Energy-Efficient Fixed-Coefficient FIR Filters for Millimeter-Wave Radios

Erik Lundell

Gustav Molin

Department of Automatic Control

MSc Thesis
TFRT-6199
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

With the introduction of millimeter-wave antenna arrays in 5G base-stations and ever-increasing data volumes, the power consumption of the signal processing in digital radio systems has increased over the last decade. This, combined with increased cost and environmental awareness, has put focus on power optimization. This thesis investigates different aspects of optimizing one important component of digital radio systems, fixed-coefficient FIR filters.

The thesis initially investigates subblocks of the FIR filters separately. The cost of fixed coefficient multiplication is found to be linearly dependent on the number of signed digits in the coefficient set while additions are linearly dependent on the number of summed bits.

The second part of the thesis considers complete filters. First, two common filter architectures are compared and the direct form is found to be more efficient. Then, a mixed-integer linear program is formulated that minimizes the number of signed digits in the filter coefficients. This optimization is shown to be able to reduce the number of signed digits with 20–30% compared to the median which translates to a similar reduction in area and power. Furthermore, allowing the filter gain to be flexible is found to reduce the optimal number signed digits additionally with around 20%. For filter specifications where there are multiple solutions with the same optimal number of signed digits, a ranking score based on other coefficient features is suggested.

In the final part of the thesis, the optimal solutions from the second part are optimized further. An algorithm for finding common sub-expressions in coefficient sets is developed. Using these sub-expressions when implementing the filter is shown to reduce both area and power consumption with 5–25%.

Acknowledgements

We would like to thank Olof Troeng, our supervisor at Ericsson who has supported us extensively throughout the entire thesis, providing great feedback and insight. In addition to Olof, there are many colleagues at Ericsson that have aided our thesis with knowledge of the ASIC synthesis toolchain of which we would like to thank Joakim Ahlgren, Hans Hagberg, Grzegorz Swiecanski, and Inayat Ullah in particular. Finally, we would like to thank our academic supervisor Bo Bernhardsson for providing valuable feedback on the report.

Contents

1

Introduction

Although radio signals are inherently analog, digital components have played an important role in radio systems since at least the 1980s when the digital radio standard Global System for Mobile communications (GSM) was introduced. Rapidly improving semiconductor technology made this entirely new way of constructing radio systems possible. Since then, the field of Digital Signal Processing (DSP) has grown massively in every aspect from algorithms to hardware implementation. These digital systems can be built using flexible embedded microcontrollers. However, the highly specialized nature of DSP makes application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs) viable choices for achieving high performance at a low production cost. The high level of control of these platforms has the added benefit of making detailed optimization possible.

Digital filters make up an integral part of any digital-analogue radio system. One filter type with several useful properties is the finite impulse response (FIR) filter. Such filters and many other DSP components contain many fixed-coefficient multiplications and summations. As such, much research has gone into reducing the power consumption, silicon area and delay of such operations. Since the fifth-generation technology standard, 5G, has been rolled out in cellular networks, the topic has been revitalized. Modern 5G base stations utilize radio waves with wavelengths in the millimeter range. Millimeter wave radios need many antennas operating in parallel, so-called antenna arrays, to get sufficient gain and to transmit multiple radio-beams simultaneously. Each antenna needs one DSP chain which means that the number of filters and accordingly their power consumption scales with the number of antennas. In addition to this, data volumes have increased steadily which further increases the power consumption.

In addition to data throughput, the focus of optimization efforts has thus shifted towards minimizing power consumption. Decreasing power consumption of course decreases the cost of running the base station but also reduces the size and weight of the equipment since less cooling is needed. Improving power consumption can therefore reduce manufacturing and transportation costs as well as environmental impact.

1.1 Goal and purpose

The goal of this thesis is to investigate how the implementation of fixed-coefficient FIR filters in ASICs can be optimized to reduce power consumption. The thesis determines areas with power-optimization potential and applies methods for utilizing this potential. Silicon area is also considered due to the link between area and power. Secondly, the thesis strives to quantify the gain of such optimizations.

Since FIR filters are common components in DSP systems such optimizations can have an impact on many applications. Reducing power consumption in ASICs reduces both cost and environmental impact of the ASIC at runtime but also in production, since excess power generates heat which requires installing heat sinks for cooling. In extreme cases, reducing power consumption of FIR filters might even be crucial to make a design feasible since the temperature of a circuit must be kept below a certain threshold.

1.2 Problem formulation

The optimization effort can broadly be split into two main stages. First, the filter is designed. The filter architecture is decided and the coefficients are chosen. At this level, this thesis investigates

- Out of the direct form and transposed direct form, which FIR filter architecture performs the best power and area-wise?
- How should the coefficients of a fixed-coefficient FIR filter be chosen to optimize power and area, given a certain filter specification?
- How does the clock frequency of an ASIC affect the power and area consumption?

Then, the designed filter is implemented in RTL code where additional optimization can be done. At this level, the thesis investigates

- Can manually implemented sub-expression sharing outperform the power and area optimization performed by the ASIC synthesis toolchain and if so, with how much?
- How is the manual RTL optimization affected by clock frequency?

1.3 Methodology

In the initial phase of this thesis, information about how to approach the problem and which parameters to consider when optimizing ASICs in general and FIR filters in particular was gathered, both from academic literature and from Ericsson internally. Then, an environment for synthesizing parameterized RTL code and estimating its power consumption was created for prototyping manual implementations and automated testing of many configurations.

Matlab was used to create batches of configurations with varying coefficients and clock frequencies. The coefficients were chosen depending on what the experiment aims to test. However, one main focus was the minimization of signed digits through a mixed-integer linear program. These configurations were then synthesized and their power consumption was estimated. Finally, the results were fed back to Matlab for analysis.

In addition to these experiments, an algorithm was developed that determines which solutions from the above linear program have good RTL-level-optimization potential. These were implemented manually to compare with the results from the automatic synthesis.

1.4 Delimitations

The filters considered in the thesis are limited to low-pass filters, though the conclusions of this investigation should easily carry over to other filter types. Regarding the architecture of the filters, only the two most common FIR realizations are considered, the direct form and the transposed direct form. Because of the work-intensive nature of hand implementing the RTL level optimizations mentioned above, these were only carried of for the most promising coefficient sets. One final limitation is that due to technical limitations, the power estimations do not include glitch power.

1.5 Outline

The following chapter lays out a theoretical background for power optimizing FIR filters. First, digital filters in general and FIR filters in particular are explained. Then, a section follows with a description of ASICs and what to consider when optimizing ASICs for power. Thirdly, these areas are combined and the implementation of FIR filters in ASICs is discussed. The chapter finishes with a description of earlier work on how to select efficient coefficients for FIR filters.

The work carried out in the thesis is then presented and discussed in three chapters. Chapter 3 describes the initial work of the thesis. The experiment setup is explained and the initial investigations of subblocks of FIR filters are presented. Building on these results, in Chapter 4 optimizations on the filter design are explored. Two architectures for implementing FIR filters are compared and a strategy for selecting power-efficient coefficients is developed. Finally, in Chapter 5, different strategies for reducing the area and power even further when implementing the filter in RTL code are explored. The main results of the thesis are summarized in the concluding Chapter 6.

1.6 Abbreviations and notation

The following list explains the abbreviations most commonly used throughout the thesis.

ASIC	Application specific integrated chip
CMVM	Constant matrix vector multiplication
CSD	Canonical signed digits
DF	Direct form
DTFT	Discrete-time Fourier transform
FIR	Finite impulse response
MCM	Multiple constant multiplication
MILP	Mixed-integer linear program
POT	Powers of two
RTL	Register transfer level
SCM	Single constant multiplication
SOP	Sum of products
SPT	Signed powers of two
TF	Transposed form

The area unit used throughout the thesis is μm^2 which is meant to be interpreted as square micrometers. Bit-shifts are denoted with the symbol “$\ll$”. $x \ll 3$ should thus be interpreted as x bit-shifted 3 bits to the left. A binary digit with a bar is used to denote that the bit is to be subtracted from the binary number rather than added. For example, $100\bar{1} = 8 - 1 = 7$. This way of representing numbers is called signed powers of two (SPT) representation or signed digits representation and is explained more thoroughly in section 2.3.3. These terms are used interchangeably throughout the thesis.

2

Background

2.1 Digital filters

2.1.1 The frequency domain of a discrete signal

To process continuous signals in digital systems, they first have to be sampled. If a real-valued signal x is sampled at a constant time interval, producing a series $x[n]$, $n \in \mathbb{Z}$, that is finitely summable, there exists a discrete-time Fourier transform (DTFT)

$$X(e^{j2\pi f}) = \mathscr{F}(x[n]) = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi fn}.$$

The DTFT is periodic with a period of one and is usually plotted symmetrically around zero, for $f = -0.5, \ldots, 0.5$. Additionally, there is a reverse formula

$$x[n] = \int X(e^{j2\pi f})e^{j2\pi fn}\,df,$$

where the integration can be done over any interval of length 1. These transforms reveal that $x[n]$ in addition to the time domain can be analyzed in the frequency domain, as a spectrum of complex oscillations with varying frequencies and amplitudes. One property of the DTFT is that since the transformation is linear, linear operations stay linear after the transform. One notable example is the time shift which has the transform

$$\begin{aligned}\mathscr{F}(x[n-m]) &= \sum_{n=-\infty}^{\infty} x[n-m]e^{-j2\pi fn} = \sum_{n'=-\infty}^{\infty} x[n']e^{-j2\pi f(n'+m)} \\ &= X(e^{j2\pi f})e^{-j2\pi fm}, \end{aligned} \tag{2.1}$$

where the variable substitution $n' = n - m$ is used. Usually, the time shift is stated in terms of the z-transform and the substitution $z = e^{j2\pi f}$ is used such that $\mathscr{F}(x[n - m]) = X(z)z^{-m}$. The time-shift operator is the basic building block for digital filters which will be described in the next section and be analyzed using DTFT.

2.1.2 Digital filters

Linear time-invariant (LTI)-filters operating on discrete signals are called digital filters. The most general class, called Infinite Impulse Response (IIR) filters, are defined by a difference equation

$$\sum_{k=0}^{K-1} a_k y[n-k] = \sum_{m=0}^{M-1} b_m x[n-m]$$

where $y[n]$ is the output signal and $x[n]$ is the input signal. M is the order of the filter and a_k and b_m are coefficients that determine the effect of the filter. The output is thus computed as a weighted sum of the input and earlier computed output values. Applying the DTFT on both sides of the equation, utilizing equation 2.1 and inserting the substitution $z = e^{j2\pi f}$ yields

$$\sum_{k=0}^{K-1} a_k z^{-k} Y(z) = \sum_{m=0}^{M-1} b_m z^{-m} X(z)$$

which can be rewritten in the common form

$$H(z) := \frac{Y(z)}{X(z)} = \frac{\sum_{m=0}^{M-1} b_m z^{-m}}{\sum_{k=0}^{K-1} a_k z^{-k}}.$$

$H(z)$, called the transfer function, is a rational function in z and completely characterizes the filter. Since it is complex-valued function, it is often separated into magnitude, $|H(z)|$ and angle, $\angle H(z)$, the gain and phase shift at the complex frequency z. Since IIR filters reuse previously calculated output values, they require feedback in their implementation. This causes more complex implementations compared to feed-forward filters but also, more importantly, means that IIR filters are not guaranteed to be stable. A finite input can have infinite gain and the filter might output infinite oscillations after the input signal has reached zero. Because of this, a subset of IIR filters where the denominator of $H(z)$ is equal to one, called Finite Impulse Response (FIR) filters, are often considered. FIR filters usually require higher orders than IIR-filters to achieve similar performance but on the other hand do not require feedback and are guaranteed to be stable. Additionally, they turn out to have other useful properties.

2.1.3 FIR filters

When discussing FIR filters, $\bar{h} = \{h_k,\ k = 0 \ldots M-1\}$ is usually used to denote the coefficients. The time domain equation for a FIR filter of order M is then

$$y[n] = \sum_{k=0}^{M-1} h_k x[n-k]$$

and the corresponding transfer function is

$$H(z) = \sum_{k=0}^{M-1} h_k z^{-k}.$$

When designing FIR filter implementations, it is often useful to view the filter in matrix form, allowing for convenient notation of different factorizations:

$$y[n] = H(z)x[n] = \begin{bmatrix} h_0 & h_1 & \dots & h_{M-1} \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-(M-1)} \end{bmatrix} x[n] =: \bar{h}^T \bar{z} x[n]. \tag{2.2}$$

A special class of FIR filters are symmetric filters where $h_k = h_{M-1-k}$. Reminding ourselves that $z = e^{j2\pi f}$, $H(z)$ can be rewritten by factorizing out $e^{-j2\pi\frac{M-1}{2}}$ which leaves pairs of complex exponentials $e^{\pm nj2\pi f}$ for some n:s. Because of the symmetric coefficients these exponentials have the same coefficient and can be combined using Eulers formula to the real-valued cosine function. The remaining complex part of the transfer function is then only the factorized $e^{-j2\pi\frac{M-1}{2}f}$ and $\angle H(z)$ is easily determined to be $-2\pi\frac{M-1}{2}f$. A filter with such a $\angle H(z)$ is said to have a linear phase property and applies same amount of delay, called group delay, to each sample. This means that the signal will not be distorted by the filter. Furthermore, the magnitude of the transfer function $|H(z)|$ is the absolute value of a sum of cosine functions. Since cosine is an even function, $|H(z)|$ is also an even function, i.e. symmetric around zero. Because of this, in this thesis $|H(z)|$ will only be plotted for $f \in [0, 0.5]$.

The exact details of this manipulation varies slightly depending on whether the filter order is odd or symmetric. For odd-ordered symmetric FIR filters, referred to as type I, there is an odd number of terms and they cannot be paired perfectly two and two. The middle coefficient is left as a constant and the transfer function becomes

$$H(\omega) = e^{-j\omega\frac{M-1}{2}} \sum_{n=0}^{\frac{M-1}{2}} a_n \cos(\omega n) \tag{2.3}$$

with

$$a_n = \begin{cases} h_{\frac{M-1}{2}} & n = 0 \\ 2h_{\frac{M-1}{2}-n} & n = 1 \dots \frac{M-1}{2}. \end{cases} \tag{2.4}$$

Note that $\cos(\omega n) = 1$ for $n = 0$. Symmetric even-ordered filters are referred to as type II and instead have the transfer function

$$H_{II}(e^{j2\pi f}) = e^{-j2\pi f\frac{M-1}{2}} \sum_{n=1}^{M/2} a_n \cos(2\pi f(n - \frac{1}{2})) \tag{2.5}$$

where

$$a_n = 2h_{M/2-n} \quad n = 1 \dots M/2 \tag{2.6}$$

Parameter	Description
f_p	The highest frequency for which the gain must be within the pass-band ripple.
f_s	The lowest frequency for which the gain must be within the stop-band ripple.
δ_p	The pass-band ripple, the maximum allowed value of $\lvert H(z)/G-1\rvert$ for $f \in [0, f_p]$
δ_s	The stop-band ripple, the maximum allowed value of $\lvert H(z)/G\rvert$ for $f \in [f_s, 1]$
G	The static gain of the filter, $\lvert H(0)\rvert$.

Table 2.1 The parameters defining a low-pass filter specification.

The set $\bar{a} = \{a_n\}$ will be referred to as the coefficient set throughout the thesis. Due to the linear phase properties and how the coefficient symmetry can be utilized for more efficient implementations, these are the FIR filter types that will be considered in this thesis. This has the additional benefit that the set of feasible integer solutions, and correspondingly the solution time, is reduced.

2.1.4 Filter specification

When designing a filter, it is of course not the specific coefficients that are of interest but rather the filter properties. While this thesis focuses on low-pass filters, similar terminology carries over for other kinds of filters. The primary parameters of a low-pass filter are the pass-band and stop-band edge frequencies f_p and f_s which define the cutoff. Furthermore, there is the pass-band and stop-band max-ripple, δ_p and δ_s which constrain how much $|H(z)|/G$ can deviate from 1 in the pass-band and 0 in the stop-band respectively, where G is the final parameter, the static gain of the filter. For convenience these parameters are listed in Table 2.1. A graphical depiction of these parameters is shown in Figure 2.1.

In a DSP application, it is common to chain multiple filters of lower order together rather than applying one long filter. Because of this, instead of forcing each filter to have a specific gain, the final gain can be adjusted after the filter chain with a single multiplication. The gain can thus be viewed as a free parameter to be used in the optimization rather than a design parameter.

2.2 ASICs

2.2.1 The design of an ASIC

Application Specific Integrated Circuits, ASICs, are designed to carry out one specific task. They can include both analogue and digital components, though fully digital ASICs using CMOS technology are most common today. The same ASIC design can be implemented using many different standard production processes,

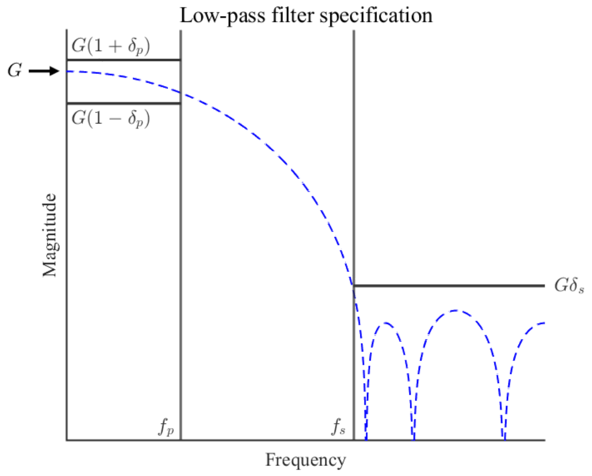


Figure 2.1 Illustration of the low-pass filter specification. The parameters are described in Table 2.1

known as technology nodes, depending on the required size, speed and cost of the ASIC, the smallest technology nodes today utilizing transistors as small as a few nanometers. In addition to the technology node, modern ASICs are usually implemented using standard libraries of predefined transistor structures called cells that implement common logic gates such as NOT and OR. One library can include multiple cells that implement the same logic gate but differ in other properties. For example, if a gate can be allowed to switch slowly, it can usually be implemented with a cheaper or more power efficient cell.

The specialized nature of an ASIC enables heavy optimization and minimal overhead cost per chip but also causes great inflexibility after the production of a new ASIC has started, putting great demand on the ASIC design. Because of this, a lot of effort goes into the design, and it is usually aided by a complex environment of tools for automation, optimization, and verification. The design process is carried out in steps, from more abstract definitions of logic down to the placement of the actual gates. The logic is defined by a designer in a register transfer level (RTL) language which is then compiled into a hardware implementation by a process called synthesizing. The synthesis decides how to implement high level logic such as addition using different cells and architectures depending on different constraints and optimization priorities.

To an extent these levels can be considered and optimized separately. For example, in [Shahein, 2014] power optimization is split into choice of architectures and algorithms, implementation in RTL code, and finally choice and placement of gates. However, the architecture choice might be dependent on the optimization po-

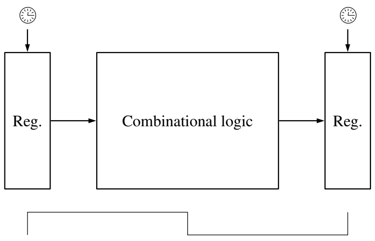


Figure 2.2 The data flow of a clocked ASIC structure over one clock cycle. On the first rising clock edge, the first register outputs the sample to be processed. As the clock cycle progresses, the combinational logic performs calculations. The calculated value is finally output to the second register which samples it on the next rising clock edge.

tential lower down in this hierarchy. This choice is complicated by the automated synthesizing which, though it significantly simplifies the design process, also introduces black-box effects where it can be unclear for the top-level designer what optimizations are performed and ultimately what impact different choices have on the final results. Even non-functional changes in the RTL code such as changing variable names have been shown to have an impact on the synthesis results [Coward et al., 2022]. This together with the fact that the area and power results are only estimations adds a noise floor below which optimizations are pointless. Because of the disconnect between design and result, the design process is usually an iterative process where choices can be made based on feedback from earlier synthesis results.

It is usually considered best-practice to write RTL code at behavior-level and let the synthesis decide on the implementation. However, it has been shown that it is possible to reduce the area of an ASIC with up to 71% while keeping it functionally equivalent only by reformulating the RTL logic [Coward et al., 2022]. This suggests that manually implementing optimized RTL can outperform the automatic optimization of the synthesis.

2.2.2 The structure of an ASIC

The cells in an ASIC can be split into two categories: registers and combinational logic. Registers store data and consist of multiple flip-flop cells that store one or more bits. The combinational logic consists of logic gates. An ASIC that performs synchronous data processing is fed a square clock signal alongside the data stream. To make sure that the input to the combinational logic stays constant over one clock period, a register smples the input value at every clock edge. The combinational logic then calculates its output as the clock period progresses, reaches a result, and outputs it to an output register. The structure of an ASIC can thus be viewed as a series of clocked registers with combinational logic in between. Figure 2.2 illustrates this structure and the data flow during one clock pulse.

The different paths the data flows between registers are called data paths and the number of gates put in series for one data path is called logical depth. It is the data path with the deepest logical depth, adjusted for the speed of the gates, that determines the maximum clock frequency the ASIC can manage. The data path with the deepest logical depth is thus called the critical path. If the critical depth does not meet the timing constraint, the ASIC design needs to be altered. Sometimes, it is enough to move the position of the next register earlier up in the logic, a strategy called re-timing. If this is not possible, additional clocked registers need to be inserted to store intermediate results which is referred to as pipelining.

2.2.3 Power consumption in ASICs

There are many factors that determine the power consumption of an ASIC. To begin with, the ASIC can be constructed using different technology nodes with different power characteristics. There are however scaling algorithms to translate from one technology node to another which means that results from one node should carry over to other nodes [Shahein, 2014]. Secondly, there is a fundamental trade-off between high clock frequency and minimizing area and power consumption, meaning that a tight timing constraint can severely impact the room for optimization [Shahein, 2014]. To meet timing, higher voltages might be needed as well as cell-types that are faster at the expense of higher area and power consumption. The synthesis can also choose to implement logical operations with structures that are more parallelized, performing redundant calculations that increase area and power consumption. As discussed above, if this is not enough, pipelining needs to be introduced, costing registers and increasing the delay of the ASIC. Because of these effects the silicon area and the power consumption grows exponentially at the shorter end of feasible clock periods [Coward et al., 2022] while the data throughput only grows linearly.

The power dissipation in an ASIC stems from multiple sources. These can mainly be divided into two categories: static effects that draw power regardless of whether the state of the transistor changes, and dynamic effects that occur at a state change [Shahein, 2014]. Static power dissipation is caused by short circuits between source and drain at transistors that are below the threshold voltage as well as dissipation in the diffusion and substrate layer of the transistor [Shahein, 2014]. These leakages are mainly affected by the cell choice. The dynamic power is caused by charging and de-charging capacitances inside and outside the cell as well as momentary short circuits at the switching moment. Out of these categories, the dynamic power dissipation dominates. The power consumption of each cell is thus approximately linearly related to the switching frequency of the cell which can be divided into the clock frequency f and the switching activity $\alpha \in [0,1]$. The complete formula is [Demirsoy et al., 2002]

$$P_{\text{switching}} = \alpha C_L V_{DD}^2 f. \tag{2.7}$$

where C_L is the load capacitance and V_{DD} is the voltage driving the cell. If all cells in an ASIC are assumed to use the same voltage and toggle at approximately the same

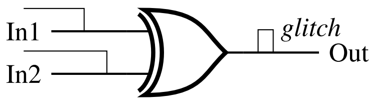


Figure 2.3 The creation of a glitch in a XOR-gate due to timing uncertainty. The inputs are supposed to toggle simultaneously but do not, creating a pulse that will propagate through the circuit and cause additional erroneous switching.

frequency, this means that the number of cells or the silicon area of the ASIC can be viewed as proxy for power consumption. For more detailed analysis however, this measure is too coarse and it has been shown that FIR designs with more adders can be more efficient than those with fewer [Horrocks and Wongsuwan, 1999].

In addition to the fact that the toggle rate can vary between cells, there is an additional effect where the inputs of a cell reach their correct value with a slight timing offset. This causes the cell to toggle when it should not, creating a "glitch" which is illustrated in Figure 2.3. When such glitches appear, they propagate through the combinational logic until they reach a register. These propagating glitches are the dominating cause of glitch power loss [Eriksson and Larsson-Edefors, 2004], suggesting that the logic depth, is an important factor for power efficiency. Though glitching is hard to estimate, there are studies suggesting that glitching accounts for up to 70% of all switching activity in a circuit [Ye et al., 2017], meaning that the effect can hardly be neglected. Glitching has been addressed in the context of FIR filters by considering the average adder depth, defined as the average number of chained adders in the multiplication implementation [Kumm et al., 2023]. Internal bit-width [Shahein, 2014] and a logic depth index called the Glitch Path [Demirsoy et al., 2002] has also been used as pseudo-metrics for glitching.

The importance of switching activity infers another aspect of evaluating power consumption, namely the structure of the data that is processed. For example, a small signal fluctuating around zero represented in two's complement will cause a lot of switching in the input. It is therefore important to consider what kind of data to use for the power estimation, ideally choosing something representative for the real-world use case of the ASIC.

In addition to the dynamic and static power categories, it is common to split the power analysis of an ASIC design into the power drawn by the registers and the power drawn by the combinatorial logic. Ideally, a design should minimize the number of flip-flops, spending its power at the combinational logic which performs the actual calculations. However, since the synthesis can choose between cell-types and implementation structures depending on the timing constraint, this balance is not straight forward. Even though one design can meet timing, introducing a pipeline stage can enable the synthesis to select cheaper cells, producing a net-positive result. The positive effect is further amplified since pipelines restrict glitches.

2.3 Hardware implementation of an FIR filter

As seen in Section 2.1.3, the essence of an FIR filter is a weighted sum, consisting of fixed coefficient multiplication and additions. How these operations are implemented in hardware depends on how numbers and coefficients are represented.

2.3.1 Fixed binary point

Though the signals in radio systems are continuous, a digital signal is of course always discrete, a quantized version of the original signal. In higher-level programming where precision and dynamic range is of importance, a continuous variable is usually represented with a floating-point number where the decimal point can shift depending on how much information is needed to represent the mantissa. In fast applications however, the fixed-point representation is preferred. The number of fractional bits F is set and the total magnitude b of the binary number is

$$b = \sum_{k=1}^{B} 2^{-k+(B-F)} b_k, \tag{2.8}$$

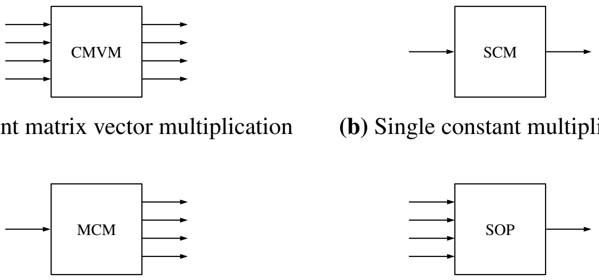
where $b_k \in \{0,1\}$ is the bit-value at index k, sorted from most significant bit to least significant bit. B is the total number of bits in the number, the bit-width. The position of the binary point corresponds to the common factor 2^{B-F}. The number of fractional bits can therefore be regarded as an arbitrary scaling factor and all operations on fixed-point binary numbers can be performed equivalently to integer operations. This can be interpreted as multiplying the operation input with 2^F and dividing again after performing the operation. Regarding power consumption, a higher bit-width increases computation complexity and thus power.

2.3.2 Two's complement

Another key aspect of number representation is how to represent negative numbers. A simple choice is the signed magnitude representation where an extra bit is introduced to keep track of the sign of the number. This simple approach, however, requires explicit logic to keep track of the signed bit and additionally introduces two representations for 0, one positive and one negative. Instead, the two's complement representation is usually favored and is the representation used in this thesis. In two's complement, all numbers are represented as

$$x_{2's} = (2^N + x)(\bmod 2^N)$$

for $x \in [-2^{(N-1)}, 2^{(N-1)} - 1]$. As with the signed magnitude representation, a single bit, in this case the MSB, determines the sign of the number. However, additionally, the representation can handle addition, subtraction, and multiplication without



(a) Constant matrix vector multiplication **(b)** Single constant multiplication

(c) Multiple constant multiplications **(d)** Sum of products

Figure 2.4 Taxonomy of different multiply-sum operations.

adding extra logic, for example

$$\begin{aligned} x_{2's} + y_{2's} &= (2^N + x)(\bmod 2^N) + (2^N + y)(\bmod 2^N) = (2^{(N+1)} + x + y)(\bmod 2^N) \\ &= (2^N + x + y)(\bmod 2^N) = (x+y)_{2's}. \end{aligned}$$

Since integer multiplication can be regarded as repeated addition, this result carries over to multiplication as well. Furthermore, the two's complement representation allows for simple conversion from X to $-X$ by inverting the positive number and adding a one, discarding any overflowing bits from the result. Subtraction in two's complement is thus performed as

$$X - Y = X + (\neg Y + 1). \tag{2.9}$$

Accordingly, subtraction can be regarded as a slightly more expensive operation than addition which is relevant in the context of power optimization.

2.3.3 Fixed coefficient multiplication and addition

Finding efficient hardware-implementations of multiplying inputs with fixed coefficients and outputting sums of the products is a well studied problem. The most general form of this problem, with multiple outputs and inputs, is the constant matrix vector multiplication (CMVM) problem. Algorithms such as [Gustafsson et al., 2004] exist that produce optimized CMVM implementations. However, often sub-problems such as single constant multiplications (SCM) or Multiple Constant Multiplications (MCM) are considered separately to improve algorithm efficiency. A taxonomy of the problem and the sub-problems, inspired by [Aksoy et al., 2014], is shown in Figure 2.4. The basic principle of fixed coefficient multiplication is realizing the operation as a sum of multiple bit-shifted copies of the inputs. Such structures are known as shift-adder trees. Usually multiplication with a negative number is computed as a positive multiplication which is then inverted using two's complement.

For example, a single multiplication with $7x$ can be implemented as $x \ll 2 + x \ll 1 + x$. Such coefficient representation, following the binary number is known as the powers of two (POT) representation. However, by allowing subtractions as well as additions the number of terms in the fixed coefficient multiplication can be reduced. For example, $7x$ can also be computed such as $x \ll 3 - x$, requiring one less addition. This coefficient representation is known as the signed powers of two (SPT), or signed digit, representation. The following notation is used:

$$7 = 100\bar{1}_{\text{SPT}}.$$

The SPT representation is not unique but can be constrained to get uniqueness. One common constraint is not allowing two non-zero signed bits next to each other, known as the canonic signed digit form (CSD), but others such as minimal signed digit exist [Shahein, 2014]. In addition to uniqueness, these constraints guarantee the representation to contain the fewest number of non-zero bits. They do not, however, guarantee the most efficient way of implementing the multiplication. For example, since subtractions are more expensive, implementing 3 as 11 is more efficient than implementing it as $10\bar{1}$.

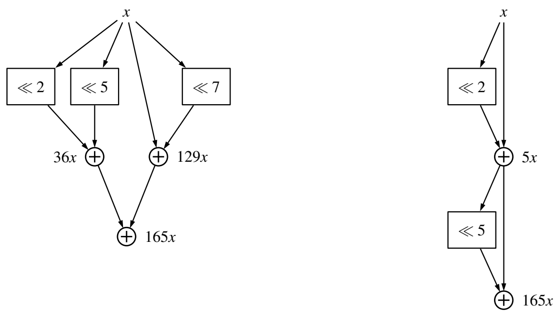
Furthermore, it is important to note that the coefficient space represented by N SPT on the CSD form is not of the same size as the space represented by N POT. While N POT can represent numbers of magnitude $\frac{1}{2}(2^N - 2)$, N CSD can represent numbers of magnitude

$$\begin{cases} \frac{2}{3}(2^N - 1) & N \text{ is even} \\ \frac{2}{3}(2^N - 1/2) & N \text{ is odd.} \end{cases}$$

Another optimization strategy is common sub-expression elimination (CSE), reusing partial results, as shown in Figure 2.5. As seen in the figure, CSE reduces the number of additions but increases the logical depth. The strategy works with both POT and SPT coefficient representation, though POT in general offers more potential for CSE due to only using two tokens $(0, 1)$ compared to three tokens in SPT $(0, 1, -1)$. Due to SPT benefitting from using fewer terms and POT benefitting from increased CSE, there is no consensus on which representation is to be preferred [Shahein et al., 2012]. However, there has been efforts to increase the potential for CSE in SPT coefficient sets by searching directly in the sub-expression space [Yu and Yong Ching, 2007] rather than applying CSE to given coefficients. Because the number of signed digits in this thesis proved to be a better predictor of the coefficient implementation cost, it has been used as the primary index of coefficient complexity.

2.4 FIR filter architectures

There are many architectures for implementing FIR filters, essentially corresponding to different factorization of the FIR difference equation, equation 2.2. Out of these, the direct form (DF) and the transposed direct form (TDF) are the two most



(a) No sub-expression sharing. **(b)** The sub-expression "101" is reused, saving one addition.

Figure 2.5 An example of sub-expression sharing.

common and will be the ones considered in this thesis. Other architectures include hybrid forms which are a middle-ground between DF and TDF [Aksoy et al., 2014], and parallel forms which utilize a polyphase structure to increase throughput [Chung and Parhi, 2002][Månsson, 2021].

In general, the hardware implementation cost of a FIR filter can be split into three sub-blocks: the registers used for delaying the inputs (structural delays, SD), the adders in the fixed coefficient multiplication (multiplication adders, MA) and the adders used for summing up the result (structural adders, SA). In addition to these components, additional registers might be needed for pipelining to meet timing constraints, further increasing the cost of the implementation.

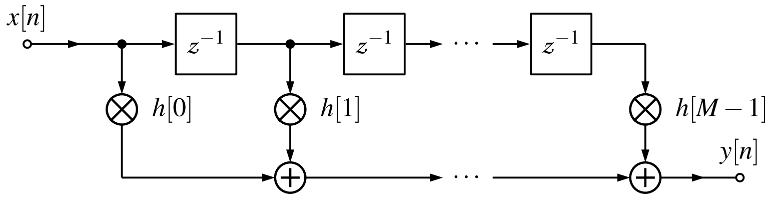


Figure 2.6 A FIR filter of order M, implemented in the direct form.

2.4.1 The direct form

The direct form delays the input first, corresponding to multiplying $x[n]$ with z first in equation 2.2. This creates the structure shown in Figure 2.6. The structure consists of a structural delay line which enters into an array of single multiplications. The products are then added with an adder tree. For symmetric FIR filters, a trivial optimization can be made where the coefficient pairs are added before the multiplications, halving the number of multiplications needed.

The cost of the delay line for a DF implementation is directly determined by the input bit-width and the filter order. For a filter of order M and input bit-width B_{in}, the number of flip-flops needed is

$$N_{\mathrm{SD}}^{\mathrm{DF}} = (M+1)B_{in}, \tag{2.10}$$

assuming the filter has registers on it's input and output. The cost of the multiplications and the adder tree can be estimated from the models in Chapter 3.

The SCM array and adder tree can also be optimized together as a sum-of-products operation, possibly offering better solutions than optimizing each subblock independently. Optimizing sums of products is in general a hard problem, though there are algorithms to find optimized shift-adder-trees, including pipelining, such as RPAG-CMM [Kumm et al., 2017]. Including pipelining drastically increases the complexity of the problem since the improvements in combinational area and possibly cheaper cells need to be balanced against the additional delay and cost of inserting more registers. In this thesis, our own heuristic algorithm will be used to find optimized implementations of given coefficients without considering pipelining.

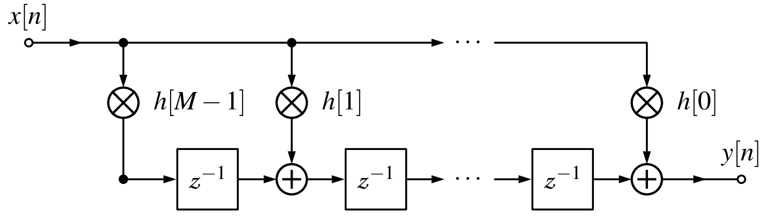


Figure 2.7 A FIR filter of order M, implemented in the transposed direct form.

2.4.2 The transposed form

The transformed direct form instead first multiplies x with the coefficients $\bar{h}$. The corresponding structure is shown in Figure 2.7. The initial operation of multiplying a single input with many constant coefficients is a multiple constant multiplication block. After the MCM, the many outputs are delayed and summed accordingly. An optimization for symmetric FIR filter similar to the one in the DF architecture is possible where each multiplication is used twice.

The main advantage of the transposed form is the potential for optimizing the shift-adder tree in the MCM block. Several algorithms exist for this such as H-cub [Voronenko and Püschel, 2007] which can produce good results in acceptable time for relevant-sized problems and will be used in this thesis. However, timing constraints can, as with the SOP block of the direct form, require pipelining in the MCM-block, again increasing the optimization complexity. Another advantage of transposed form is that the logical depth is smaller than for the direct form due to having no adder tree. This reduces the number of times a glitch can propagate and thus the power that is wasted on glitching.

On the other hand, the transposed form has a drawback where the structural delays store the samples after multiplication and summing, which increases the required number of flip-flops and makes it dependent on the chosen coefficients. Assuming for notation simplicity that the coefficients are integer, the number of flip-flops needed is

$$N_{\mathrm{SD}}^{\mathrm{TDF}} = (M+1)B_{\mathrm{in}} + \sum_{k=1}^{M} \left(\lceil \log_2 \sum_{i=1}^{k} |h_{M-i}| \rceil \right), \tag{2.11}$$

which is much greater than the number of flip-flops needed by the direct form. Furthermore, the total cost of the structural adders follows a similar trend and has been shown to dominate the power consumption of the TDF for filters of order 36 and higher [Ye et al., 2017]. The number of bits that need to be added by the structural adders is, based on [Ye et al., 2017],

$$N_{\mathrm{SA}}^{\mathrm{TDF}} = MB_{\mathrm{in}} + \sum_{k=1}^{M} \left(\lceil \log_2 \sum_{i=1}^{k} |h_{M-i}| \rceil - l_{M-i} \right), \tag{2.12}$$

where l_i is the final bit-shift of the coefficient h_i, i.e., the number of times 2 divides h_i. Additionally, the structural additions need to be implemented one-by-one and cannot be structured in an adder tree which reduces optimization potential compared to the direct form.

2.5 Coefficient selection

The problem of selecting fixed-point binary coefficients for hardware-implemented FIR filters has been studied since at least the late 70s. The linearity of the transfer function and its constraints in terms of the filter coefficients makes the problem suitable for linear programming. Early formulations such as [Lim and Parker, 1983] and [Lu, 2001] used linear programming to improve the filter performance compared to quantized continuous coefficients. Later developments focused on reducing the computation complexity, either by constraining the number of POT or SPT and minimizing frequency ripples or, vice versa, minimizing the number of POT or SPT in a filter fulfilling a certain specification. A mixed-integer linear program for minimizing the number of signed digits was formulated in [Gustafsson et al., 2001].

In parallel to the research on choosing filter coefficients, research has been done on how to implement efficient multiplication with constant coefficients using shift-adder graphs as discussed above in Subsection 2.3.3. Further optimization efforts have incorporated this research into the filter design problem, creating algorithms that minimize the number of adders or the adder depth such as [Dong and Yu, 2011]. As the optimization tools have developed, the problem formulations have grown more fine-grained, the last few years targeting minimizing the number of gates and flip-flops in the implementation, for example [Kumm et al., 2023], [Garcia et al., 2022], [Aksoy et al., 2011]. A more extensive list of algorithms minimizing the implementation complexity of FIR filters can be found in [Aksoy et al., 2014].

Though there has been a lot of research on this subject, there is still room for exploration. A lot of the research focus has been on FPGA implementations, results that might not carry over to an ASIC context. For example, though many articles concern the transposed form due to the optimization potential in MCM blocks, it has been demonstrated that for ASICs, the transposed form occupies more area and consumes more power than direct form filters [Aksoy et al., 2014]. Furthermore, as the required clock frequency increases, the maximum feasible logical depth decreases, offsetting the gain of optimization techniques such as sub-expression sharing. Finally, there are practical concerns when synthesizing RTL code such as which cells and implementations are chosen by the synthesis tool in different situations. For example, the gate-level optimization in [Aksoy et al., 2011] assumes that all additions are implemented using a ripple-carry adder which is efficient area-wise but has a high logic depth. Accordingly, the implementations have high time delays, around 9 ns. This thesis therefore empirically evaluates optimizations in a realistic synthesis environment.

3

Initial Investigations

3.1 Experiment setup

In the initial stage of the thesis, an environment for synthesizing and estimating the power consumption of many parameterized ASICs automatically was developed. Examples of parameters are the choice of coefficients, the clock period, and the input bit-width.

The RTL code is written in SystemVerilog and verified using Cadance's Incisive. The environment performs the synthesis using a tool chain from Synopsys, most importantly the RTL compiler DC Ultra. To reduce overhead time, the ASIC IPs contain a top-level module that instantiates multiple instances of the same submodule with different parameters to be synthesized at the same time. Module-boundary optimization was turned off to avoid this influencing the results of the synthesis. In addition to an ASIC design, the synthesis produces a number of reports which include a report of the silicon area, rounded to the closest square micrometer. After synthesis, the environment estimates the power consumption from a simulation at gate level done in Incisive. Incisive records the switching activity in the ASIC when stimulated by a test signal. This data is then fed to Synopsys' tool PrimeTime which finally produces average power estimates for different power dissipation sources in both the registers and the combinational logic. Because the combinational logic is the target of most optimizations in this thesis, "power" will refer to combinational dynamic power unless stated otherwise.

The test signal used for the power estimation was chosen to be uniform white noise of full amplitude as a worst-case scenario with the maximum amount of switching and since the characteristics of typical data was unknown. The input bit-width was 12 bits for all experiments.

3.2 FIR filter subblocks

To verify optimization metrics, to test the optimization capabilities of the synthesis tool, and to gather data for modelling, subblocks of FIR filters were initially investigated individually. The subblocks that were tested were single constant multiplications, multiple constant multiplications, adder trees and manual implementations of sums of products. Since register sizes are more directly determined by the input bit-width, coefficient set and filter architecture, the focus was on the combinational logic. Therefore, the structural delay subblocks were not investigated.

3.2.1 Single constant multiplication

The SCM subblock used in direct form filters was tested by synthesizing single multiplications with all coefficients between -1024 and 1024 at a clock period of 2 nanoseconds. The area and power results are compared to the number of signed digits in Figure 3.1. The figure shows that there is a clear correlation for both area and power. Each signed digit costs circa $3.6\,\mu\text{m}^2$ in area and $4.2\,\mu\text{W}$ in power. Additionally, the figure shows that negative coefficients were in general more expensive than positive, being on average $0.96\,\mu\text{m}^2$ larger and consuming $1.6\,\mu\text{W}$ more power. This constant offset can be explained by the handling of negative numbers in the two's complement representation, requiring an extra inversion and addition.

While the number of signed digits predicts the coefficient cost well, there is still a rather large variation within coefficients with the same number of signed digits. A secondary feature to consider is the coefficient width, defined as the difference between the index of the first and last non-zero SPT bit. For example, the coefficient $15 = 1000\bar{1}_{\text{CSD}}$ has a coefficient width of four. Wider coefficients need to compute more bits and should therefore be more expensive. Figure 3.2 shows two heatmaps with coefficient width on the x-axis and area on the y-axis. The left plot shows data for all positive coefficients with three signed digits and the right plot shows the data for four signed digits. While the trend is not as obvious as with signed digits, the coefficient indeed correlates positively with the coefficient width. For power, the results showed no correlation with the coefficient width.

The size of single multiplications will vary with the timing constraint and the total logical depth. However, the empirical results described above can serve as a model for estimating the area and power required for single multiplications. For coefficients outside the tested range, a linear model can be applied. A simple model would be to only include the number of signed digits and sign of the coefficient c in a linear regression model,

$$\hat{y}_{\text{SPT}}(c) = \theta_0 + \theta_1 \text{SPT}(c) + \theta_2 \text{sign}(c) + e, \tag{3.1}$$

where the error e is assumed to be normally distributed and where $\text{sign}(c)$ takes the value 0 for negative coefficients and 1 for positive. However, the model can be extended. If subtractions are more expensive than additions, negative signed digits should be more expensive than positive and should be included separately. Further, as discussed above, the coefficient width should be included:

$$\hat{y}_{\text{full}}(c) = \theta_0 + \theta_1 \text{SPT}_+(c) + \theta_2 \text{SPT}_-(c) + \theta_3 \text{width}(c) + \theta_4 \text{sign}(c) + e. \tag{3.2}$$

The parameters of these models, fitted for both area and power, are presented in Table 3.1. The full models show an improvement over the SPT models, increasing

Model	R^2	RMSE	θ_0	θ_1	θ_2	θ_3	θ_4
Area [µm^2], SPT	0.898	1.17	-1.77	3.59	-0.96	–	–
Area [µm^2], full	0.936	0.924	−2.97	2.76	3.417	0.33	−0.53
Power [µW], SPT	0.875	1.56	2.01	4.22	-1.61	–	–
Power [µW], full	0.919	1.25	-3.29	3.24	4.22	0.32	-0.95

Table 3.1 Results of the SCM linear regression models given by Equation 3.1 in the SPT case and Equation 3.2 in the full case. It can be seen that area and power grows with the number of signed digits and coefficient width. The results also indicate that negative coefficients are more expensive and that a negative signed digit is more expensive than a positive signed digit. The reasons for this are discussed in Section 3.2.1.

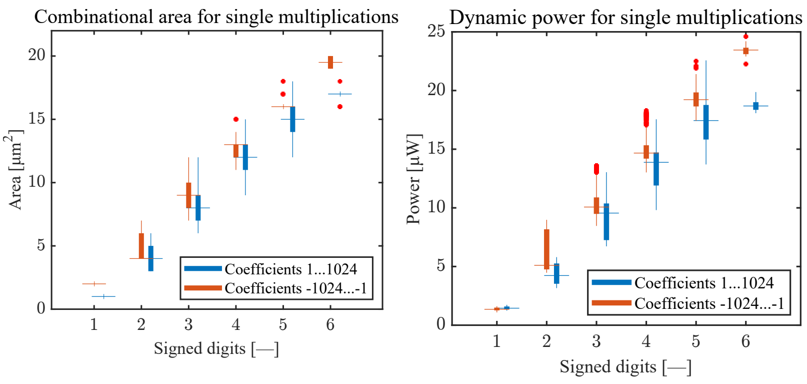
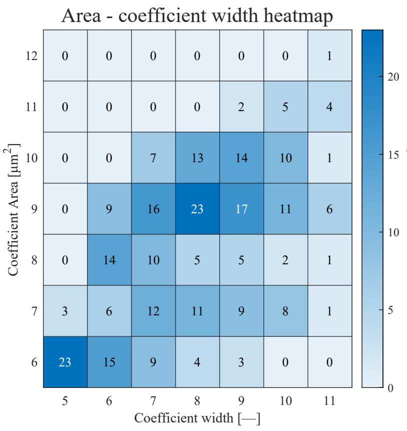


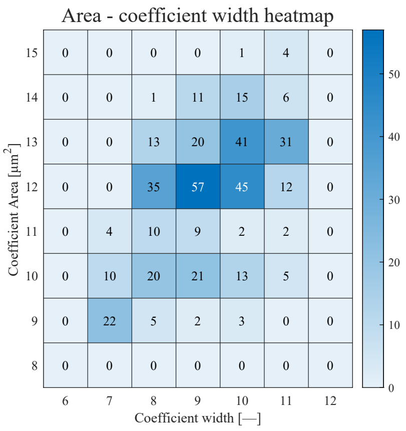
Figure 3.1 Area and power consumption results for single constant multiplications, compared to the number of signed digits in the coefficients. The relation is clearly linear and the parameters of a linear model is presented in table 3.1.

the R^2 value from 0.898 to 0.936 for area and from 0.875 to 0.919 for power. The full model also confirms that the cost of negative signed digits are more expensive, costing more than 24% in area, and 30% more in power. Finally, it can be noted that the impact of bit-width is rather small compared to the impact of signed digits. This makes sense when considering that a signed digit adds an addition or subtraction while the coefficient width affects the cost of implementing the operations slightly.

3.2.2 Multiple constant multiplications

The MCM subblock was tested similarly to the SCM subblock. Two hundred multiple constant multiplications with randomized coefficient sets with varying number of signed digits were synthesized. Twelve signed, possibly zero, coefficients with a maximum magnitude of 2047 were used. In addition to synthesizing the MCM blocks with a clock period of both 2 nanoseconds, they were also synthesized at 10





(a) Three signed digits.
Pearson correlation coefficient $\rho = 0.56$

(b) Four signed digits.
Pearson correlation coefficient $\rho = 0.53$

Figure 3.2 Two heatmaps of the area and coefficient width for coefficients with 3 and 4 signed digits respectively. The number in each cell indicates the number of coefficients with the corresponding area and coefficient width. The results indicate a postive correlation and the coefficient width is therefore included in the full SCM model of Equation 3.2.

nanoseconds to test the synthesis sub-expression-sharing optimization with a negligible timing constraint. The results were then compared to the predictions from the above full SCM model to estimate the gain of performing multiple constant multiplications on one input rather than single multiplications on many inputs. This gain achieved by the synthesis tool was then compared to the percentage of adders saved using the Hcub algorithm in [Voronenko and Püschel, 2007] as a measure of theoretical gain.

Figure 3.3 shows the area and power per sample results compared to the total number of signed digits in the coefficient set. Since the power is linearly dependent on the clock frequency, as shown in equation 2.7, the power results are divided with the clock frequency to get comparable results for different clock periods, called power per sample rate, or simply adjusted power. As with the SCM block, there is a clear correlation between area and power and the number of signed digits. A linear regression gives a cost of each SPT of circa 3.4 μm^2 at 2 ns, which is similar to the result from the SCM SPT model, 3.6 μm^2. The adjusted power cost is also similar but slightly higher, 9.1 µW/Gsps compared to 8.4 µW/Gsps.

Comparing the clock frequencies, the area is on average 10.3% smaller at 10 ns. For the adjusted power however, the improvement is only 4.5%. Additionally, the synthesis reports show that the combinational logic of the 10 ns MCMs is done when there is about 6 ns left of the clock cycle. While it is clear that some additional optimizations are done at the lower clock frequency, this suggests that the timing constraint is not a limiting factor on the optimization at 2 ns for these rather simple

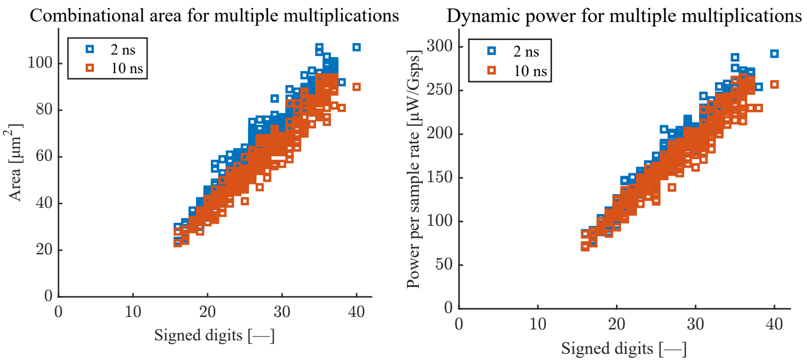


Figure 3.3 Area and power consumption per sample rate for multiple constant multiplications, compared to the number of signed digits in the coefficient set. As with the single multiplications in figure 3.1, the trend is linear. The lower clock frequency shows some improvement over the higher clock frequency, though only on average 10.3% with regards to area and 4.5% with regards to power.

operations.

Figure 3.4 compares the area and power of the MCM blocks to the sum of applying the above SCM model on all coefficients. This is done for the 2 ns results to get a fair comparison. The figure shows that MCM is better in terms of area for most cases, though only on average 7.6%. For power, the MCM results are worse than the SCM model. This indicates that there is less optimization than what could be expected. Considering only area where there were some savings, the improvement is still much less than the reduction of adders by the Hcub algorithm, clearly shown in the histogram in Figure 3.5. On average, the MCM structure produced by Hcub reduced the number of adders with 38.6% which should translate to similar area gains. This can be to the average area reduction of 7.6%.

In conclusion, the MCM structure does not seem to offer any benefit over performing multiple single multiplications. Why this happens, however, is unclear and could be both due to a lack of sub-expression sharing or sub-expression sharing not actually being beneficial in this scenario due to for example timing constraints. However, since the 10 ns results were not significantly better and did not use the entire clock period, it seems unlikely that the clock frequency is the constraining factor.

3.2.3 Adder trees

A subblock consisting of summing eight inputs of different bit-lengths and offset by bit-shifts was implemented to simulate the structural adders of the direct form filter architecture. One hundred randomized such blocks were synthesized at a period of 2 nanoseconds. The area and power results were then compared to the total number of summed bits in the adder tree. The results together with a linear fit are shown in

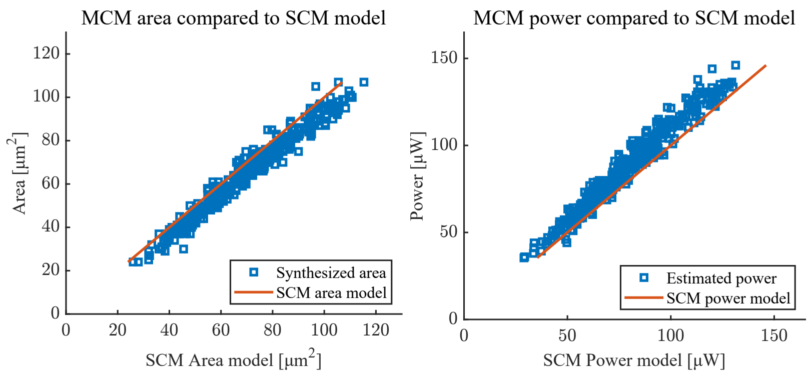


Figure 3.4 The results of the MCM operations (blue markers), synthesized at 2 ns, compared to the full SCM model from Section 3.2.1 (red line). While the area results show some gain from MCM compared to many SCM, the power results are worse.

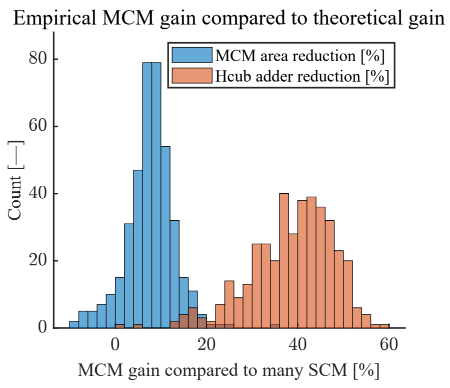


Figure 3.5 A histogram of the percentage of the area saved by MCM in Figure 3.4, and the percentage of adders saved by the Hcub algorithm for the same coefficient sets. Under the assumption that area is proportional to the number of adders, these histograms should overlap approximately if an optimized MCM structure is utilized in the implementation. They do not, indicating either that such a structure is not used or that the area is not proportional to the number of adders. Most likely, an MCM structure is not used.

Figure 3.6. The linear regression model fits the data very well, achieving a R^2 value of 0.986 for area and 0.959 for power. The slope of the area model is 0.28 µm^2 per additional bit in the adder tree. For the power, the slope is 0.33 µW. Comparing this to the impact of signed digits in Table 3.1, each summed bit is about 10-20 times cheaper than a signed digit. This is reasonable since one signed digit essentially adds one addition which includes 10-20 bits. Still, it is clear that there is some optimization potential in selecting coefficients with a lower width, especially since a lower width also was shown to reduce the multiplication cost in Subsection 3.2.1.

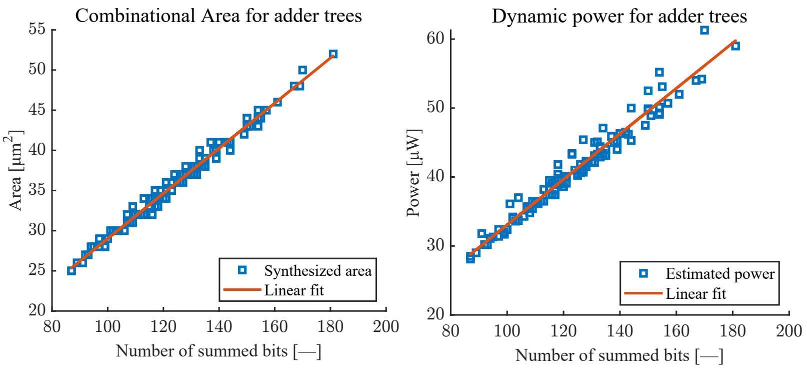


Figure 3.6 The area and power results for an 8-input adder tree synthesized at 2 ns, compared to the number of summed bits. The red line is a linear fit to the data with the slope 0.28 µm^2 for area and 0.33 µW for power.

3.2.4 Manual sum of products

Since the MCM block showed little evidence of optimizing many multiplications together using sub-expression sharing, the investigations of the SOP subblock focused on determining the potential of outperforming RTL code that is written on a behavioral level by writing RTL code that manually implements sub-expression sharing. A minimal example, selected to have good sub-expression-sharing potential, was implemented both on a behavioral level and with the sub-expressions implemented manually. In this way, the automatic optimization can be compared to manual optimization. The following coefficients were chosen:

$$
\begin{aligned}
3+5\cdot 2^{10} &= 5123 = 1010000000010\bar{1}_{\text{CSD}} \\
3+5\cdot 2^{9} &= 2563 = 101000000010\bar{1}_{\text{CSD}} \\
3+5\cdot 2^{8} &= 1283 = 10100000010\bar{1}_{\text{CSD}} \\
3+5\cdot 2^{7} &= 643 = 1010000010\bar{1}_{\text{CSD}} \\
3+5\cdot 2^{6} &= 323 = 101000010\bar{1}_{\text{CSD}} \\
3+5\cdot 2^{5} &= 163 = 10100010\bar{1}_{\text{CSD}},
\end{aligned}
$$

forming the sum of products $y = \sum_{k=0}^{5}\left(3+5\cdot 2^{(5+k)}\right)x_k$. If the inputs are simply shifted and summed using the CSD representation, 23 additions and subtractions are needed. However, 10 additions can be saved by utilizing the common sub-expressions 101_{CSD} and $10\bar{1}_{\text{CSD}}$. The SOP is then implemented as

$$
\begin{aligned}
&x_{\text{sum}} = \sum_{k=0}^{5} x_k, \quad x_{\text{shift}} = \sum_{k=0}^{5} \left(x_k \ll (5+k)\right), \\
&y = (x_{\text{sum}} \ll 2) - x_{\text{sum}} + (x_{\text{shift}} \ll 2) + x_{\text{shift}}.
\end{aligned} \tag{3.3}
$$

Table 3.2 provides the area results for three different implementations, synthesized at 2 ns and 4 ns respectively. Power estimates are not included since the experiment was intended as a proof of concept and the area results were convincing enough. **Manual sub-expressions** is implemented in accordance with (3.3). The **Linear Combination** implementation performs the sum of products in the most straightforward way as

$$y = 163x_0 + 323x_1 + 643x_2 + 1283x_3 + 2563x_4 + 5123x_5.$$

The **Shift & Add** implementation utilizes the CSD representation of the coefficients, adding and subtracting shifted version of the input as

$$
\begin{aligned}
y =& (x_0 \ll 7) + (x_0 \ll 5) + (x_0 \ll 2) - x_0 \\
&+ \ldots + (x_5 \ll 12) + (x_5 \ll 10) + (x_5 \ll 2) - x_5.
\end{aligned}
$$

The results of the Linear Combination and Shift & Add implementations are identical, indicating that Shift & Add is how the SOP is implemented automatically, or at least that the different implementations are analyzed and optimized in the same way. In contrast, the manual sub-expression implementation has a 25% lower area at 2 ns and 37% at 4 ns. These improvements can be compared to the 10 additions saved compared to 23 initial additions, an improvement of 43%. While the results are not quite as good, they still show that it is possible to outperform the automatic synthesis and the results are convincing enough that it seems feasible also when the example is not handpicked. Additionally, the number of saved additions should be some indication of which coefficient sets have good potential when implemented manually.

Minimal example

Implementation	Comb. area, 2 ns	Comb. area, 4 ns
Linear Combination	79 µm^2	75 µm^2
Shift & Add	79 µm^2	75 µm^2
Manual Sub-expressions	59 µm^2	47 µm^2

Table 3.2 Table of results of combinational area for different implementations of minimal example. While the Shift & Add implementation results are to those of the Linear Combination implementation, the optimized manual sub-expression implementation performs significantly better. Since the experiment was only meant as a proof-of-concept, no power estimates were made.

4 Filter-level optimization

In this chapter, optimization of full filters is considered, building on the theory and the results from the previous chapter, as well as synthesis results. First, the direct form is compared to the transposed form. Then, a mixed-integer linear progam that minimizes the number of signed digits in the coefficient set is formulated and evaluated for five filter specifications. Finally, a criterion is suggested for selecting between solutions with the same number of signed digits.

4.1 Direct and transposed form comparison

As mentioned above, the transposed form architecture is more expensive in terms of structural delays while allowing for better optimized multiplication by increased sub-expression sharing potential. However, in Section 3.2.2, little benefit was observed from the MCM optimization of the synthesis tool. To compare the two architectures, a symmetric filter of order 25 with the unique coefficients

$$\bar{c} = \{256, 192, 57, -36, -41, 0, 22, 10, -7, -8, 0, 4, 1\}$$

was implemented in both transposed and direct form and synthesized at 4 ns to reduce the impact of the timing constraint. Using Hcub [Voronenko and Püschel, 2007], the number of additions can be reduced from 10 to 7 by utilizing sub-expression sharing. To make sure that sub-expression sharing was utilized, the shift-adder tree produced by Hcub was additionally implemented manually.

The results, shown in Table 4.1, show that the direct form outperforms the transposed form in every aspect. The greater number of flip-flops needed for the transposed form is expected and can be calculated using equation 2.10 and 2.11. The larger number of flip-flop then of course carries over to a greater register area and power consumption. More surprisingly, the combinational area was not lower for the transposed form. The explicit implementation was somewhat better than the automatic implementation, although not near the 30% theoretical reduction. Still, it was 18% larger and consumed 12% more energy than the direct-form implementation. This is partly explained by the results in Section 3.2.2 showing the limited gain of utilizing MCM optimization but also that the structural adders make up a significant part of the combinational logic. Using the SCM data from the previous

Filter type	Comb. area	Comb. power	Reg. bits	Reg. area	Reg. power
TF	131	79	506	197	69
Manual TF	127	76	506	197	69
DF	108	71	324	109	41
Best ratio	1.18	1.12	1.57	1.81	1.68

Table 4.1 Comparison between direct and transposed form for one S1a coefficient set synthesized at 4 ns. The area results are given in square micrometers and the power results are given in microwatts. The results show that the direct form (DF) outperforms the transposed form (TF) in every aspect. It is surprising that the combinatorial area was not lower for the transposed form.

section, the multiplications needed for the filter can be estimated to need 47 μm^2, less than half of the total combinational area. The rest of the logic consists of structural adders which can be better optimized in the direct form. As mentioned in the theory, the direct filter architecture performs the additions with an (optimized) adder tree while the transposed performs the additions one-by-one. It should be noted that the lack of glitching in the power estimations could work in favor of the direct form, due to the potentially shorter logical paths of the transposed architecture. Still, the transposed filter architecture appears unlikely to ever outperform the direct form in our synthesis environment. Further investigations will therefore focus on the direct form.

4.2 Coefficient optimization

The optimization strategy consists of two parts. First, a Mixed-Integer Linear Program is applied that minimizes the number of signed digits for a filter that fits a low-pass specification following Section 2.1.4. Then, if there are multiple optimal solutions with the same number of signed digits, a secondary heuristic selection criteria is used to determine the final candidate.

4.2.1 Minimizing the number of signed digits in the coefficient set

The MILP is formulated for symmetric type I or type II FIR filters with N unique coefficients of bit-width B. All coefficient bits are fractional, corresponding to setting $F = B$ in Equation 2.8. As pointed out in Subsection 2.3.1, this only scales the coefficients with a certain factor and is ultimately an arbitrary choice. The optimized vector consists of integer SPT bits $b_{n,i}$, continuous help variables that measure the magnitude of the SPT bits, $b_{n,i}^{\text{mag}}$ and the continuous gain, G. The gain can either be set to a fixed value, usually 1, or constrained to a range. The frequency domain constraints are defined by sampling the absolute value of equation 2.3 for type I and equation 2.5 for type II at K frequencies. These frequencies can be selected

Name	Range	Indices	Description
$b_{n,i}$	$\{-1,0,1\}$	$n=1\dots N,\ i=1\dots B$	SPT bits.
$b_{n,i}^{\text{mag}}$	$[0,1]$	$n=1\dots N,\ i=1\dots B$	Magnitude of the SPT bits.
G	$[G_{\min},G_{\max}]$ or 1	–	The static gain of the filter.
f_k	$F_p=[0,f_p],F_s=[f_s,0.5]$	$k=1\dots K$	Sampled frequencies.

Table 4.2 A description of the variables used in the MILP. N is the number of unique coefficients, B is the bit-width of the coefficients, and K is the number of points where the transfer function is sampled.

arbitrarily but were chosen to be evenly spaced in the pass band and stop band. The term of this function with index n, sampled at frequency f, will be referred to as $c(f,n)$ in the MILP formulation. Following [Gustafsson et al., 2001], a canonical signed digit constraint is applied to reduce the feasible space and avoid degenerate solutions. A description of the variables used in the MILP is given in Table 4.2 and the full optimization problem is formulated as following:

$$\begin{aligned}
\text{minimize} \quad & \sum_{n=1}^{N}\sum_{i=1}^{B} b_{n,i}^{\text{mag}} & \\
\text{subject to} \quad & \sum_{n=1}^{N} c(f_k,n)\sum_{i=1}^{B} 2^{-i} b_{n,i} \le G(1+\delta_p) & f_k \in F_p \\
& \sum_{n=1}^{N} c(f_k,n)\sum_{i=1}^{B} 2^{-i} b_{n,i} \ge G(1-\delta_p) & f_k \in F_p \\
& \sum_{n=1}^{N} c(f_k,n)\sum_{i=1}^{B} 2^{-i} b_{n,i} \le G\delta_s & f_k \in F_s \\
& \sum_{n=1}^{N} c(f_k,n)\sum_{i=1}^{B} 2^{-i} b_{n,i} \ge -G\delta_s & f_k \in F_s \\
& b_{n,i}^{\text{mag}} \ge b_{n,i} & \\
& b_{n,i}^{\text{mag}} \ge -b_{n,i} & \\
& b_{n,m}^{\text{mag}} + b_{n,m+1}^{\text{mag}} \le 1 & m = 1\dots B-1 \\
& G_{\min} \le G \le G_{\max} &
\end{aligned}$$

Using modern optimization software, this MILP can be solved in reasonable time, often a couple of seconds, for the problem sizes considered in this thesis. Additionally, the optimization software can be configured to save a number of the best solutions. In this way, the search space can be explored more thoroughly and multiple good candidates can be found. However, due to the flexible gain, there will be some degenerate solutions differing only by factors of two. These are essentially equivalent since a multiplying with a factor of two can be achieved with a bit-shift that is free implementation-wise. Because of this, such solutions are filtered out after the MILP is solved. The coefficient sets that are accepted are also divided such that there is no common factor of 2 in the set.

4.2.2 Secondary heuristic coefficient criteria

If there are multiple solutions remaining, one feasible strategy might be to just synthesize all candidates. Another strategy is to apply a secondary ranking calculated from some features of the coefficient sets. In this thesis, four such features are considered:

- Number of zeros, N_z. Due to how the symmetry of the filter coefficients are utilized in direct form, each zero-valued coefficient saves one extra addition.
- Dynamic range, $\mathrm{DR} = \log_2(|c_{\max}/c_{\min}|)$ where $c_{\max}$ and $c_{\min}$ are the non-zero coefficients with the largest and smallest amplitude. A low dynamic range should allow the adder tree to be more compressed.
- Fraction of all signed digits in the coefficient set that are negative, SPT_-. As discussed in Chapter 3, negative signed digits are more expensive to implement.
- Sum of the coefficient bit-widths B. As discussed in Chapter 3, adding fewer bits produces a simpler adder tree. The number of bits to be added in the direct form filter is determined by the bit-width of the coefficients.

After these features are calculated, they are min-max normalized, producing $\hat{N}_z$, $\hat{\mathrm{DR}}$, $\hat{\mathrm{SPT}}_-$ and $\hat{B}$. They are then combined in a weighted sum to produce the final score r:

$$r = -\theta_0 \hat{N}_z + \theta_1 \hat{\mathrm{DR}} + \theta_2 \hat{\mathrm{SPT}}_-. + \theta_3 \hat{B} \tag{4.1}$$

Due to lack of data to do a proper fit of this model, in this thesis all θ were set to 1. This heuristic ranking can be compared to the empirical SCM model which estimates the cost of the multiplication only.

4.2.3 Results

This section presents the results from applying the coefficient optimization on the specifications given in Table 2.1. $K = 50$ frequency points were used for the frequency constraints and $B = 12$ was used as the input bit-width. The optimization was performed both with the gain fixed to one and with the gain constrained to the range $[1/16, 2]$. The number of coefficients and the filter type varied between the specifications.

Name	f_p, f_s	δ_p, δ_s	Found in
S1a	0.15, 0.25	0.00645, 0.00645	[Kumm et al., 2023]
Y1	0.15, 0.25	0.0316, 0.0316	[Ye et al., 2017]
A	0.075, 0.1125	0.01, 0.01	[Shahein et al., 2012], [Aktan et al., 2008]
E	0.25, 0.4	0.001 0.001	–
G	0.15, 0.35	0.01, 0.0009	–

Table 4.3 Investigated filter specifications.

As an initial illustrating example, filter specification S1a will be investigated more thoroughly. Using the parameters in [Kumm et al., 2023], with 13 unique coefficients, a coefficient bit-width of 9 and flexible gain, 6207 unique solutions could be found. The distribution of the number of signed digits can be found in Figure 4.1 and shows that the median is 30 while the optimal solution has 21 signed digits, an improvement of 30%. Furthermore, the distribution shows the importance of choosing the coefficients carefully as the number of good solutions is vastly smaller than number of average solutions. The specification could be met using only 12 unique coefficients. However, the optimal solution then required 41 signed digits, essentially doubling the complexity of the multiplication logic while only saving two registers and one addition.

Rather, it is beneficial to allow the MILP greater flexibility if possible. For example, not allowing flexible gain increases the optimal solution from 21 signed digits up to 28. Additionally, the allowed coefficient bit-width was increased from 9 to 13. The total number of solutions found then had to be capped at 10000 and the number of good solutions (best and second best with regards to signed digits) increased from 2 to 65. The amplitude of the transfer function, normalized by the gain, of four of these good solutions is plotted in Figure 4.2, demonstrating that the proposed MILP produces correct solutions. Furthermore, Figure 4.3 shows the corresponding unique coefficient set and shows that, except in magnitude, the good solutions are almost identical.

Out of all solutions, a selection was synthesized at 2 ns and 4 ns. The area and dynamic combinational power consumption is shown in Figure 4.4, clearly showing the linear impact of reducing the number of signed digits. Compared to the worst candidate with one less coefficient but 41 SPT, the savings are up to 50% for 2 ns for both power and area. Comparing the best results with a solution closer to the median, with 29 SPT, the reduction is 25.8% for area and 26.0% for power at 2 ns and 24.7% and 22.7% respectively at 4 ns. While not as extreme, this is still a considerable improvement. A linear regression on the area shows that the cost per SPT at 6.2 μm^2 is almost twice of what was measured for the SCM and MCM blocks. This could indicate that reducing SPT also reduces the cost of the adder tree. Finally, the worst choice for 21 signed digits was worse than the best choice for 22 signed digits, illustrating the need for a secondary selection criteria to achieve the final circa 5% of optimization potential.

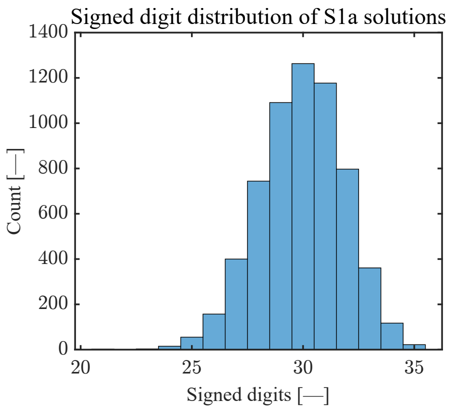


Figure 4.1 The distribution of signed digit solutions with a coefficient bit-width of 9 for the filter specification S1a.

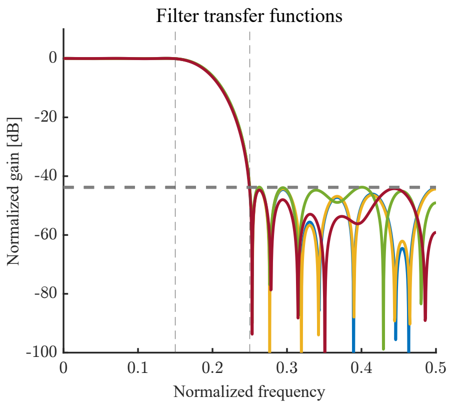


Figure 4.2 The amplitude of the transfer function of four of the solutions to filter specification S1a. All amplitudes are normalized by the filter gain.

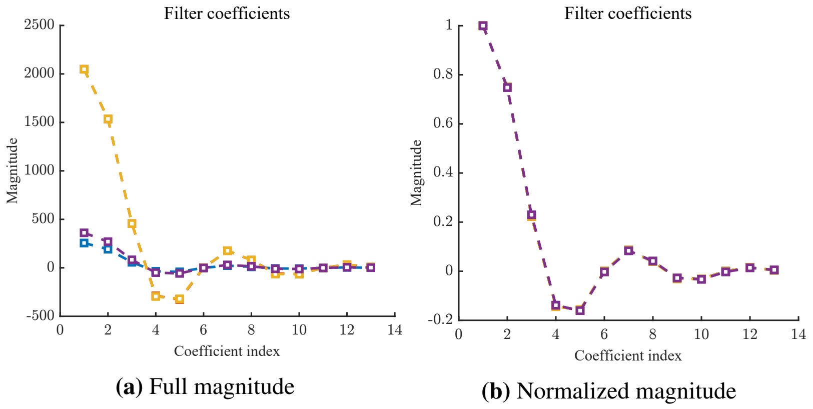


Figure 4.3 The unique coefficients of the filters shown in Figure 4.2. With normalized magnitudes there is significant overlap of the curves as the variation in coefficient magnitude between the filters are very small.

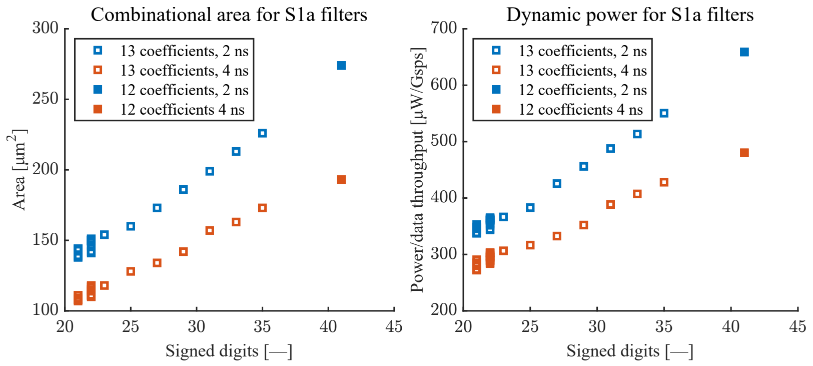


Figure 4.4 Illustration of the area SPT dependency for filter specification S1a. Compared to the median, the optimal signed digit solution provides an approximate 25% reduction in area and power.

Name	Order	Opt. SPT	Opt. SPT, unit gain	No. Opt. SPT	No. Opt. SPT+1.
S1a	23	21	28	2	63
Y1	38	26	31	8	147
A	43	33	37	1	148
E	24	24	24	1	36
G	15	15	Infeasible	1	6

Table 4.4 Table of results for the different filter specifications. No. Opt. SPT refers to the number of optimal signed digit solutions with flexible gain and No. Opt. SPT+1 refers to the number of second-to-optimal solutions.

Next, the optimization of all specifications is discussed. Table 4.4 shows the optimal number of signed digits for all specifications for both flexible and unit gain, together with the required filter order and how many unique optimal and second-to-optimal solutions could be found. The results show that on average, allowing flexible gain decreases the optimal number of signed digits with 20%, which is in the same order of magnitude as performing the SPT optimization at all. For the G specification, unit gain could not be achieved with the same order as the flexible gain, further increasing the benefit of flexible gain. Finally, the number of unique solutions increases significantly when increasing the number of signed digits with only one. This larger solution space could contain candidates that outperform the optimal solutions due to other factors than the number of signed digits.

Due to the clear demonstration of SPT impact for S1a and the effort of synthesizing, only the optimal and a random selection of second-to-optimal solutions were synthesized for the other specifications. All selected solutions were synthesized at 2 ns and 4 ns. The resulting area and power per sample rate is shown and compared to the number of signed digits in Figure 4.5. The linear dependency of area and power on signed digits still holds between filter specifications, meaning that changing the number of coefficients (without changing the number of signed digits) does not appear to have a significant impact on the cost of the combinational logic. the

The timing constraint has a significant impact on the results. The area at four nanoseconds is smaller than the area at two nanoseconds but not that much smaller that it compensates for the halved data throughput. If the area is adjusted similarly to the power, it is on average is 56% larger per sample rate. For the adjusted power, however, the consumption is on average 19% lower at the lower timing constraint. This suggests that parallelizing the FIR filter is a viable option for power optimization, at the cost of increased area.

For register area and adjusted power, the cost instead mostly depends on the number of coefficients. Figure 4.6 shows that, as expected, both the area and power of the registers grows linearly with the number of unique coefficients, costing about 7.9 μm^2 or 38 μW/Gsps per coefficient at 2 ns. Comparing this with the signed digit cost of 7.2 μm^2 and 16 μW/Gsps means that to gain from adding one unique coefficient, only about 3 SPT need to be saved. This is significantly less than the actual

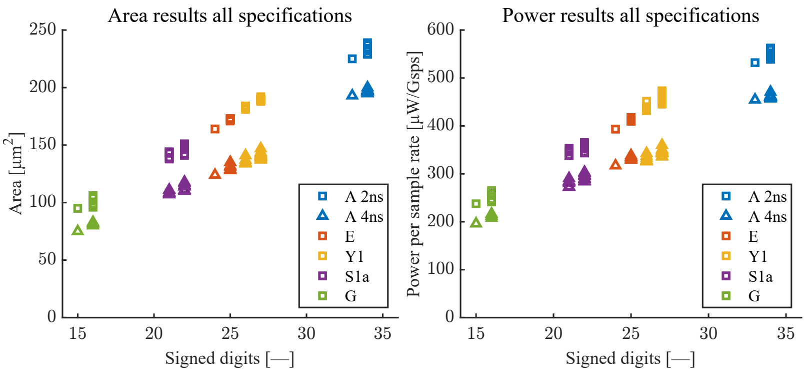


Figure 4.5 Area and power per sample rate for all filter specifications. Even with varying number of coefficients, the cost of the combinational logic can be considered linearly dependent on the number of signed digits.

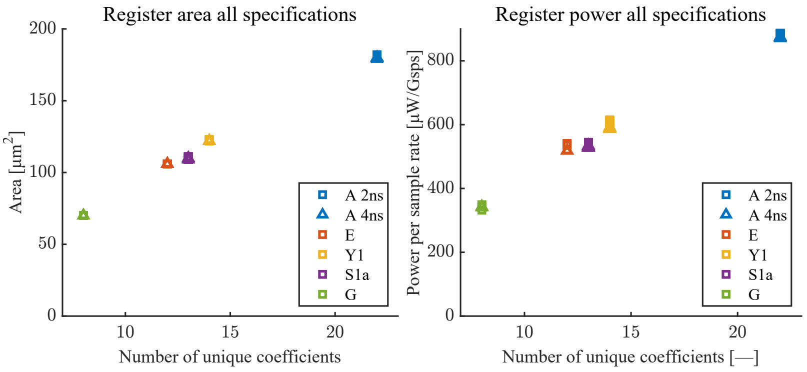


Figure 4.6 Register area and power compared to the number of unique coefficients in the filter.

saving of 20 SPT in the S1a example. Regarding the timing constraint, it does not appear to affect neither area nor power of the registers significantly. Doubling the clock period doubles the adjusted area while the power per sample stays the same due to the halved switching activity.

Lastly, the results of Figure 4.5 show that while there is a spread between solutions with the same number of signed digits, selecting the best of the optimal SPT solutions always gives the best performance. The histograms in Figure 4.7 confirm this, showing the area and power distribution of solutions relative to the best solution for the specification. The figure includes some results for degenerate solutions which explains why the numbers do not align with Table 4.4. For three out of five filter specifications there was only one unique optimal solution. For the other two, S1a and Y1, equation 4.1 was used to calculate a ranking score. In addition to this, a second ranking was calculated from applying the SCM model from Section 3.2.1 to the coefficients.

Figure 4.8 shows these rankings together with the difference in area and power. For S1a, there are only two unique solutions, though there is also a variation of 1 µm^2 and 2 µW in the results for the coefficient sets that differ only by a factor of two which should be essentially equivalent. The heuristic ranking manages to separate these and determine which solution is better. The SCM model does not perform well at all. The results are similar for Y1a with eight unique solutions. The heuristic model can not rank the solutions in order but manages to rake out bad candidates. The SCM model, on the other hand, is hardly better than random.

The bad results of the SCM model can be explained by multiple factors. To begin with, as pointed out earlier, the multiplication only makes up about half of the combinational logic and can not be the only logic considered when ranking the solutions at this level of detail. Furthermore, the synthesis area results are rounded to the closest square-micro meter which introduces noise that becomes significant when considering differences in area that are at square-micro meter level. Finally, the implementation of the multiplication is heavily dependent on the logic depth which limits the accuracy of using earlier synthesis results. The heuristic ranking rather provides some measure of logic complexity which seems to better determine which solution is better. While not perfect, the heuristic criterion can be used to either reduce the number of candidates to be synthesized or to find a best-effort coefficient set that will likely be one of the best candidates. Applying a best-effort strategy is further motivated by the fact that the results at this stage of the ASIC design are still estimations. Selecting between candidates that only differ a couple of µm^2 can be viewed to be within the margin of error of these estimations.

Finally, it should be noted that the size of the data set considered in this investigation is quite small, meaning that the results cannot be considered to be particularly conclusive.

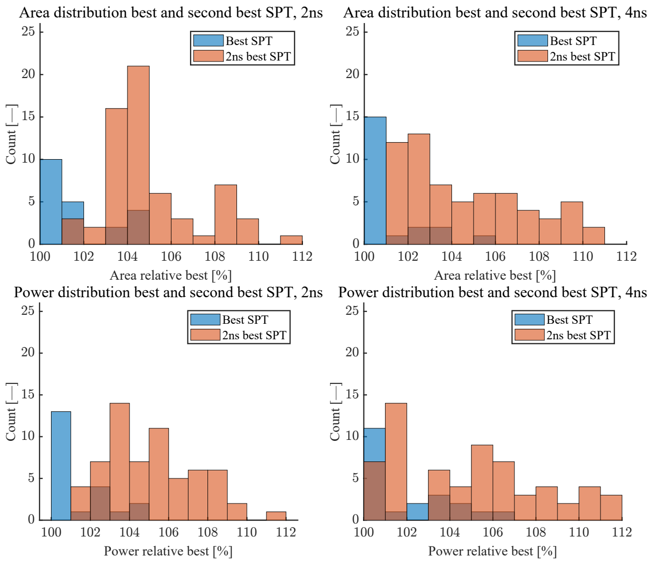


Figure 4.7 Relative distribution of best and next best candidates for all filter specifications.

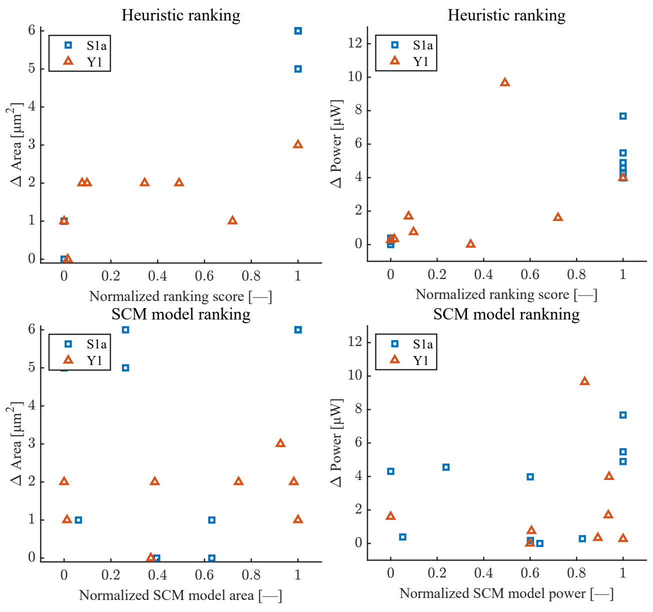


Figure 4.8 Ranking of optimal solutions to the S1a and Y1 specifications using the heuristic criteria and SCM model.

5

RTL-level Optimization

The previous chapter managed to find good coefficient set candidates, though it was hard to decide which set, out of multiple candidates with the same number of signed digits, would give better results. A better strategy for this could be to use the candidate with the best potential of optimizing the filter at the RTL level, further reducing the cost of the optimized coefficients. In this chapter one such technique, manual sub-expression sharing, is explored.

5.1 Sub-expression sharing

Section 3.2.4 showed that it is possible to outperform the synthesis tool with manual sub-expression sharing in sum of products operations. In this section, this method is generalized and applied to realistic coefficient sets. A greedy algorithm is developed that for a given coefficient set finds up to three common non-overlapping sub-expressions which can be used to implement sub-expression sharing that minimizes the number of additions needed by the sum of products (SOP) operation. Note that since the algorithm is greedy, it does not guarantee the solution to be optimal. After the sub-expressions are known, the number of adders needed for the SOP can be calculated and the optimization potential can be estimated. By applying the algorithm on a set of good candidates, these can be ranked based on the number of adders rather than the number of signed digits or the heuristic criterion of Section 4.2.2. A candidate with a suboptimal number of signed digits might for example outperform a candidate with fewer signed digits if it has good sub-expression-sharing potential that is utilized.

The algorithm is then tested and evaluated at the candidates of the specifications generated in Chapter 4. The best sub-expression solutions are implemented explicitly, synthesized and compared to the automatic synthesis results.

5.1.1 Description of the sub-expression extraction algorithm

At an overview level, the algorithm extracts many combinations of non-overlapping sub-expressions from the set of coefficients and chooses the combination with the most sub-expression occurrences, translating to the fewest adders in the implementation. The algorithm works in three stages. In the first stage, a list s_o with sub-expressions found in the coefficient set (in canonical signed digit representation) is generated. The search space of potential sub-expressions s_p is restricted to the canonical signed digit representation of odd integers between 3 and 75, i.e. $\{\pm 10\bar{1}, \pm 101, \pm 100\bar{1}, \ldots, \pm 1010\bar{1}0\bar{1}\}$. In the second stage, a second list, S_i, of pairs of non-overlapping sub-expressions in s_o that exist in at least one coefficient is produced. Finally, after the pairs of sub-expressions have been found there is a third step where all coefficients containing neither of the two previous sub-expressions are scanned for any remaining sub-expressions. The most frequently occurring sub-expression, or none if no more sub-expressions exist, is selected. Once all triplets of sub-expressions are found, the number of sub-expression occurrences can be counted and the best triplet is selected. A more detailed algorithmic description can be found in Algorithm 1.

The problem of finding sub-expressions to reduce the number of adders in SOPs could probably be solved optimally in a reasonable time, considering all possible sub-expressions and searching for more than three sub-expressions. However, for the purpose of this thesis, this algorithm was considered good enough to show whether it is possible to optimize the filter on the RTL level. Additionally, more sub-expressions requires more complex manual implementations.

5.1.2 Results

The sub-expression extraction algorithm was run on all solutions to the specifications in Table 4.3 found in Chapter 4. Manual implementations of the found sub-expressions were then synthesized at both 2 ns and 4 ns. Both the coefficient set with the fewest number of adders after optimization and the coefficient set with the optimal number of signed digits were implemented. The area and power results for these implementations, referred to as **Subexp.** and **Best SPT**, as well as the best automatic results, **Synth.**, can be found in the tables 5.1, 5.2, 5.3, 5.4 and 5.5.

For both S1a and Y1, the best coefficient sets found by Algorithm 1 also happened to be one of the solutions with the fewest signed digits. The manual sub-expression implementation were therefore chosen as another of the best signed digit solutions. The results for these specifications indicate that given two sets of coefficients with the same number of signed digits, the coefficient set with more sub-expressions will perform better when manually implemented. However, both manual implementations outperformed the behavior-level synthesis.

The best sub-expression solutions of specifications A and E both have one more signed digit than the best signed digit solution. For these specifications there is no clear trend in which solution is better as the sub-expression implementation of A perform slightly better at 2 ns but slightly worse at 4 ns, while the best signed digit solution of E outperforms the sub-expression solution at both clock periods.

Specification G is the most extreme investigated case in terms of difference in number of signed digits between the optimal signed digit solution with 15 signed digits and the best sub-expression solution with 18 signed digits. It is also the spec-

Algorithm 1 Greedy sub-expression extraction algorithm

```
• Step 1. Extract all potentially overlapping sub-expressions.
for all $h_i$ do                                              ▷ $h$: Coefficients.
    for all $s_{p,j}$ do                             ▷ $s_p$: Potential sub-expressions.
        if $s_{p,j} \in P(h_i)$ then                ▷ $P(h_i)$: Set of all subsets of $h_i$.
            $s_o \leftarrow [s_o, s_{p,j}]$      ▷ $s_o$: (Potentially) Overlapping sub-expressions.
        end if
    end for
end for

• Step 2. Form combinations of two non-overlapping sub-expressions.
for all $s_{o,i}$ do
    $S_i \leftarrow s_{o,i}$                     ▷ $S$: Non-overlapping sub-expressions-matrix.
    for all $h_j$ do
        for all $s_{o,k}$ such that $k \neq i$ do
            if $s_{o,k} \in P(h_j \setminus s_{o,i})$ then
                $S_i \leftarrow [S_i, s_{o,k}]$
                break
            end if
        end for
    end for
end for

• Step 3. Look for sub-expressions in those coefficients that do not contain current
combination of sub-expressions.
for all $S_i$ do
    for all $h_j$ such that $h_j \cap S_i = \emptyset$ do
        for all $s_{o,k} \notin S_i$ do
            if $s_{o,k} \in P(h_j)$ then
                $S_i \leftarrow [S_i, s_{o,k}]$
            end if
        end for
    end for
end for
Sort $S$, extract corresponding occurrences of sub-expressions into $N$
return $S, N$
```

Filter specification S1a

Period	Implem.	Comb. area	Rel. area	Comb. power	Rel. power	Rel. adders
2 ns	Synth.	138 µm^2	1	169 µW	1	1
	Subexp.	119 µm^2	**0.86**	146 µW	**0.86**	0.87
	Best SPT	132 µm^2	0.96	165 µW	0.98	0.90
4 ns	Synth.	107 µm^2	1	68 µW	1	1
	Subexp.	98 µm^2	**0.92**	61 µW	**0.90**	0.87
	Best SPT	102 µm^2	0.95	65 µW	0.96	0.90

Table 5.1 Table of results for different filter implementations following the specification S1a. Synth. refers to the behavior-level implementation, Subexp. refers to to the manual implementation of the coefficient set with the fewest number of adders after optimization, and Best SPT refers to a manual implementation of a coefficient set with the optimal number of signed digits.

Filter specification Y1

Period	Implem.	Comb. area	Rel. area	Comb. power	Rel. power	Rel. adders
2 ns	Synth.	181 µm^2	1	216 µW	1	1
	Subexp.	152 µm^2	**0.84**	190 µW	**0.88**	0.83
	Best SPT	164 µm^2	0.91	204 µW	0.94	0.86
4 ns	Synth.	134 µm^2	1	82 µW	1	1
	Subexp.	117 µm^2	**0.87**	74 µW	**0.90**	0.83
	Best SPT	122 µm^2	0.91	78 µW	0.95	0.86

Table 5.2 Table of results for different filter implementations following the specification Y1. The implementations are explained in the caption of Table 5.1.

Filter specification A

Period	Implem.	Comb. area	Rel. area	Comb. power	Rel. power	Rel. adders
2 ns	Synth.	229 µm^2	1	270 µW	1	1
	Subexp.	186 µm^2	**0.81**	224 µW	**0.83**	0.85
	Best SPT	191 µm^2	0.83	234 µW	0.87	0.87
4 ns	Synth.	195 µm^2	1	114 µW	1	1
	Subexp.	177 µm^2	0.91	115 µW	1.01	0.85
	Best SPT	166 µm^2	**0.85**	103 µW	**0.90**	0.87

Table 5.3 Table of results for different filter implementations following the specification A. The implementations are explained in the caption of Table 5.1.

Filter specification E

Period	Implem.	Comb. area	Rel. area	Comb. power	Rel. power	Rel. adders
2 ns	Synth.	164 µm^2	1	197 µW	1	1
	Subexp.	141 µm^2	0.86	177 µW	0.90	0.83
	Best SPT	122 µm^2	**0.74**	157 µW	**0.80**	0.87
4 ns	Synth.	124 µm^2	1	79 µW	1	1
	Subexp.	115 µm^2	0.93	72 µW	0.91	0.83
	Best SPT	92 µm^2	**0.74**	57 µW	**0.72**	0.87

Table 5.4 Table of results for different filter implementations following the specification E. The implementations are explained in the caption of Table 5.1.

Filter specification G

Period	Implem.	Comb. area	Rel. area	Comb. power	Rel. power	Rel. adders
2 ns	Synth.	95 µm^2	1	119 µW	1	1
	Subexp.	90 µm^2	**0.95**	114 µW	**0.96**	0.91
	Best SPT	90 µm^2	**0.95**	114 µW	**0.96**	0.95
4 ns	Synth.	75 µm^2	**1**	49 µW	1	1
	Subexp.	75 µm^2	**1**	46 µW	**0.94**	0.91
	Best SPT	76 µm^2	1.01	49 µW	1	0.95

Table 5.5 Table of results for different filter implementations following the specification G. The implementations are explained in the caption of Table 5.1.

ification with the lowest number of optimal signed digits and so the relative difference thus becomes even bigger. With the results of previous specifications in mind it might be expected that the best signed digit solution would outperform the best sub-expression solution. However, this is not the case. Both manual implementations perform very similarly, with a slight reduction in area and power over the behavior-level implementation at 2 ns and similar area at 4 ns. Power-wise the best sub-expression solution performs slightly better than the other two implementations at 4 ns.

The mean absolute difference between the relative area and power at 2 ns and the relative additions for all specifications were 4.3% with a standard deviation of 3.3%. Ten filters from five specifications are not enough to make general statements about the potential of using relative adders to estimate the area and power gains of manual sub-expression sharing in sets of coefficient. However, these results indicate that relative adders could prove to be a decent metric for general savings given more data. The relative-addition metric will always assume the best sub-expression solution to perform better than all other solutions and will thus never be able to predict cases such as specification E where the optimal signed digit solution significantly outperforms its best sub-expression solution.

It can be concluded that while the performance improvement varies from 28% down to 5%, additional optimization can often be performed on the RTL level to decrease area and power after the best coefficients have been selected. Almost all manual implementations were better than the automatic synthesis. Finally, for all specifications except E, the relative performance improvements of the manual implementations were higher at 2 ns than at 4 ns which is somewhat surprising, since the minimal SOP subblock example in Section 3.2.4 had the reverse relationship with a significantly higher relative gain at 4 ns. The shorter clock period should make the increased logic depth from reusing sub-expressions less beneficial. However, due to the black-box optimization of the synthesis toolchain the structure of the synthesized results are not completely known and it could be that manual implementations does not actually increase logic depth over behavior-level implementations.

6

Summary & Conclusions

This thesis has investigated different aspects of implementing power-efficient fixed-coefficient FIR filters in ASICs. Multiple strategies have been found to reduce the power consumption while adhering to a filter specification.

Investigating MCM and SCM filter blocks showed that each signed digit had a linear impact on the multiplication cost of around 3.5 μm^2 and 4.2 μW, showing significant optimization potential. For the single multiplications, the costs of all coefficients between -1024 and 1024 were measured at 2 ns and a model for estimating the cost of coefficients outside this range was developed. This model was used to compare the gain from performing MCM compared to many SCM, showing an area decrease of only 8%, which was significantly lower than the theoretical adder reduction which was 39%. The power consumption of the MCM was worse than the power consumption of many SCM. Additionally, the area and power of an adder-tree was shown to be linearly dependent on the number of summed bits, though only with 0.28 μm^2 and 0.33 μW per bit. Therefore, reducing the number of signed digits was selected to be the main optimization goal when selecting coefficients. Finally, by implementing an example of explicit sub-expression sharing in a sum-of-products block, it was shown that it is possible to outperform the automatic synthesis.

Next, the optimization of the filter design was considered. When comparing architectures, the transposed form was found to perform significantly worse than the direct form, mainly due to the almost double number of flip-flops which was not compensated by optimized combinational logic. On the contrary, both the combinational area and power was worse for the transposed form. This was partly due to the small gain from utilizing MCM as mentioned above but also due to about half of the logic consisting of the structural adders, which in the direct form can be optimized together in an adder tree.

For selecting the coefficients, a mixed-integer linear program was formulated to minimize the number of signed digits for a low-pass filter specification. For one filter specification where all possible solutions could be found, the optimal solution had 30% fewer signed digits than the median. Additionally, it was found that it is important to, within reason, give the problem flexibility. Allowing flexible gain, compared to forcing the gain to be one, decreased the optimal number of signed digits with on average 26%, and in one case made the problem feasible with fewer coefficients. Furthermore, in one case using one more coefficient than what was needed for the problem to be feasible reduced the number of signed digits with almost half. However, even with this flexibility the optimization only found one unique solution with the optimal number of signed digits for three of the five investigated specifications.

When the optimal and some of second-to-optimal solutions were synthesized, the number of signed digits was confirmed to have a linear impact of the cost, even for different filter orders ranging from 15 to 43. The combinational cost of adding coefficients can thus be considered to be negligible. Of course, more coefficients instead increases the register cost. However, this cost was comparable to the cost of a few signed digits, confirming that increasing the number of coefficients can be worth it if the number of signed digits decreases.

When comparing synthesis results at different clock periods, the additional optimization possible at 4 nanoseconds compared to 2 nanoseconds increased the frequency adjusted area with 56% while the adjusted power was decreased by 19%. This suggests that a parallelized filter structure running at half clock frequency could be more power efficient, at the cost of requiring more area.

For specifications that had multiple optimal solutions, a secondary heuristic criterion was developed to select the best solution from these candidates. This criterion was compared to a model based on data from earlier synthesized single multiplications. While the model did not manage to rank the solutions, the criterion showed some potential but ultimately, there was not enough data to draw a conclusion.

Instead, it seems promising to base the choice of the best candidate on its sub-expression-sharing potential. In the final part of the thesis, potential improvements at the RTL level were investigated to further optimize the best candidates. Most importantly, the potential for sub-expression sharing was investigated and an algorithm was developed for reducing the number of adders in the SOP sub-block. Good candidates from this algorithm were then hand-implemented and compared to the behaviour-level implementation. For each specification two solutions were manually implemented with sub-expressions: the best sub-expression solution given by the algorithm, and the optimal signed digit solution. The area and power gains of these manually implemented solutions varied between 5% and 28%. However, only one of the specifications with a subtoptimal signed digit solution as its best sub-expression solution had that solution perform better than the optimal signed digit solution with manually implemented sub-expressions. In conclusion, explicit sub-expression sharing can decrease area and power but it is generally not worth implementing suboptimal solutions over optimal solutions despite their greater sub-expression-sharing potential.

6.1 Method critique

The empirical results in this thesis are based on a specific synthesis environment with internal black-box optimization, which limits the level of control for optimization and makes it hard to determine the exact cause and effect of certain results. This both affects the general applicability of the empirical results and introduces noise in the results. Specifically, the selection of slower, energy efficient cells versus fast, leakier cells was found to be hard to control while potentially impacting the results drastically. Because of this, the finer optimizations such as the secondary criterion and manual sub-expression sharing should be applied carefully.

Finally, to be able to produce results that take the logical depth into account, the power estimation needs to take glitching into account which was not possible in this thesis. Because of this, the area and power results for a specific IP were always linearly dependent when comparing parameters for the same IP. The power estimation was therefore only interesting when comparing two different IPs or two different clock periods.

6.2 Future work

Assuming a straight-forward filter implementation with no pipelining registers, the developed strategy for selecting coefficients is probably close to ideal, at least within the noise introduced by the black-box optimization. However, as the development of digital-analog technology continues, the clock frequency requirements of radio ASICs can be expected to increase, further increasing the impact of the timing constraint. Future optimization efforts therefore need to put focus on pipelining and possibly architectures for parallelizing the filter.

Furthermore, there are known optimization methods not discussed in this thesis allow approximations of the filter output. For example, applying internal truncation before an adder tree would both reduce the size and power as shown in the thesis as well as improve timing, since fewer bits need to be computed. However, an investigation into the potential gain of this technique, as well as the characteristics of the noise, is needed.

Bibliography

Aksoy, L., E. Costa, P. Flores, and J. Monteiro (2011). “Design of low-power multiple constant multiplications using low-complexity minimum depth operations.” *Proceedings of the 21st Edition of the Great Lakes Symposium: Great Lakes Symposium on VLSI*, pp. 79–84.

Aksoy, L., P. Flores, and J. Monteiro (2014). “A tutorial on multiplierless design of FIR filters: algorithms and architectures.” *Circuits, Systems, and Signal Processing* **33**:6, pp. 1689–1719.

Aktan, M., A. Yurdakul, and G. Dundar (2008). “An algorithm for the design of low-power hardware-efficient FIR filters.” *IEEE Transactions on Circuits and Systems I: Regular Papers* **55**:6, pp. 1536–1545.

Chung, J.-G. and K. K. Parhi (2002). “Frequency spectrum based low-area low-power parallel FIR filter design.” *EURASIP Journal on Applied Signal Processing* **2002**:3, pp. 944–953.

Coward, S., G. A. Constantinides, and T. Drane (2022). “Automatic datapath optimization using e-graphs.” *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH)*, pp. 43–50.

Demirsoy, S., A. Dempster, and I. Kale (2002). “Power analysis of multiplier blocks.” *2002 IEEE International Symposium on Circuits and Systems (ISCAS)* **1**.

Dong, S. and Y. Yu (2011). “Design of linear phase FIR filters with high probability of achieving minimum number of adders.” *IEEE Transactions on Circuits and Systems I: Regular Papers* **58**:1, pp. 126–136.

Eriksson, H. and P. Larsson-Edefors (2004). “Glitch-conscious low-power design of arithmetic circuits.” *2004 IEEE International Symposium on Circuits and Systems (ISCAS)* **2**.

Garcia, R., A. Volkova, and M. Kumm (2022). “Truncated multiple constant multiplication with minimal number of full adders.” *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 263–267.

Gustafsson, O., H. Ohlsson, and L. Wanhammar (2004). "Low-complexity constant coefficient matrix multiplication using a minimum spanning tree approach." *Proceedings of the 6th Nordic Signal Processing Symposium (NORSIG)*, pp. 141–144.

Gustafsson, O., H. Johansson, and L. Wanhammar (2001). *An MILP approach for the design of linear-phase FIR filters with minimum number of signed-power-of-two terms.* LiTH-ISY-R: 2390. Linköping University.

Horrocks, D. H. and Y. Wongsuwan (1999). "Reduced complexity primitive operator FIR filters for low power dissipation". In: *Proc. European Conference Circuit Theory and Design*, pp. 273–276.

Kumm, M., M. Hardieck, and P. Zipf (2017). "Optimization of constant matrix multiplication with low power and high throughput." *IEEE Transactions on Computers* **66**:12, pp. 2072–2080.

Kumm, M., A. Volkova, and S. Filip (2023). "Design of optimal multiplierless FIR filters with minimal number of adders." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **42**:2, pp. 658–671.

Lim, Y. and S. Parker (1983). "FIR filter design over a discrete powers-of-two coefficient space". *IEEE Transactions on Acoustics, Speech, and Signal Processing* **31**:3, pp. 583–591.

Lu, W.-S. (2001). "Design of FIR filters with discrete coefficients: a semidefinite programming relaxation approach." *2001 IEEE International Symposium on Circuits and Systems (ISCAS)* **2**, p. 297.

Månsson, J. (2021). *Adder Minimization and Retiming in Parallel FIR-Filters*. MSc thesis. Linköping University.

Shahein, A. (2014). *Power Optimization Methodologies for Digital FIR Decimation Filters*. PhD thesis. University of Freiburg.

Shahein, A., Q. Zhang, N. Lotze, and Y. Manoli (2012). "A novel hybrid monotonic local search algorithm for FIR filter coefficients optimization." *IEEE Transactions on Circuits and Systems I: Regular Papers* **59**:3, pp. 616–627.

Voronenko, Y. and M. Püschel (2007). "Multiplierless multiple constant multiplication." *ACM Transactions on Algorithms* **3**:2.

Ye, W. B., X. Lou, and Y. J. Yu (2017). "Design of low-power multiplierless linear-phase FIR filters." *IEEE Access* **5**, pp. 23466–23472.

Yu, Y. J. and L. Yong Ching (2007). "Design of linear phase FIR filters in subexpression space using mixed integer linear programming." *IEEE Transactions on Circuits and Systems I: Regular Papers* **54**:10, pp. 2330–2338.

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER'S THESIS *Date of issue* June 2023 *Document Number* TFRT-6199
Author(s) Erik Lundell Gustav Molin	*Supervisor* Olof Troeng, Ericsson, Sweden Bo Bernhardsson, Dept. of Automatic Control, Lund University, Sweden Emma Tegling, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Energy-Efficient Fixed-Coefficient FIR Filters for Millimeter-Wave Radios

Abstract

With the introduction of millimeter-wave antenna arrays in 5G base-stations and ever-increasing data volumes, the power consumption of the signal processing in digital radio systems has increased over the last decade. This, combined with increased cost and environmental awareness, has put focus on power optimization. This thesis investigates different aspects of optimizing one important component of digital radio systems, fixed-coefficient FIR filters.

The thesis initially investigates subblocks of the FIR filters separately. The cost of fixed coefficient multiplication is found to be linearly dependent on the number of signed digits in the coefficient set while additions are linearly dependent on the number of summed bits.

The second part of the thesis considers complete filters. First, two common filter architectures are compared and the direct form is found to be more efficient. Then, a mixed-integer linear program is formulated that minimizes the number of signed digits in the filter coefficients. This optimization is shown to be able to reduce the number of signed digits with 20–30% compared to the median which translates to a similar reduction in area and power. Furthermore, allowing the filter gain to be flexible is found to reduce the optimal number signed digits additionally with around 20%. For filter specifications where there are multiple solutions with the same optimal number of signed digits, a ranking score based on other coefficient features is suggested.

In the final part of the thesis, the optimal solutions from the second part are optimized further. An algorithm for finding common sub-expressions in coefficient sets is developed. Using these sub-expressions when implementing the filter is shown to reduce both area and power consumption with 5–25%.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-58	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/