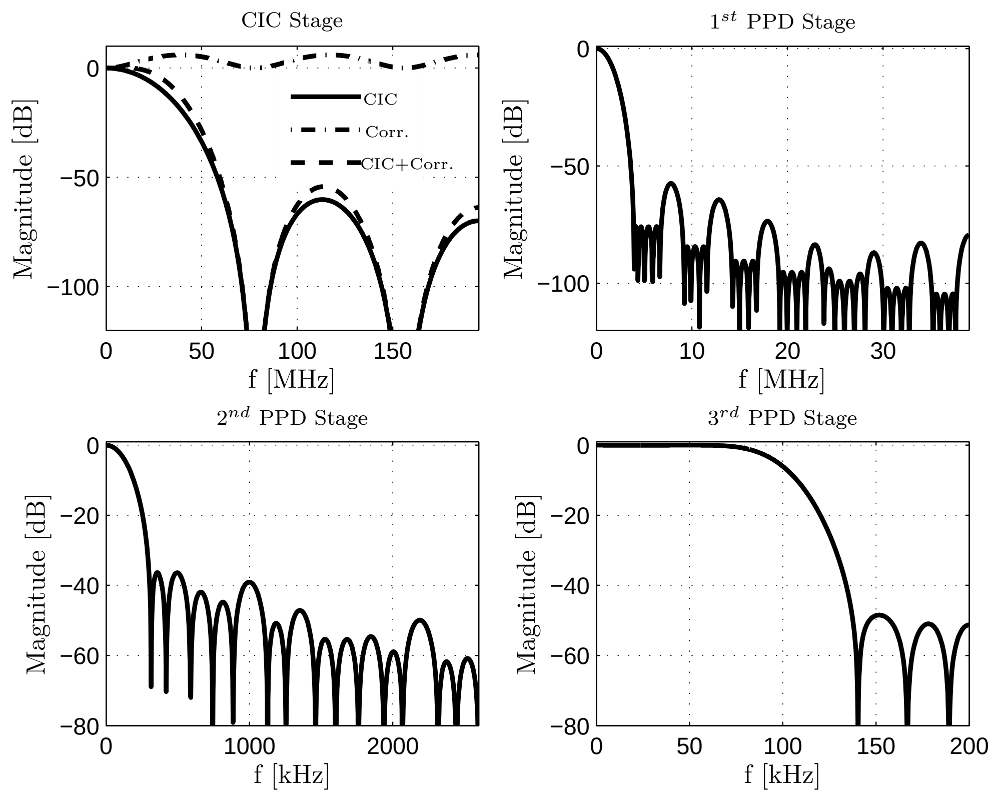


**Figure 9.19.:** Decimation filters frequency responses.

The individual frequency responses of the four stage decimation filters are shown in Fig. 9.19.

## 9.7. Power Simulation and Analysis

The conventional as well as the proposed DFE architectures have been implemented using the VHDL IPs described in chapter 4 using the design procedure and toolchain presented in the same chapter as well. The design specifications are shown in Table 9.5. Table 9.6 summarizes the total power consumption and the power consumption distribution in the conventional and the proposed architectures involving both the $\pi/2$ as well as $\pi/4$+LAP. A remarkable reduction in power consumption of about 50% has been achieved by the proposed architecture employing the $\pi/2$ NCO and about 60% employing the $\pi/4$+LAP NCO compared to the conventional architecture.
It has to be noted that, part of this work were published in [4] involving the $\pi/2$ NCO architecture. Therefore, the following discussion compares the proposed DFE employing the $\pi/4$+LAP NCO implementation.
The proposed architecture shows 95% savings in the power consumption in the

**Table 9.5.:** Design Specifications

| Parameter | Value |
|---|---|
| Sampling frequency $f_s$ | 390 MHz |
| Oversampling ratio $OSR$ | 1950 |
| Bandwidth $f_B$ | 100 kHz |
| Decimation stages $K$ | 4 |
| Decimation factors $M$ | [5 15 13 2] |
| Decimation filter orders $N$ | [5 66 31 17] |
| Quantization bit-width $Q$ | [- 12 10 8] |
| NCO LUT size | $2^4 \times 12$ |
| Output resolution | 16-bit |

**Table 9.6.:** Power Simulation Results

| | Conventional - mW | Proposed ($\pi/2$) - mW | Proposed ($\pi/4$) - mW |
|---|---|---|---|
| NCO | 15.9 | 2.69 | 0.89 |
| Mixer | 0.19 | 2.5[CM]+0.02[QM] | 2.5[CM]+0.02[QM] |
| CIC | 2.77 | 2.15 | 2.14 |
| Correction | 0.705 | 0.008 | 0.007 |
| PPD | 0.62 | 1.96 | 1.97 |
| Total | 19.52 | 10.12 | 8.32 |

NCO which is due to the fact that the NCO runs at 78 MHz in the proposed design rather than 390 MHz. In addition, the reduction in the NCO ROM size by employing the linear approximation. Furthermore, the phase accumulator wordlength $n$ is 14-bit rather than 24-bit. This is due to the fact that practical limitations restrict the maximum NCO output frequency to about 0.33 of the clock frequency [123] whereas Tierney has considered frequencies up to 0.25 of the clock frequency [135]. The longer word length of the phase accumulator $n$ within the conventional architecture is necessary because the phase ratio 0.278 $f_c/f_s = 107.9/390$ is higher than 0.25, since otherwise a lot of spurs are caused in the signal spectrum [135]. The power consumption of some blocks in the proposed architecture is increased compared to the conventional one, especially for the PPD and the mixer. This is due to the considerably larger word length as depicted in Fig. 9.18. The complex mixer output word length is truncated to 14-bit as shown in Fig. 9.18.a, though preserving the same in-band noise (IBN) level. Nonetheless, the proposed architecture offers a reduction in the total power consumption of about 60%, as the NCO is the dominant source of power consumption in both designs.

## 9.8. Summary

An advantageous implementation of a power efficient narrow-band tunable digital-front-end for digital FM radio receivers employing continuous time bandpass sigma delta modulators was presented in this chapter. About 60% reduction in power dissipation has been achieved by the proposed architecture.
The proposed digital-front-end considers splitting the digital mixer up to two mixers, a quadrature and a complex mixer. Moreover, one of the decimation stages is transposed between the two mixers with an appropriate decimation factor. Consequently, the reference clock of the numerically controlled oscillator is reduced in addition to the phase accumulator bit-width. Furthermore, a linear approximation criterion was integrated into a quadrature numerically controlled oscillator employing $\pi/4$ symmetry was proposed. The proposed numerically controlled oscillator achieved more than 50% reduction in power consumption compared to the conventional architecture.

# 10. Conclusion and Outlook

This dissertation has presented novel power optimization methodologies for digital FIR filters at the system level (algorithmic and architectural) and RTL level. Contributions of this dissertation can be seen in the three different aspects (a) computation time, (b) encoding scheme (c) block rearrangement.

**Computation time** - The computation time consumed by systematic/heuristic solvers for the filter optimization problem has been reduced remarkably, up to a factor of 400, compared to the state-of-the-art algorithm, by an extensive reliable pre-processing analysis on filter coefficients. The proposed algorithm together with the developed heuristic solver contributes to both the speed and the efficiency of the optimization process. The improvement of the computation time is due to:

- Bounding the search space for candidates for each coefficient by upper and lower bound.
- Introducing two states during computation, the No-Change and the Break-Off states, in order to terminate the computation at a certain condition.
- Monotonous incremental steps for candidates for each coefficient.
- Presorted coefficient sets according to several allocation schemes.

In addition, the improvement of the optimization results is due to:

- Multiple optimization iterations which mimic the backward and forward behavior of the tree structure of the conventional approach.
- Increasing the stopband attenuation by 3-dB in the initial (unoptimized) filter provides further flexibility in the optimization constraints.
- Unconstrained number of non-zero terms for candidates.

**Encoding scheme** - The number of full-adders used within the constant multiplier has been reduced considerably compared to the state-of-the-art scheme for binary and CSD representations. This reduction is due to an efficient encoding scheme for the internal shift-and-add operations within the constant multiplier which employs nested multiplication.

**Block rearrangement** - A power aware scheme has been developed to facilitate the combination of direct-form and transposed-form filter topologies within a multi-stage decimation filter according to the decimation factor, filter order, and input bit-width in a specific stage. It revealed that direct-form is recommended for small input bit-width stages. The combination of different topologies reduces the power dissipation in the decimation filter by approximately 15% compared to the conventional architecture.

**Application** - The proposed and developed power optimization methodologies in this dissertation have been combined to emphasize their contribution in the reduction of dissipated power for an FM digital radio receiver. A remarkable reduction in power dissipation by 60% compared to the conventional architecture for the digital front end has been achieved due to splitting the mixing stage to quadrature and complex mixers, in addition to placing a decimation stage between the two mixers. Further, a power optimized numerical controlled oscillator was designed and implemented for this application employing one-eighth symmetry and piecewise linear approximation.

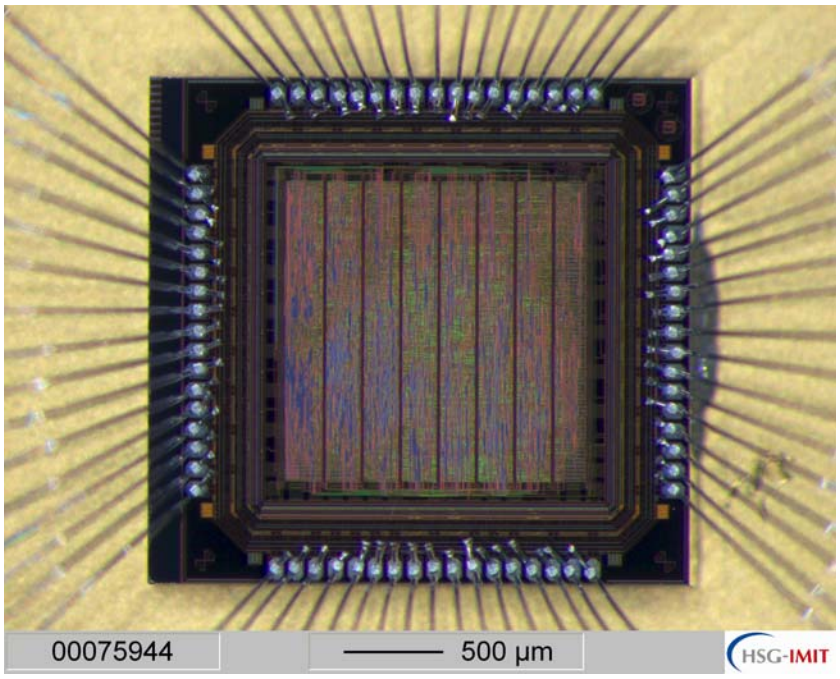


**Figure 10.1.:** I2MR decimation filter die-photo.

## Outlook

The power optimization methodologies proposed and developed in this work will be of a distinct value for taping-out an ASIC for a Sigma Delta ADC. A magnetic field sensor combing a feedback gyroscope Sigma Delta is one of many applications which integrates a digital FIR decimation filter. An initial attempt for taping-out a gyroscope sensor is carrying on. The design specification is given in Table 10.1. The design is fabricated using the CMOS XFab 0.35$\mu$m high density and low threshold technology node. The die-photo is shown in Fig.10.1. The implementation parameters determined using the MSD toolbox are presented in Table 10.2. The preliminary results without inserting IO PADs is shown in Table 10.3.

**Table 10.1.:** Design Specs.

| Parameter | Value |
|---|---|
| $f_s$ | 2 MHz |
| OSR | 50 |
| $W_i$ | 5-bit |

**Table 10.2.:** Design Parameters Evaluated using MSD-toolbox.

| Parameter | Value |
|---|---|
| $k$ | 3 |
| $M$ | [5,5,2] |
| $W_o$ | 16-bit |
| $N$ | [38,57,19] |
| $Q$ | [16,16,10] |
| Topology | TF PPD |

**Table 10.3.:** Place-and-route Implementation Results.

| | Conventional | Proposed |
|---|---|---|
| Power [mW] | 5 | 0.9 |
| Area [mm$^2$] | 4.0 | 2.5 |
| SNR [dB] | 78 | 78 |

# Bibliography

[1] A. Shahein, M. Becker, N. Lotze, M. Ortmanns, and Y. Manoli, "Optimized Scheme for Power-of-two Coefficient Approximation for Low Power Decimation Filters in Sigma Delta ADCs," in *Proc. IEEE Midwest Symposium on Circuits and Systems (MWSCAS'08)*, Aug. 2008, pp. 787–790.

[2] A. Shahein, Q. Zhang, N. Lotze, and Y. Manoli, "A Novel Hybrid Monotonic Local Search Algorithm for FIR Filter Coefficients Optimization," *IEEE Transactions on Circuits and Systems—Part I: Regular Papers*, vol. 59, no. 3, pp. 616–627, Mar. 2012.

[3] A. Shahein, M. Becker, N. Lotze, and Y. Manoli, "Power Aware Combination of Transposed-form and Direct-form FIR Ppolyphase Decimators for Sigma-Delta ADCs," in *Proc. IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'09)*, Aug. 2009, pp. 607–610.

[4] A. Shahein, M. Afifi, M. Becker, N. Lotze, and Y. Manoli, "A Power-Efficient Tunable Narrow-Band Digital Front End for Bandpass Sigma-Delta ADCs in Digital FM Receivers," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 11, pp. 883–887, Nov. 2010.

[5] R. Schreier and G. Temes, *Understanding Delta-Sigma Data Converters.* IEEE press Piscataway, NJ, 2005.

[6] F. Medeiro, A. Pérez-Verdú, and A. Rodríguez-Vázquez, *Top-down Design of High-Performance Sigma-Delta Modulators.* Kluwer Academic Pub, 1999.

[7] S. Norsworthy, R. Schreier, G. Temes *et al.*, *Delta-Sigma Data Converters: Theory, Design, and Simulation.* IEEE press New York, 1997.

[8] C. Quintans, A. Colmenar, M. Castro, M. Moure, and E. Mandado, "A Methodology to Teach Advanced A/D Converters, Combining Digital Signal Processing and Microelectronics Perspectives," *IEEE Transactions on Education*, vol. 53, no. 3, pp. 471–483, 2010.

[9] B. Murmann and B. Boser, *Digitally Assisted Pipeline ADCs: Theory and Implementation.* Springer Netherlands, 2004.

[10] B. Razavi and R. Behzad, *RF Microelectronics.* Prentice Hall Upper Saddle River, NJ, 1998.

[11] G. Temes, “Micropower Data Converters: a Tutorial,” *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 57, no. 6, pp. 405–410, 2010.

[12] B. Brannon, “Understanding State of the Art in ADCs,” *RF design magazine*, pp. 30–34, May 2008.

[13] B. Murmann. ADC Performance Survey 1997-2010. [Online]. Available: http://www.stanford.edu/~murmann/adcsurvey.htm

[14] T. Leslie and B. Singh, “An Improved Sigma-Delta Modulator Architecture,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS’90)*, 1990, pp. 372–375.

[15] A. Marques, V. Peluso, M. Steyaert, and W. Sansen, “Optimal Parameters for ΔΣ Modulator Topologies,” *IEEE Transactions on Circuits and Systems—Part II: Express Briefs*, vol. 45, no. 9, pp. 1232–1241, 2002.

[16] M. Keller, “Systematic Approach to the Synthesis of Continuous-Time Multistage Noise-Shaping Delta-Sigma Modulators,” Ph.D. dissertation, Univ. of Freiburg, Freiburg, Feb. 2010.

[17] T. Xu and M. Condon, “Comparative study of the MASH digital delta-sigma modulators,” in *Research in Microelectronics and Electronics (PRIME’09)*, Jul. 2009, pp. 196–199.

[18] J. Jarvinen and K. Halonen, “A 1.2 V Dual-Mode GSM/WCDMA ΣΔ Modulator in 65nm CMOS,” in *Proc. IEEE International Solid-State Circuits Conference (ISSCC’2006)*, 2006, pp. 1972–1981.

[19] A. Buhmann, M. Keller, M. Maurer, M. Ortmanns, and Y. Manoli, “DISCO - A Toolbox for the Discrete-time Simulation of Continuous-Time Sigma-Delta Modulators using MATLAB,” in *Proc. Midwest Symposium on Circuits and Systems (MWSCAS’07)*, Aug. 2007, pp. 1082–1085.

[20] L. Breems and J. Huijsing, *Continuous-Time Sigma-Delta Modulation for A/D Conversion in Radio Receivers.* Springer Netherlands, 2001.

[21] L. Samid, “The Design of Low Power and Low Voltage Continuous Time ΣΔ Modulators with Single Bit and Multibit Quantizer,” Ph.D. dissertation, Univ. of Freiburg, Freiburg, Jan. 2004.

[22] P. Aziz, H. Sorensen, and J. Van Der Spiegel, “An Overview of Sigma-Delta Converters,” *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 61–84, 1996.

[23] Y. Le Guillou, “Analyzing Sigma-Delta ADCs in Deep-submicron CMOS Technologies,” *RF Design Magazine*, pp. 18–26, 2005.

[24] R. Shively, “On Multistage Finite Impulse Response (FIR) Filters with Decimation,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 4, pp. 353–357, Aug. 1975.

[25] J. S. Lim and A. V. Oppenheim, *Advanced Topics in Signal Processing.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[26] P. Vaidyanathan, *Multirate Systems and Filter Banks.* Englewood Cliffs, NJ, USA: Prentice-Hall, 1993.

[27] M. José, B. Pérez-Verdú, and A. Rodríguez-Vázquez, *Systematic Design of CMOS Switched-current Bandpass Sigma-Delta Modulators for Digital Communication Chips.* Springer Netherlands, 2002.

[28] R. Crochiere and L. Rabiner, “Optimum FIR Digital Filter Implementations for Decimation, Interpolation, and Narrow-band Filtering,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 5, pp. 444–456, 1975.

[29] Crochiere, R.E. and Rabiner, L.R., *Multirate Digital Signal Processing.* Prentice-Hall Englewood Cliffs, NJ, 1983.

[30] M. Willmott, “Advanced Digital Physical Implementation Flow,” Workshop, iDESA, Saloniki, Greece, Feb. 2008.

[31] “System-level Design and Chip Architecture for Low-Power ICs,” Techtorial & Workshop, Cadence, Feldkirchen, Germany, Oct. 2009.

[32] S. Henzler, *Power Management of Digital Circuits in Deep sub-micron CMOS Technologies.* Springer Verlag, 2007.

[33] *PrimePower Manual*, Synopsys, Inc., 2006, version Y-2006.06.

[34] *Low Power in Encounter RTL Compiler*, Cadence, Inc., 2006, version 6.1.2.

[35] E. Macii, L. Bolzani, A. Calimera, A. Macii, and M. Poncino, “Integrating Clock Gating and Power Gating for Combined Dynamic and Leakage Power Optimization in Digital CMOS Circuits,” in *Proc. Conference on Digital System Design Architectures, Methods and Tools (DSD’08)*, Sept. 2008, pp. 298–303.

[36] T. Kuroda, “Optimization and Control of VDD and VTH for Low-power, High-speed CMOS Design,” in *Proc. IEEE International Conference on Computer Aided Design (ICCAD’02)*, Nov. 2002, pp. 28–34.

[37] A. Chandrakasan and R. Brodersen, *Low-power CMOS Design.* IEEE press, 1998.

[38] S. Jayapal, “Robust Energy Efficient Design for Ultra-Low Voltage CMOS VLSI,” Ph.D. dissertation, Univ. of Freiburg, Freiburg, Jul. 2009.

[39] "Expanding the Synopsys PrimeTime Solution with Power Analysis," White Paper, Synopsys, Inc., 2006.

[40] O. Gustafsson, H. Johansson, and L. Wanhammar, "An MILP Approach for the Design of Linear-phase FIR Filters with Minimum Number of Signed-power-of-two Terms," in *Proc. European Conf. Circuit Theory Design, Espoo, Finland*, 28-31 2001.

[41] H. Baher, *Analog and Digital Signal Processing.* Chichester, England: John Wiley and Sons Ltd., 2001.

[42] R. Mehboob, S. Khan, and R. Qamar, "FIR Filter Design Methodology for Hardware Optimized Implementation," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1669 –1673, august 2009.

[43] B. Widrow and I. Kollár, *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications* . Cambridge, UK: Cambridge University Press, 2008. [Online]. Available: http://www.mit.bme.hu/books/quantization/

[44] T. Ciloglu, "Normalized Peak Ripple Magnitude as an Objective Function in Discrete Coefficient FIR Filter Design," in *Proc. IEEE Midwest Symposium on Circuits and Systems (MWSCAS'01)*, vol. 1, 2001, pp. 122 –125.

[45] R. Yates. (2007) Practical Considerations in Fixed-point FIR Filter Implementations. Internet draft. [Online]. Available: http://www.digitalsignallabs.com/fir.pdf

[46] Z. G. Feng and K. L. Teo, "A Discrete Filled Function Method for the Design of FIR Filters With Signed-Powers-of-Two Coefficients," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 134–139, Jan. 2008.

[47] Z. Ye and C.-H. Chang, "Local Search Method for FIR Filter Coefficients Synthesis," in *Proc. IEEE International Workshop on Electronic Design, Test and Applications (DELTA'04)*, 28-30 2004, pp. 255 – 260.

[48] N. Takahashi and K. Suyama, "Design of CSD coefficient FIR Filters based on Branch and Bound Method," in *Proc. International Symposium on Communications and Information Technologies (ISCIT'10)*, Oct. 2010, pp. 575–578.

[49] L. Aksoy, E. da Costa, P. Flores, and J. Monteiro, "Exact and Approximate Algorithms for the Optimization of Area and Delay in Multiple Constant Multiplications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 6, pp. 1013–1026, Jun 2008.

[50] L. Aksoy, E. O. Gunes, E. Costa, P. Flores, and J. Monteiro, "Effect of Number Representation on the Achievable Minimum Number of Operations in Multiple Constant Multiplications," in *IEEE Workshop on Signal Processing Systems*, Oct. 2007, pp. 424–429.

[51] M. Imran, K. Khursheed, M. O'Nils, and O. Gustafsson, "On the Number Representation in Sub-expression Sharing," in *International Conference on Signals and Electronic Systems (ICSES'10)*, Sept. 2010, pp. 17–20.

[52] V. Rosa, E. Costa, and S. Bampi, "A VHDL Generation Tool for Optimized Parallel FIR Filters," in *Proc. International Conference on Very Large Scale Integration (IFIP'06)*, Oct. 2006, pp. 134–139.

[53] R. Mahesh and A. Vinod, "A New Common Subexpression Elimination Algorithm for Realizing Low-Complexity Higher Order Digital Filters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 217–229, Feb. 2008.

[54] C.-H. Chang and M. Faust, "On "A New Common Subexpression Elimination Algorithm for Realizing Low-Complexity Higher Order Digital Filters"," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 844–848, May 2010.

[55] M. Faust and C.-H. Chang, "Minimal Logic Depth Adder Tree Optimization for Multiple Constant Multiplication," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'10)*, June 2010, pp. 457–460.

[56] C. Koc and S. Johnson, "Multiplication of Signed-digit Numbers," *Electronics Letters*, vol. 30, no. 11, pp. 840–841, May 1994.

[57] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation.* John Wiley and Sons Ltd., 1999.

[58] M. Faust, O. Gustafsson, and C.-H. Chang, "Fast and VLSI Efficient Binary-to-CSD Encoder using Bypass Signal," *Electronics Letters*, vol. 47, no. 1, pp. 18–20, Jun. 2011.

[59] I.-C. Park and H.-J. Kang, "Digital Filter Synthesis based on Minimal Signed Digit Representation," in *Proc. Design Automation Conference*, 2001, pp. 468–473.

[60] J. Yli-Kaakinen and T. Saramaki, "A Systematic Algorithm for the Design of Multiplierless FIR Filters," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'01)*, vol. 2, May 2001, pp. 185–188.

[61] C.-Y. Yao and C.-L. Sha, "Fixed-point FIR Filter Design and Implementation in the Expanding Subexpression Space," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'10)*, Jun. 2010, pp. 185–188.

[62] T. Fujie, R. Ito, K. Suyama, and R. Hirabayashi, "A New Heuristic Signed-power of two term Allocation Approach for Designing of FIR Filters," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'03)*, vol. 4, May 2003, pp. IV–285–IV–288.

[63] Y.-C. Lim, R. Yang, D. Li, and J. Song, "Signed Power-of-two (SPT) term Allocation Scheme for the Design of Digital Filters," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'98)*, vol. 5, 1998, pp. 359–362.

[64] S. Takriti, "AMPL: A Modeling Language for Mathematical Programming," pp. 144–146, 1994.

[65] M. Aktan, A. Yurdakul, and G. Dundar, "An Algorithm for the Design of Low-Power Hardware-Efficient FIR Filters," *IEEE Transactions on Circuits and Systems—Part I: Regular Papers*, vol. 55, no. 6, pp. 1536–1545, Jul. 2008.

[66] Y. Lim, "Design of Discrete-coefficient-value Linear Phase FIR Filters with Optimum Normalized Peak Ripple Magnitude," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 12, pp. 1480–1486, Dec. 1990.

[67] H. Q. Ta and T. Le-Nhat, "Design of FIR Filter with Discrete Coefficients based on Mixed Integer Linear Programming," in *Proc. International Conference on Signal Processing (9th ICSP'08)*, 26-29 2008, pp. 9 –12.

[68] W.-S. Lu and T. Hinamoto, "Design of FIR Filters with Discrete Coefficients via Polynomial Programming: Towards the Global Solution," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'07)*, 27-30 2007, pp. 2048 –2051.

[69] W.-S. Lu, "Design of FIR Filters with Discrete Coefficients: a Semidefinite Programming Relaxation Approach," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'01)*, vol. 2, 6-9 2001, pp. 297–300.

[70] Y. J. Yu and Y. C. Lim, "Design of Linear Phase FIR Filters in Subexpression Space Using Mixed Integer Linear Programming," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 10, pp. 2330–2338, Oct. 2007.

[71] F. Xu, C. H. Chang, and C. C. Jong, "Design of Low-Complexity FIR Filters Based on Signed-Powers-of-Two Coefficients With Reusable Common Subexpressions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 10, pp. 1898–1907, Oct. 2007.

[72] C.-L. Chen and J. Willson, A.N., "A Trellis Search Algorithm for the Design of FIR Filters with Signed-powers-of-two Coefficients," *IEEE Transactions on Circuits and Systems—Part II: Analog and Digital Signal Processing*, vol. 46, no. 1, pp. 29–39, Jan. 1999.

[73] Y. Lim and S. Parker, "Discrete Coefficient FIR Digital Filter Design based upon an LMS Criteria," *IEEE Transactions on Circuits and Systems*, vol. 30, no. 10, pp. 723–739, Oct 1983.

[74] H. Samueli, “An Improved Search Algorithm for the Design of Multiplierless FIR Filters with Powers-of-two Coefficients,” *Circuits and Systems, IEEE Transactions on*, vol. 36, no. 7, pp. 1044–1047, Jul 1989.

[75] W. S. Lu, “Design of FIR Digital Filters with Discrete Coefficients via Convex Relaxation,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS’05)*, vol. 2, 23-26 2005, pp. 1831 – 1834.

[76] J. Löfberg, “ YALMIP : A Toolbox for Modeling and Optimization in MATLAB ,” in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip

[77] Y. Labit, D. Peaucelle, and D. Henrion, “SeDuMi interface 1.02: a tool for solving LMI problems with SeDuMi,” in *Proc. IEEE International Symposium on Computer Aided Control System Design*, 2002, pp. 272–277.

[78] D. Henrion and J. Lasserre, “GloptiPoly: Global optimization over polynomials with Matlab and SeDuMi,” in *Proc. IEEE Conference on Decision and Control*, vol. 1, 2001, pp. 747–752.

[79] FIRsuite, “Suite of constant coefficient FIR filters,” 2010. [Online]. Available: http://www.firsuite.net

[80] D. Li, J. Song, and Y. C. Lim, “A Polynomial-time Algorithm For Designing Digital Filters With Power-of-two Coefficients,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS’93)*, 1993, pp. 84–87.

[81] D. Shi and Y. J. Yu, “Design of Linear Phase FIR Filters With High Probability of Achieving Minimum Number of Adders,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 1, pp. 126–136, Jan. 2011.

[82] A. Vinod and E.-K. Lai, “An Efficient Coefficient-partitioning Algorithm for Realizing Low-complexity Digital Filters,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 12, pp. 1936–1946, Dec. 2005.

[83] R. Hartley, “Subexpression Sharing in Filters using Canonic Signed Digit Multipliers,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[84] K. Kato, Y. Takahashi, and T. Sekine, “A New Horizontal and Vertical Common Subexpression Elimination Method for Multiple Constant MMultiplication,” in *Proc. IEEE International Conference on Electronics, Circuits, and Systems (ICECS’09)*, Dec. 2009, pp. 124–127.

[85] A. Yurdakul and G. Dundar, “Fast and Efficient Algorithm for the Multiplierless Realisation of Linear DSP Transforms,” *IEE Proceedings - Circuits, Devices and Systems*, vol. 149, no. 4, pp. 205–211, Aug. 2002.

[86] M. Martinez-Peiro, E. Boemo, and L. Wanhammar, “Design of High-speed Multiplierless Filters using a Nonrecursive Signed Common Subexpression Algorithm,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 3, pp. 196 –203, Mar. 2002.

[87] Y. Voronenko and M. Püschel, “Multiplierless Multiple Constant Multiplication,” *ACM Transactions on Algorithms*, vol. 3, no. 2, 2007.

[88] Spiral, “Software/Hardware generation for DSP Algorithms.” [Online]. Available: http://spiral.ece.cmu.edu/mcm/gen.html

[89] O. Gustafsson and A. Dempster, “On the use of Multiple Constant Multiplication in Polyphase FIR Filters and Filter Banks,” in *Proc. Nordic Signal Processing Symposium (NORSIG’04)*, 2004, pp. 53 – 56.

[90] H. Aboushady, Y. Dumonteix, M.-M. Louerat, and H. Mehrez, “Efficient Polyphase Decomposition of Comb Decimation Filters in Sigma Delta Analog-to-Digital Converters,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,*, vol. 48, no. 10, pp. 898 –903, Oct. 2001.

[91] M. Becker, N. Lotze, M. Ortmanns, and Y. Manoli, “Implementation and Analysis of Power Consumption for a Power Optimized Decimator Designed for Cascaded Sigma-Delta A/D Converters,” in *Proc. IEEE International Midwest Symposium onCircuits and Systems (MWSCAS’06)*, vol. 2, Aug. 2006, pp. 654–658.

[92] V. Rosa, E. Costa, J. Monteiro, and S. Bampi, “Performance Evaluation of Parallel FIR Filter Optimizations in ASICs and FPGA,” in *Proc. IEEE Midwest Symposium on Circuits and Systems (MWSCAS’05)*, Aug. 2005, pp. 1481–1484.

[93] O. Gustafsson, J. Coleman, A. Dempster, and M. Macleod, “Low-complexity Hybrid form FIR Filters using Matrix Multiple Constant Multiplication,” in *Proc. Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, vol. 1, Nov. 2004, pp. 77–80.

[94] T. Shahana, B. Jose, K. Jacob, and S. Sasi, “Decimation Filter Design Toolbox for Multi-Standard Wireless Transceivers using MATLAB,” *International Journal of Signal Processing*, vol. 5, p. 2, 2009.

[95] L. Fujcik, A. Kuncheva, T. Mougel, and R. Vrba, “New VHDL design of decimation filter for sigma-delta modulator,” in *Proc. IEEE Asian Conference on Sensors and the International Conference on new Techniques in Pharmaceutical and Biomedical Research*, 2005, pp. 204–207.

[96] V. Rosa, E. Costa, and S. Bampi, “A VHDL Generation Tool for Optimized Parallel FIR Filters,” in *Proc. IEEE International Conference on Very Large Scale Integration (IFIP’06)*, 2006, pp. 134–139.

[97] F. Daitx, V. Rosa, E. Costa, P. Flores, and S. Bampi, "VHDL Generation of Optimized FIR Filters," in *Proc. IEEE International Conference on Signals, Circuits and Systems (SCS'08)*, 2008, pp. 1–5.

[98] V. Verma and C. Chien, "A VHDL based Functional Compiler for Optimum Architecture Generation of FIR Filters," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'96)*, vol. 4, May 1996, pp. 564–567.

[99] K.-Y. Jheng, S.-J. Jou, and A.-Y. Wu, "A Design Flow for Multiplierless Linear-phase FIR Filters: from System Specification to Verilog Code," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'04)*, vol. 5, May 2004, pp. V–293–V–296.

[100] N. Lotze, "Ein Generisches Modell fuer die Effizienzermittlg und Implementierung von Digitalen Polyphasenfiltern in VHDL," Master's thesis, Univ. of Freiburg, Freiburg, 2004.

[101] M. Ortmanns, "Error Compensation in Continuous-Time Sigma-Delta A/D Converters," Ph.D. dissertation, Univ. of Freiburg, Freiburg, Dec. 2003.

[102] M. Coffey, "Optimizing Multistage Decimation and Interpolation Processing," *IEEE Signal Processing Letters*, vol. 10, no. 4, pp. 107–110, Apr. 2003.

[103] M. Coffey, "Optimizing Multistage Decimation and Interpolation Processing: Part II," *IEEE Signal Processing Letters*, vol. 14, no. 1, pp. 24–26, Jan. 2007.

[104] D. Schlichthärle, *Digital filters: basics and design.* Springer Verlag, 2000.

[105] E. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 2, pp. 155–162, 1981.

[106] G. Dolecek and F. Harris, "Design of CIC Compensator Filter in a Digital IF Receiver," in *Proc. International Symposium on Communications and Information Technologies (ISCIT'08)*, 2008, pp. 638–643.

[107] R. Zimmermann, "Binary Adder Architectures for Cell-Based VLSI and their Synthesis," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, 1998.

[108] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays.* Springer Verlag, 2007.

[109] P. Vaidyanathan, "Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: A tutorial," *Proceedings of the IEEE*, vol. 78, no. 1, pp. 56–93, 1990.

[110] T. Shahana, R. James, B. Jose, K. Poulose Jacob, and S. Sasi, "Polyphase Implementation of Non-recursive Comb Decimators for Sigma-Delta A/D Converters," in *Proc. IEEE Conference on Electron Devices and Solid-State Circuits (EDSSC'07)*, Dec. 2007, pp. 825–828.

[111] O. Gustafsson, K. Johansson, H. Johansson, and L. Wanhammar, "Implementation of Polyphase Decomposed FIR Filters for Interpolation and Decimation Using Multiple Constant Multiplication Techniques," in *Proc. IEEE Asia Pacific Conference on Circuits and Systems (APCCAS'06)*, Dec. 2006, pp. 924–927.

[112] H. Zhu, X. Wu, and X. Yan, "Low-Power and Hardware Efficient Decimation Filters in Sigma-Delta A/D Converters," in *Proc. IEEE Conference on Electron Devices and Solid-State Circuits*, Dec. 2005, pp. 665–668.

[113] K. Khoo, Z. Yu, and A. Willson Jr, "Design of Optimal Hybrid form FIR Filter," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'01)*, vol. 2, 2001, pp. 621–624.

[114] R. Teymourzadeh and B. Othman, "An Enhancement of Decimation Process using Fast Cascaded Integrator Comb (CIC) Filter," in *Proc. IEEE International Conference on Semiconductor Electronics (ICSE'06)*, 2006, pp. 811–815.

[115] E. Cheney and D. Kincaid, *Numerical Mathematics and Computing.* Brooks/Cole Pub Co, 2007.

[116] J. Epperson, *An Introduction to Numerical Methods and Analysis.* Wiley-Blackwell, 2007.

[117] M. Afifi, M. Ortmanns, and Y. Manoli, "Design Study of a Tunable Bandpass Continous Time Sigma Delta Modulator for FM Digital Reciver," in *Proc. International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES'08)*, June 2008, pp. 219–224.

[118] A. Abidi, "The Path to the Software-Defined Radio Receiver," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 5, pp. 954–966, May 2007.

[119] R. Schreier and W. Snelgrove, "Decimation for Bandpass Sigma-Delta Analog-to-Digital Conversion," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'90)*, vol. 3, May 1990, pp. 1801–1804.

[120] B. White and M. Elmasry, "Low-power Design of Decimation Filters for a Digital IF Receiver," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 339–345, Jun. 2000.

[121] H.-J. Pfleiderer and S. Lachowicz, "Numerically Controlled Oscillators using Linear Approximation," in *Proc. International Conference on Field Programmable Logic and Applications (FPL'09)*, Sep. 2009, pp. 695–698.

[122] J. Vankka, "Methods of Mapping from Phase to Sine Amplitude in Direct Digital Synthesis," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 44, no. 2, pp. 526–534, Mar. 1997.

[123] Analog Devices, “Fundamentals of Direct Digital Synthesis (DDS),” Tutorial: MT-085, 2009. [Online]. Available: http://www.analog.com/static/imported-files/tutorials/MT-085.pdf

[124] Analog Devices, “Understand SINAD, ENOB, SNR, THD, THDN, and SFDR so you Don’t Get Lost in the Noise Floor,” Tutorial: MT-003, 2009. [Online]. Available: http://www.analog.com/static/imported-files/tutorials/MT-003.pdf

[125] J. Langlois and D. Al-Khalili, “Phase to Sinusoid Amplitude Conversion Techniques for Direct Digital Frequency Synthesis,” *Proc. IEE - Circuits, Devices and Systems*, vol. 151, no. 6, pp. 519–528, Dec. 2004.

[126] H. Samueli, T.-j. Lin, R. Hawley, and S. Olafson, “VLSI Architectures for a High-Speed Tunable Digital Modulator/Demodulator/Bandpass-filter Chip Set,” in *Proc. IEEE International Symposium on Circuits and Systems (IS-CAS’92)*, vol. 3, May 1992, pp. 1065–1068.

[127] M. Kumm, “FPGA Realization of a Offset Local Oscillator based on PLL and DDS Technologies,” Diploma thesis, Technical University of Darmstadt, Darmstadt, Jul. 2007. [Online]. Available: http://www.martin-kumm.de/Diplomarbeit_Martin_Kumm_TUD_2007.pdf

[128] L. K. Tan and H. Samueli, “A 200 MHz Quadrature Digital Synthesizer/Mixer in 0.8$\mu$m CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 193–200, Mar. 1995.

[129] S. Khilar, K. Parmar, S. Saumi, and K. Dasgupta, “Design and Analysis of Direct Digital Frequency Synthesizer,” in *Proc. of the First International Conference on Emerging Trends in Engineering and Technology (ICETET’08).* IEEE, 2008, pp. 1302–1306.

[130] G. Zimmerman and M. Flanagan, “Spur-reduced Numerically-Controlled Oscillator for Digital Receivers,” in *Proc. Conference Record of The Twenty-Sixth Asilomar Conference on Signals, Systems and Computers.* IEEE, 1992, pp. 517–520.

[131] Y. Yang, J. Cai, and L. Liu, “A Novel DDS Array Structure with Low Phase Noise and Spurs,” *Channels*, vol. 1, p. 2, 2011.

[132] D.-U. Lee, R. Cheung, W. Luk, and J. Villasenor, “Hardware Implementation Trade-Offs of Polynomial Approximations and Interpolations,” *IEEE Transactions on Computers*, vol. 57, no. 5, pp. 686–701, May 2008.

[133] E. Lopelli, J. van der Tang, and A. van Roermund, “Minimum Power-Consumption Estimation in ROM-Based DDFS for Frequency-Hopping Ultra low-Power Transmitters,” *IEEE Transactions on Circuits and Systems—Part I: Regular Papers*, vol. 56, no. 1, pp. 256–267, Jan. 2009.

[134] A. Mohieldin, A. Emira, and E. Sanchez-Sinencio, “A 100-MHz 8-mW ROM-less Quadrature Direct Digital Frequency Synthesizer,” *IEEE Journal of Solid-State Circuits*, vol. 37, no. 10, pp. 1235–1243, Oct. 2002.

[135] J. Tierney, C. Rader, and B. Gold, “A Digital Frequency Synthesizer,” *IEEE Transactions on Audio and Electroacoustics*, vol. 19, no. 1, pp. 48–57, Mar. 1971.

[136] R. Lyons, *Understanding Digital Signal Processing.* Prentice Hall, 2011.

# List of Figures

# List of Tables

# A. Polynomial Programming

The regular FIR filter response is described by

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k}, \tag{A.1}$$

where $h_k$ is the filter coefficients and $N$ is the filter length. Assume odd filter order, then

$$n = \frac{N+1}{2}. \tag{A.2}$$

Equation (A.1) can be written as follow using the previous assumption:

$$A(w) = \sum_{k=0}^{n} a_k \cos kw. \tag{A.3}$$

The error function, or the quadratic problem to be minimzed is:

$$e^2(w) = [A(w) - A_d(w)]^2 \tag{A.4}$$

Let's defiene the following paramters:

$$\begin{aligned} a_{mk} =& \frac{\overline{a}_k + \underline{a}_k}{2}, \\ \delta_k =& \frac{\overline{a}_k - \underline{a}_k}{2}, \\ x_k \in& \{-1, 1\}, \\ a_k =& a_{mk} + x_k \delta_k, \end{aligned}$$

where $\overline{a}_k$ and $\underline{a}_k$ is the smallest SPT upper bound and largest SPT lower bound of a$_k$,respectively, and $A_d(w)$ is the desired frequency response. Hence (A.3) can be written as:

$$\begin{aligned} A(w) =& \sum_{k=0}^{n} a_k \cos kw, \\ =& \sum_{k=0}^{n} (a_{mk} + x_k \delta_k) \cos kw, \\ =& \sum_{k=0}^{n} a_{mk} \cos kw + x_k \delta_k \cos kw. \end{aligned}$$

Consequently, (A.3) can be rewritten as follow:

$$A(w) = A_m(w) + x^T C(w), \tag{A.7}$$

where:

$$\begin{aligned} A_m(w) =& \sum_{k=0}^{n} a_{mk} \cos kw, \\ C(w) =& x_k d_k \cos kw, \\ x_k =& [x_0 x_1 \cdots x_n]^T. \end{aligned}$$

Substitute in (A.4)

$$\begin{aligned} e^2 =& [A(w) - A_d(w)]^2 && \text{(A.9a)} \\ =& [A_m(w) + x^T C(w) - A_d(w)]^2 && \text{(A.9b)} \\ =& (A_m(w) - A_d(w))^2 + 2(A(w) - A_d(w)) x^T C(w) + x^T C(w) x C^T(w) && \text{(A.9c)} \\ =& 2(A_m(w) - A_d(w)) x^T C(w) + x^T C(w) x C^T(w) && \text{(A.9d)} \end{aligned}$$

The first term in (A.9c) can be neglected because it is constant with respect to $x$ which leads to

$$P(x) = x^T Q x + q^T x, \tag{A.10}$$

where:

$$\begin{aligned} Q =& \int_{-\pi}^{\pi} W(w) C(w) C^T(w) dw, \\ q =& 2 \int_{-\pi}^{\pi} W(w)[A(w) - A_d(w)] C(w) dw, \end{aligned}$$

and $W(w)$ is a weighting function.

# B. Downsampling

A sampled signal sequence is shown in Fig. B.1. To downsample the signal by a downsample factor $M = 3$, sample $x(0)$ is preserved and the following two samples are discarded, then sample $x(3)$ is preserved and the following two samples are discared, and so on, as shown in Fig. B.1 [136]. It should be noted that, the decrease in the sampling rate is preserved by increasing the output word-length.

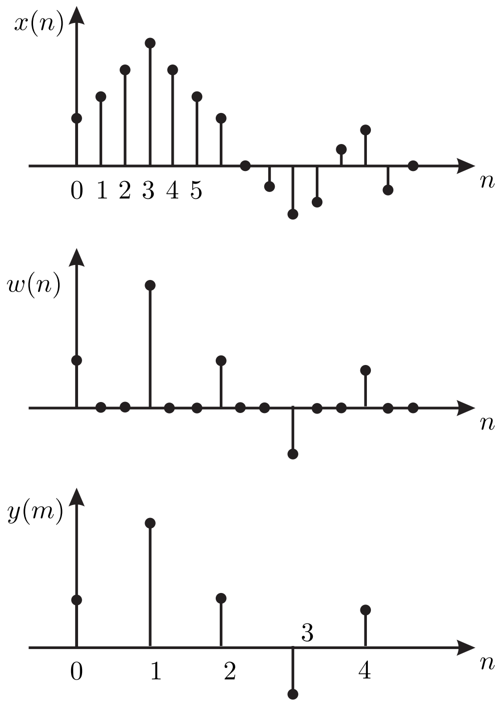


**Figure B.1.:** Sample rate conversion for downsampled by 3.

The spectral of a sampled signal is centered around zero shown by the solid line in Fig. B.2. The spectral replications is located at integer multiples of $2\pi$ rad/s, as shown by dashed lines in Fig. B.2. An $M-1$ copies of the primary spectral (which centered around zero) are inserted between the spectral replications at equally spaced spectral of $2\pi/M$, as shown by gray dashed lines in Fig. B.2. The frequency axis is scaled by factor of $M$, yielding to the new axis $\omega_{new}$, as shown in Fig. B.2. Afterwards, the magnitude axis is scaled by factor of $1/M$, as shown by $P/3$ in Fig. B.2 [136].

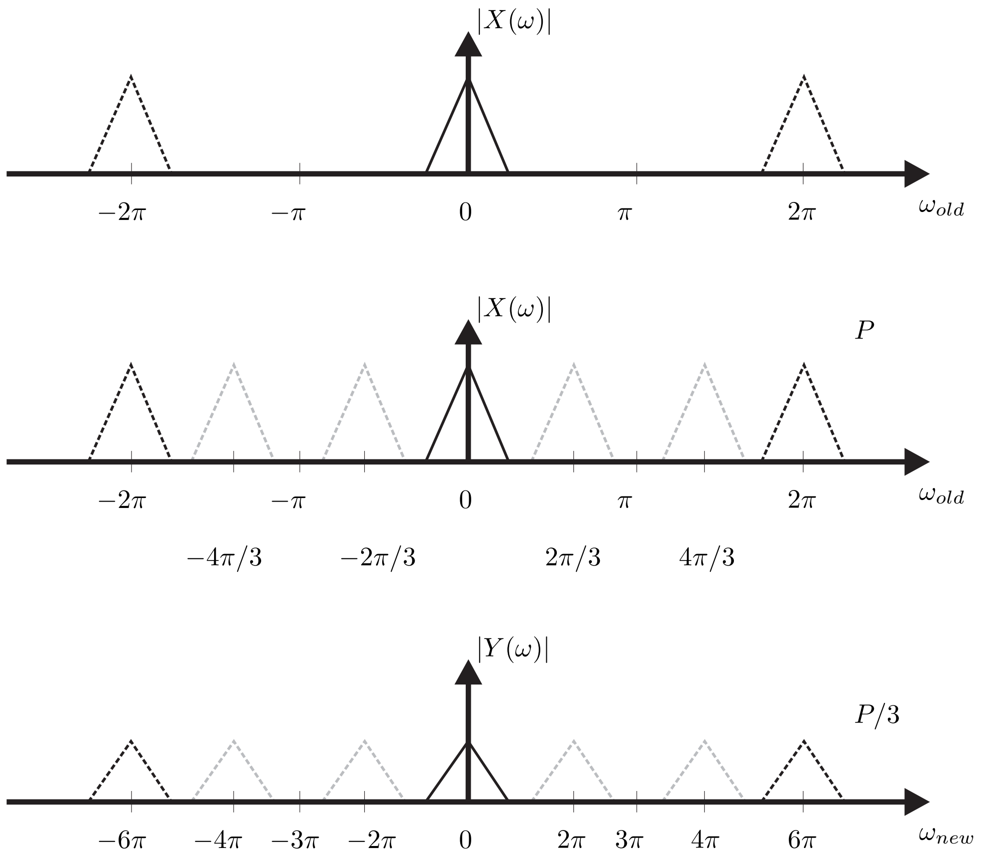
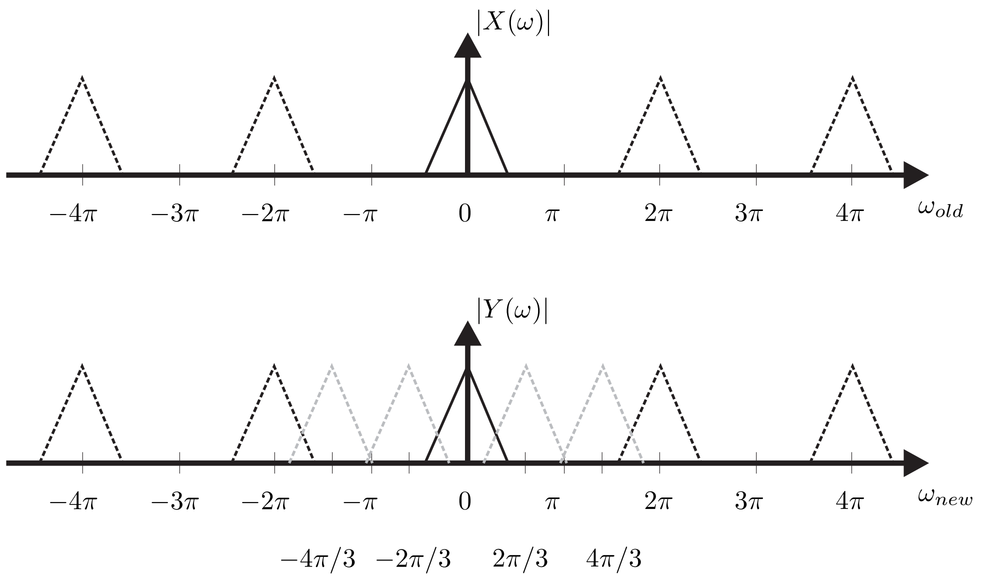


**Figure B.2.:** Spectra of downsampling by 3.

The spectral of a sampled signal is centered around zero shown by the solid line in Fig. B.3. The spectral replications is located at integer multiples of $2\pi$ rad/s, as shown by dashed lines in Fig. B.3. Insert $L$ images of the primary spectral, where $L$ is the interpolation factor. The inserted images will be attenuated by the subsequent lowpass filter [136].



$|X(\omega)|$
$-4\pi$ $-3\pi$ $-2\pi$ $-\pi$ $0$ $\pi$ $2\pi$ $3\pi$ $4\pi$
$\omega_{old}$
$|Y(\omega)|$
$-4\pi$ $-3\pi$ $-2\pi$ $-\pi$ $0$ $\pi$ $2\pi$ $3\pi$ $4\pi$
$\omega_{new}$
$-4\pi/3$ $-2\pi/3$ $2\pi/3$ $4\pi/3$

**Figure B.3.:** Spectra of interpolating by 3.

# Index

# Acknowledgements

I owe sincere thankfulness and gratitude to my mentor and supervisor “mein Doktorvater” Prof. Dr.-Ing. Yiannos Manoli for his valuable guidance and extensive support. I am so grateful for the opportunity being part of his team in Microelectronics. Working at his group under his supervision has been invaluable experience for me. I learned a lot from him and I am keen to learn more from him.

I am sincerely grateful to Didier Henrion, Michael Grant, Stephen Boyd, Johan Löfberg, and Oscar Gustafsson for there valuable assistance to apprehend their toolboxes.

I am extremely grateful to Niklas Lotze for his extensive and comprehensive valuable discussions and constructive feedback.

I am indebted to Micheal Maurer and Mohamed Afifi for their extensive valuable discussions which have been very helpful during my research.

Many thanks to my students Markus Kocum, Qiang Zhang and Tobias Meinert.

Special thanks to all members of the Microelectronics group in particular Elisabeth-Theresia Maurer, Claudia Jahn, Anna Zimmermann, Zahra Dandani, Mirjam Hedinger and Rolf Schlecker. Warm thanks to my colleagues Markus Becker, Maurits Ortmanns, Matthias Keller, Stains Trendelenburg, Thorsten Hehn, Senthil Jayapal, Khald Aljasem, Sherif Mohamed, Armin Taschwer and Hussam Kloub.

I would like to express my sincere gratitude to the International Graduate Acadamey (IGA) for affording various valuable workshops and courses in soft skills during my studies.

I am extremely grateful to Christian Panis and Evert-Jan Paul my colleges at Catena Radio Design B.V. for their valuable remarks and discussions.

Beyond engineering, I am indebted to my beloved parents, brothers and sisters, and my wife for supporting me throughout all my studies.