Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Andreas Bjørsvik
Sevat Mestvedthagen

Optimization of CBADC Digital Estimation Filter for RISC-V Implementations

Master's thesis in Electronic Systems Design
Supervisor: Trond Ytterdal
Co-supervisor: Fredrik Esp Feyling
June 2023

NTNU
Norwegian University of
Science and Technology

Andreas Bjørsvik
Sevat Mestvedthagen

Optimization of CBADC Digital Estimation Filter for RISC-V Implementations

Master's thesis in Electronic Systems Design
Supervisor: Trond Ytterdal
Co-supervisor: Fredrik Esp Feyling
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

Control-bounded analog-to-digital (A/D) conversion has emerged as a promising conversion method allowing less constrained analog and digital circuit architectures. However, a digital post-processing step is needed. In this thesis, we investigate how the digital estimation filter of the control-bounded A/D converter (CBADC) can be implemented as an accelerator complementing a 32-bit RISC-V Central Processing Unit (CPU). The implementation is a fixed-point unit made for a single input system.

As the digital estimation filter for the CBADC is not thoroughly investigated, we find what configurations of filter length, the number of fixed-point bits, and oversampling ratio are needed for each number of analog states to reach the desired signal-to-noise ratio (SNR) of 70 dB.

The developed accelerator employs a finite impulse response (FIR) filter algorithm with lookback and lookahead recursion to calculate the estimates. The accelerator parallelizes and pipelines the task to achieve the target sampling frequency of 20 MHz. The most energy-efficient configurations are the ones with the lowest number of analog states, as they also have the shortest filter lengths. This compensates for the higher oversampling ratio they need.

To reduce power consumption, multiple variations of the accelerator, aiming to optimize the circuitry, is implemented. The area and power consumption of the reference version of the accelerator with four analog states is estimated to be 170 255 µm^2 and 11.70 mW. By reducing bit widths of coefficient registers and corresponding logic units where larger bit widths are unnecessary, the power consumption can be reduced by up to 27.4%, and area reduced up to 30.7%, to 125 300 µm^2 and 9.17 mW. The implementation of lookup tables (LUTs) enables a power reduction of up to 40.1% at the expense of an area increase of up to 61.9%, thus 279 102 µm^2 and 7.44 mW. Among the variations of LUT implementations, the two-input LUT proves to be the most power efficient. A commercially available 28 nm CMOS technology is used for all the simulations.

Sammendrag

Kontrollbegrenset analog-til-digital (A/D) omforming har vokst frem som en lovende omformingsmetode som tillater færre begrensninger på de analoge og digital kretsarkitekturene. Derimot er et digitalt etterprossesseringssteg nødvendig. I denne avhandlingen vil vi undersøke hvordan det digitale estimeringsfilteret til en kontrollbegrenset A/D omformer (CBADC) kan bli implementert som en akselerator tilkoblet en 32-bit RISC-V prosessor (CPU). Implementasjonen er en fasttallsenhet laget for et system med et enkeltinngangssignal.

Ettersom det digitale esimeringsfilteret for CBADCen ikke er grundig undersøkt, har vi funnet konfigurasjonene av filterlengde, antall fasttallbits, oversamplingsrate som er nødvendig for hvert antall analoge tilstander for å nå ønsket signal-støy forhold (SNR) på 70 dB.

Den utviklede akselleratoren benytter en avgrenset impulsrespons (FIR) filter algoritme med rekursjoner fremover og bakover for å regne ut estimatene. Akselleratoren paralelliserer oppgaven for å nå målet om en samplingfrekvens på 20 MHz. De mest energieffektive konfigurasjonene viser seg å være de med lavest antall analoge tilstander siden disse også har de korteste filterlengdene. Dette kompenserer for de høye oversamplingsratene de trenger.

For å redusere effektforbruket har flere variasjoner av akselleratoren, med mål om å optimalisere kretsen, blitt implementert. Arealet og effektforbruket til referanseversjonen av akselleratoren med fire analoge tilstander er estimert til å være 170 255 µm^2 og 11.70 mW. Ved å redusere bitbredden for koeffisientregistre og samsvarende logiske enhter der større bitbredde er unødvendig, kan effektforbruket reduseres med opptil 27.4% og arealet reduseres med opptil 30.7%, til 125 300 µm^2 og 9.17 mW. Ved å implementere oppslagstabeller (LUTer) muliggjør en reduksjon av effektforbruket på opptil 40.1% på bekostning av en økning av areal på opptil 61.9%, til 279 102 µm^2 og 7.44 mW. Blant variasjonene av LUT-implementasjoner, ble en to-inngangs LUT funnet mest effektsparende. En kommersielt tilgjengelig 28 nm CMOS teknologi er brukt i simuleringene.

Acknowledgements

Firstly, we would like to thank our supervisors, Professor Trond Ytterdal and Fredrik Esp Feyling, for their guidance over the past year. The meetings we have had throughout the year have been inspiring and enlightening.

We would also like to thank Leon Mayrhofer for our conversations regarding the implementation of the digital estimation filter and the comparison of results.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the world we live in is analog, all digital systems collecting data from the physical world need to convert those analog signals into digital representations. The conversion is achieved using an analog-to-digital converter (ADC).

The analog signals are often weak, and the translation to the digital world consumes energy. By utilizing a vast amount of resources on an ADC, the results can be precise. The power consumption, however, will also be high. In many applications, this energy consumption is critical and should be minimized.

There exist ADCs in a variety of versions, but all ADCs perform both analog signal processing and digital signal processing. The amount of each of these steps can vary, but if either the analog or the digital signal processing is inaccurate, the ADC will not be precise. Finding the optimal weighting of resources between analog and digital processing is important to reduce power consumption. Research shows that by combining a complex analog system with multiple simple digital controls, precise analog-to-digital (A/D) conversion can be accomplished [2]. This converter is called a control-bounded analog-to-digital converter (CBADC). The digital control signal is analyzed by a digital estimator to find and compute the digital representation of the analog signal.

1.1 Control-Bounded Conversion

The CBADC is a new and innovative method for A/D conversion first proposed by Loeliger et al. in [5]. Later these ideas have been further developed [2, 6, 7]. The main difference between CBADC and A/D converters based on sampling theory is that CBADC separates the analog and digital parts, whereas the digital control is used to stabilize the analog system. The separation of these two parts also makes room for new innovations in both the analog and digital parts.

One problem the CBADC has a solution for is the loss of performance from imperfect components and the impact of real-world environments. A calibration based on the transfer function of the analog system will synchronize the digital estimator with the analog system. The calibration will maintain the performance of the system even if the characteristics of the analog system are changed by real-world impact.

1.2 Related Work

In [5] Loeliger et al. introduced a new type of ADCs called unstable-filter ADC. This concept was developed further in [6]. The first time the control bounded A/D converter was mentioned was in the dissertation of [2]. At the date of writing, the material made by Malmberg is the main source on this topic. How the analog part of the system can be realized has been discussed by Esp Feyling

in [8].

The work presented in this thesis is based on a previous project by the authors exploring design implementations of two different digital estimation filters for the CBADC [9] on a RISC-V processor. These implementations were a single-core finite impulse response (FIR) filter and a multi-core batch filter. Both filters performed poorly, but the FIR filter showed more promising qualities in terms of potential speedup. The batch filter is thus deserted, but the goal of this project is somewhat similar. In addition to this, another study of digital estimation filter architectures has been conducted, where a hybrid estimation filter also was implemented [10].

1.3 Scope

The thesis aims to give insight into how power consumption and the area of the digital estimation filter of a control-bounded A/D converter are affected by design choices. The thesis discusses how the implemented filter using a RISC-V CPU and an accelerator performs with variations in hardcoded parameters and post-synthesis parameterizability. The thesis explains how parameters can be changed to optimize power consumption while maintaining a high signal-to-noise ratio (SNR). The digital implementation filter is made for a single-input A/D converter.

The digital estimation filter is implemented in multiple different versions, some with optimizations, to find how optimization techniques can reduce area or power. The thesis discusses how power consumption and area are affected by enabling post-synthesis parameterization, reduction of bit widths, and change of filter algorithm.

The designs will be synthesized on a 28 nm FDSOI technology, with a supply voltage of 0.80 V. Power estimates are generated using Synopsys PrimeTime. PrimeTime uses the design's synthesized netlist and switching activity from simulation and information from the cell library. The CPU core is an open-source CPU core, the Picorv32 [4], configured with the extension RV32E. The accelerator is designed using SystemVerilog.

The focus will be on the accelerator and the CPU. In terms of power consumption, hierarchy, or delay, the memory system is not considered. The analog system of the CBADC, its calibration, and how it interacts with the memory are not considered.

1.4 Main Contributions

The main contribution of this thesis is the insight into how the digital estimator part of the CBADC can be implemented and optimized when using a CPU and a connected accelerator. Some other contributions are:

- The area and power of the digital estimator are highly correlated with the number of analog states used and the filter depth. The digital estimator will be smaller and consume less power if fewer analog states are used. The area will also be reduced when the filter length is lowered.
- By pre-calculating some filter coefficients of the digital estimator and saving them as lookup tables, power can be reduced at the cost of a larger area
- Both power and area can be reduced by reducing the bit width of registers for the smaller coefficients and the size of modules computing using these coefficients.
- By making the circuit for the digital estimator configurable after synthesis, the area and power consumption are unnoticeably higher.

1.5 Specifications

The target bandwidth of the system is set to be 10 MHz. The Nyquist frequency is thus 20 MHz, and the minimum sampling frequency is 20 MHz. The system should have at least 70 dB SNR, and the digital estimation filter shall not be the limiting factor. The design should be parameterizable and should be implemented to work for designs with three to eight analog states. The specifications are displayed in Table 1.1.

Table 1.1: Target specifications.

Sampling frequency	$\geq$ 20 MHz
Bandwidth	10 MHz
Analog states	3 - 8
SNR	$>$ 70 dB

1.6 Outline

The thesis starts by presenting the theory needed to understand the rest of the thesis. Chapter 2 explains different principles on how to optimize circuits for power, like clock gating, changing supply voltage, and pipelinging. In the end, details about how the control-bounded A/D converter work and details on the digital estimator are presented.

Chapter 3 explains how the digital estimation filter is implemented by first describing the CPU core and accelerator module. Then different optimizations implemented in variations of the accelerator are explained. At the end of the section, the software tools used are mentioned, and a description of how they work is provided.

The next part of the thesis presents the results from the simulations and discusses how different oversampling ratio (OSR), number of analog states N, and filter length K will impact the performance. Chapter 4 starts by presenting how the system is configured to get the target SNR and frequency and what limitations this gives. The rest of the section is looking at the performance of the different versions of the accelerator and how their optimizations reduce the power consumption of the circuitry.

In Chapter 5, a discussion of possible optimizations to reduce the size, power consumption, or increase sampling frequency is provided. The section presents possible solutions that are generally known principles on how to optimize the power of the circuit and, more specifically, how the DE of the CBADC can be optimized.

Chapter 6 concludes the thesis.

Chapter 2

Background

2.1 Number Representation

The most common ways to represent decimal numbers in computer systems are floating-point and fixed-point representation. The main difference is that in fixed-point representation, a fixed number of bits are used to represent the integer and fraction part of the number. Whereas floating point numbers vary the precision depending on the use. Because floating-point numbers use variable numbers of bits to represent the number, they have a higher dynamic range and precision than the fixed-point representation. A drawback of floating-point representation is increased complexity, resulting in larger and often slower arithmetic logic units to do floating-point operations [11]. Recent studies have shown that utilizing fixed-point representation instead of floating-point in the MPEG-2 video compression algorithm results in a significant decrease in cycle count (75%) and an increase in energy efficiency (76%) [12].

The bits in a fixed-point number are divided into three parts: the sign bit, the integer part, and the fraction part. The organization of the bits can be seen in Table 2.1. The sign bit's value determines whether the number is negative or positive. The integer part is represented by bits that decide the integer value of the number, with the largest value it can represent being $(2^n - 1)$, where n is the number of bits used in the integer part. For negative numbers, the lowest number that the integer part can represent is -2^n. The remaining bits after the integer part are called the fractional bits, representing the decimal value of the number. The smallest interval between two adjacent values is determined by 2^{-f}, where f represents the number of bits allocated to the fractional part. This determines the resolution limit for numerical representation and indicates the minimum step size between representable numbers. Therefore, the numerical range that can be represented is limited by $(2^n - 2^{-f})$ and -2^n, with the step size between adjacent numbers being 2^{-f}.

Table 2.1: Organization of bits with fixed-point representation.

Sign bit	Interger part	Fraction part
1-bit	n-bits	f-bits

When changing how many bits are used to store a number, sign extending needs to be done to not change the value. This means copying the most significant bit to all the newly added bits. For example, if you have a 4-bit number and want to store it in an 8-bit register, you can store the number 0110 (6) as 00000110. However, if it is a negative number, such as 1101 (-3), it is stored as 11111101. If sign extension is not done, the information that indicates whether the number is negative or positive will be lost since the leftmost bit is used to determine that.

2.2 Oversampling and Subsampling

Oversampling is a technique commonly used in ADCs to improve the SNR of the resulting digital signal. The method involves sampling the analog input signal at a frequency that exceeds the Nyquist frequency. The degree of oversampling is measured by the oversampling ratio (OSR) and is calculated as

$$\text{OSR} = \frac{F_s}{F_N} = \frac{F_s}{2BW} \tag{2.1}$$

where F_s is the sampling frequency, F_N is the Nyquist frequency and BW is the bandwidth of the signal. By increasing the sampling frequency above the Nyquist rate, oversampling provides additional information about the analog signal, thereby reducing the effects of noise and increasing the SNR of the ADC[13].

To mitigate this computational burden, downsampling can be applied to the oversampled signal. Downsampling reduces the number of calculated samples to $1/DSR$ of the original signal. If the oversampling ratio is equal to the downsampling ratio $DSR = OSR$, the resulting sampling frequency of the ADC would be equal to the Nyquist frequency F_n. However, if DSR is greater than OSR, aliasing and information loss can occur, leading to degraded signal quality and lower SNR. The maximum DSR can be without aliasing can be given by

$$\text{DSR} = \frac{F_s}{F_N}. \tag{2.2}$$

2.3 Power Consumption in a Digital Circuit

The power P in an electrical circuit is given by

$$P(t) = I(t)V(t), \tag{2.3}$$

where I(t) and V(t) are the total current and voltage in the circuit at time t. The energy consumed by the circuit can be found by integrating the power over a time period T:

$$E = \int_{t=t_0}^{t_0+T} P(t)dt. \tag{2.4}$$

In an ideal circuit, only the charging of the capacitances in the circuit would consume power. The energy to charge one capacitor can be found using Equation 2.4. By charging a capacitor with capacitance C to the voltage V_C the energy, E_C, consumed is

$$E_C = \int_0^\infty I(t)V(t)dt = \int_0^\infty C\frac{dV}{dt}V(t)dt = C\int_0^{V_{DD}} V(t)dv = \frac{1}{2}CV_{DD}^2. \tag{2.5}$$

In a digital circuit, the two contributors to power consumption can be divided into static power and dynamic power. The total power consumed by the circuit is then

$$P_{tot} = P_{dynamic} + P_{static}. \tag{2.6}$$

The static power of the circuit is leakage power that is dissipated from the circuit when a voltage is applied [1]. The static power is primarily dependent on the voltage V_{DD}, technology node, and area of the circuit. When measuring leakage, the currents can be divided into several sources of leakage currents, denoted as Isub, Igate, Ijunct, and Icont. These sources respectively represent the subthreshold current that flows through turned-off transistors, the current that leaks through the thin gate dielectric, the current that arises from diffusions between the source and drain, and the current due to contention in ratioed circuits.

The total static power consumed by the circuit is given by the product of the supply voltage VDD and the sum of these individual current components, as shown here

$$P_{static} = (I_{sub} + I_{gate} + I_{junct} + I_{cont}) \cdot V_{DD}. \tag{2.7}$$

Therefore, minimizing the individual components of static power is crucial in designing low-power digital circuits that optimize power consumption and performance.

Dynamic power, however, is the power consumed by the circuit due to the transistors switching between states. Some dynamic power is needed to charge load capacitances, but some currents are unwanted. Short-circuit currents are not wanted and are caused when a complementary pMOS and nMOS transistor are both partially turned on during switching, leading to a current flow from V_{DD} to V_{SS}.

Modern complementary metal-oxide-semiconductors (CMOS) chips consist of large amounts of transistors. The number of transistor transitions in the circuit each clock period can be described by the activity factor α. The activity factor is the probability of a circuit node transitioning from 0 to 1, as this is the only time a node consumes active power. The power consumed can then be written as

$$P_{switching} = \alpha C V_{DD}^2 f, \tag{2.8}$$

where f is the clock frequency. To reduce the power consumption of the circuit, either of these coefficients must be reduced. Because the voltage is squared, a change in this coefficient would have the largest impact on the power.

2.4 Supply Voltage Level and Device Threshold Voltage

The power consumed by the circuit is highly dependent on the supply voltage V_{DD}, as seen in Equation 2.7 and Equation 2.8. A reduction would profoundly affect the total power consumption of the circuit, as the relation is linear for the static power dissipation and quadratic for the dynamic power dissipation, even when considering modifications to the microarchitecture to maintain the performance of the circuitry.

Although having a low supply voltage is a crucial part of a low-power system, the supply voltage has limitations on how low it can be adjusted. As the supply voltage is reduced, the circuit's performance is also reduced. When the supply voltage approaches the device threshold voltage, the delay of signals through logic gates increases immensely [14]. The device threshold voltage thus limits the lowering of the supply voltage. However, reducing the threshold voltage simultaneously can maintain performance when reducing the supply voltage. This would drastically decrease the switching power of Equation 2.8. Sadly, the subthreshold leakage current will increase exponentially, leading to a higher static power consumption [14].

2.5 Power Gating and Clock Gating

One method of reducing the circuit's power consumption is by power gating unused parts of the chip. By cutting off power to a logic block that is not used, both the leakage power and the switching power will be removed. From Equation 2.3 and either a voltage or current is needed to draw power, and both of them will be very close to 0 if the voltage source is cut off.

When power gating, the logic block uses a virtual V_{DD} rail called V_{DDV}. Between V_{DD} and V_{DDV}, a set of header switch transistors can be turned on to raise V_{DDV} to V_{DD}. If the transistors are turned off, the connection between V_{DD} and V_{DDV} disappears, and V_{DD} will gradually fall towards ground.

The time to change between *active* and *sleep*-mode takes time and energy. In addition, some logic must be added to the chip to control the header switches. Because power gating requires extra logic, which consumes power during transitions, the size of the gated part of the logic should be significant and not be turned on and off often. When power gating, all volatile memory components of the power gated logic will lose their information. To store information needed later, state retention registers can be used at a low power cost, or the states can be stored in the memory to power gate the entire logic block. If the output from the block needs to be valid while the block is powered

down, an output isolation gate can be used.

Another effective way to reduce power consumption is clock gating. The coefficient will not reduce the static power but aims to reduce the activity factor, α, of Equation 2.8. Clock gating can be implemented by ANDing the clock signal with an enable signal to stop the propagation of the clock in idle logic blocks. This is shown in Figure 2.1. The enable latch is added to make the clock stable when the clock is active. The clock signal significantly impacts a circuit's power consumption, leaving clock gating as an effective method to reduce power consumption. As well as reducing power consumption of the clock tree, clock gating the input register to a logic block will prevent switching inside the block. The clock enable signal can be implemented with simple logic but can often be part of some of the most critical paths on the chip. The clock signal must remain stable while the clock is active to ensure that the datapath is not being corrupted. The clock gating can be fine-tuned, but turning off the clock for larger blocks can reduce the power consumption even more because the clock is cut off higher in the clock tree.

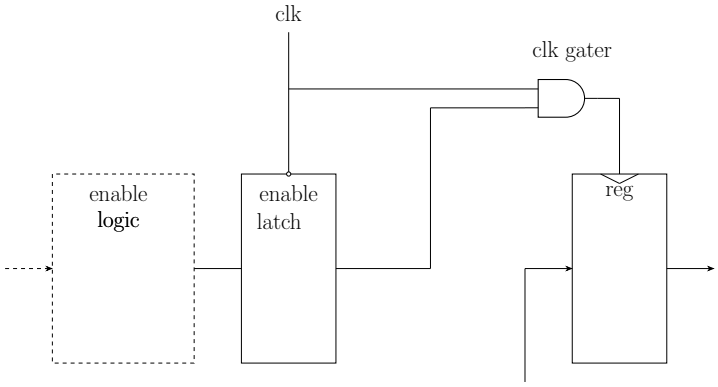


Figure 2.1: Clock gating circuit [1].

2.6 Pipelining

Replacing one large functional unit with several simpler units placed in a pipeline with registers in between reduces the critical path of the system. The result of each pipeline stage is fed into the next stage at the following clock cycle. This could allow an increase in frequency and higher computational speeds. Another possibility is to keep the low frequency and lower the voltage to reduce the power consumption. The cost of adding pipeline stages is an increase in registers and higher latencies. The pipeline stages can be hard to balance correctly, as each step should take approximately the same time to compute. With an unbalanced pipeline, the faster stages are not appropriately exploited, as the clock frequency must be low enough for the slowest stages to finish [1].

2.7 Parallelism

Instead of or in addition to pipelining, parallelism can be implemented to reduce power or increase the performance of a task. As with pipelining, parallelism computes several tasks at the same time. However, with parallelism, the computational blocks are identical. If one task is repeatedly performed, parallelism can be used to perform multiple of these tasks simultaneously. This could either be used to increase performance or, by lowering the clock frequency and supply voltage, reduce the power consumption. Some disadvantage of parallelizing a task is a significant increase in area, as well as more complex control logic. Using a parallelized structure will only provide a significant speedup if the data being processed is not highly dependent on intermediate answers from other calculations. If data is highly dependent, some processing units must wait for others before starting their own calculation.

A method of implementing a parallel architecture is multiprocessing, either by using a SIMD (Single Input Multiple Data) processor or MIMD (Multiple Input Multiple Data) processor. The SIMD processor performs the same instruction on multiple data elements using multiple PEs (Processing Elements), requiring only a single instruction fetch and instruction decode unit. A MIMD processor, on the other hand, has multiple PEs, each having its own instruction fetch and instruction decode unit. The MIMD processor is more flexible but requires a larger area and thus consumes more power.

2.8 Accelerators and Coprocessors

An accelerator is a special-purpose function unit that can offload the processor on frequently used and compute-intensive tasks. A CPU is very flexible and easy to program. However, for some tasks, they can not provide the performance, latency, or power efficiency that is required. An accelerator can not replace the CPU, but by offloading some tasks from the CPU to the accelerator, the requirements for the CPU can be relaxed [15]. Increasing the performance of a CPU might be very costly, both in terms of area and development cost, and using an accelerator might be more cost-effective. An accelerator is designed to do a specific or small set of tasks to meet system requirements at high performance, lower cost, and reduced power consumption. A single chip can consist of multiple accelerators that increase the performance. The accelerator works independently of the CPU and has its own memory.

A coprocessor has some similarities to an accelerator, as it is a specialized hardware unit designed to improve the performance of a computer system. The main difference is that while an accelerator works independently of a CPU, the coprocessor works in conjunction with the CPU. The coprocessor works alongside the CPU and shares the same workload. The coprocessor will share the same memory as the CPU and will often be used for multiple different tasks.

In [16], the power of RISC instructions in a CPU is measured. They found that for a 32-bit addition in a 90 nm RISC processor that consumes 125 pJ, only seven pJ is the ALU operation. This means that the vast majority of the power consumption comes from overhead, like fetching the instructions, decoding, and scheduling them. Using an accelerator or a coprocessor on some tasks can drastically reduce the amount of overhead and increase performance, as well as lower power consumption.

Hardware accelerators take advantage of not being constrained by the design rules that apply to general-purpose hardware, as they only will be used for one or a small set of tasks. Design considerations not improving the general case might speed up the accelerated task considerably [15]. Accelerators can take advantage of the concepts of pipelining and parallelism described in Section 2.6 and Section 2.7. By utilizing both parallelism and pipelining, performance can be significantly increased without increasing power consumption for some tasks.

2.9 RISC-V

RISC-V is an open-source instruction set architecture (ISA) first introduced in 2010 [17]. It was introduced as an alternative to the other main ISAs, such as the ARM ISA and x86 used by Intel. RISC-V uses the same principles as the ARM ISA architecture, which is also a reduced instruction set architecture (RISC) and therefore has few and simple instructions, compared to a complex instruction set architecture (CISC) such as Intels x86.

RISC-V is an ISA that tries to fit in all kinds of applications. The way that is solved is by modularity. The base ISA is simple, and if the system needs more complex instructions, they can be added with extensions. The ISA is also supported on 32, 64, and 128-bit processors. The flexibility makes it suitable for small embedded systems that need to be efficient. Another advantage of the RISC-V instruction set is its large number of "free" opcodes reserved for adding custom instructions. Using an open-source ISA like RISC-V gives the opportunity of designing a

new core, buying one from one of the several companies selling RISC-V IP blocks, or downloading a free core [18].

To make a power optimized system, the E extension can be used. It will reduce the number of CPU registers used from 32 to 16. This would reduce the area of the processor, but it could reduce the performance of the system. Some other normal extensions are the F extension for floating-point operations and the M extension for multiplication and division.

2.10 CBADC

Control-bounded analog-to-digital (A/D) conversion defines a new interface between the analog and digital worlds that utilize the strengths of both parts. This is achieved by having an unstable analog system controlled by a single or multiple digital control loops. The main contributor to the research on this topic is Malmberg in [2], which will be the primary source for this subchapter.

The control-bounded analog-to-digital converter (CBADC) breaks down the conversion task into three parts: analog system (AS), digital control (DC), and digital estimator (DE). The interaction between these steps is shown in Figure 2.2, where the input signal $u(t)$ and estimate $\hat{u}(t)$ are assumed to be scalar values. The other signals, $\tilde{\boldsymbol{s}}(t)$, $\boldsymbol{s}[k]$ and $\boldsymbol{s}(t)$, are vector-valued. The CBADC can be implemented as an A/D converter with multiple input channels to increase overall performance [2].

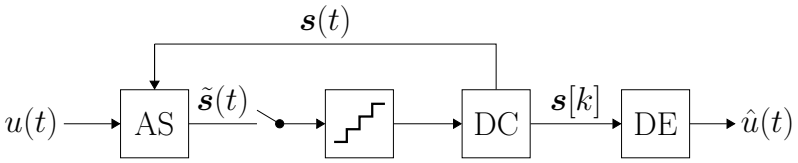


Figure 2.2: The parts of the control-bounded A/D converter [2].

2.10.1 Analog System

The AS is an analog continuous-time filter that enhances specific signal attributes while suppressing unwanted ones, typically by amplifying certain frequency bands and suppressing others. The complexity of the AS depends on the degree of amplification and sharpness of transitions needed. The performance of the control-bounded converter is closely linked to the AS's ability to amplify desired signal characteristics. The AS operates as an open-loop system controlled by the DC loop, meaning that stability of the AS is not necessary as it is enforced via the DC loop.

2.10.2 Digital Control

The DC's task is to maintain bounded AS states by observing a sampled and quantized version of the control signal $\boldsymbol{s}[k]$ and producing a control contribution in response. The DC uses both analog-to-digital converters (ADCs) and digital-to-analog converters to create a continuous-time analog signal. In order to be deemed effective, the DC must maintain a bounded AS state for an input signal that is also bounded.

2.10.3 Digital Estimator

The main task of the DE is to compute an estimate $\hat{u}(t)$ of $u(t)$. It uses the control signal $\boldsymbol{s}[k]$ created by the DC and its knowledge of the AS and control contribution, $\boldsymbol{s}(t)$, to generate the estimate. In [2] Malmberg uses a non-standard Kalman smoothing algorithm to calculate the samples in the DE. The algorithm is based on factor graphs, but it can be applied without extensive

knowledge of this topic, as it can be reduced to a linear filter. The estimation algorithm is reduced to three steps. The first is forward recursion

$$\overrightarrow{\boldsymbol{m}}_{k+1} \triangleq \boldsymbol{A}_f \overrightarrow{\boldsymbol{m}}_k + \boldsymbol{B}_f \boldsymbol{s}[k], \tag{2.9}$$

then backward recursion

$$\overleftarrow{\boldsymbol{m}}_{k-1} \triangleq \boldsymbol{A}_b \overleftarrow{\boldsymbol{m}}_k + \boldsymbol{B}_b \boldsymbol{s}[k-1], \tag{2.10}$$

and finally, the calculation of the estimates

$$\hat{u}(t_k) \triangleq \boldsymbol{W}^{\mathsf{T}} (\overleftarrow{\boldsymbol{m}}_k - \overrightarrow{\boldsymbol{m}}_k). \tag{2.11}$$

2.10.4 FIR Filter

The digital estimator part of the CBADC can be implemented using an FIR filter. When calculating the FIR estimate, Equation 2.9, 2.10, and 2.11 are used to derive the formula. The FIR filter estimate is calculated as

$$\hat{u}(t_k) = \sum_{l1=0}^{K1} \overleftarrow{\boldsymbol{h}}_{l1} \boldsymbol{s}[k+l1] + \sum_{l2=1}^{K2} \overrightarrow{\boldsymbol{h}}_{l2} \boldsymbol{s}[k-l2] \tag{2.12}$$

when the samples are uniformly spaced. In Equation 2.12 the first part is calculating the backward recursion with *K1* samples and the last summation is calculating the forward recursion with *K2* samples. To simplify further, the forward and backward recursion can be combined. This is shown in

$$\hat{u}(t_k) = \sum_{l=-K_1}^{K_2-1} \boldsymbol{h}_l \boldsymbol{s}[k+l] \tag{2.13}$$

where all filter coefficients are given by

$$\boldsymbol{h}_l \triangleq \begin{cases} \boldsymbol{W}^{\mathsf{T}} \boldsymbol{A}_b^l \boldsymbol{B}_b & \text{if } l \geq 0, \\ -\boldsymbol{W}^{\mathsf{T}} \boldsymbol{A}_f^{-l+1} \boldsymbol{B}_f & \text{else.} \end{cases} \tag{2.14}$$

When the FIR filter is downsampled, the difference is that window of $\boldsymbol{s}$ that is used moves with the value of DSR instead of just 1. That leads to the estimate given by

$$\hat{u}(t_k) = \sum_{l=-K_1}^{K_2-1} \boldsymbol{h}_l \boldsymbol{s}[Rk+l] \tag{2.15}$$

where R is the downsampling ratio. This shows that downsampling is an effective way of calculating fewer samples without losing information.

2.11 Software Tools

2.11.1 CBADC Python Library

The focus of the thesis is on the digital estimator of the CBADC. To get useful input to the DE the analog system and digital control need to be simulated. To do that the python toolbox made by Malmberg is used [19]. It is used to determine the system's parameters needed to reach the target SNR and calculate the coefficients for the digital estimation filter. The tool can generate the filter coefficients the digital estimation filter uses to calculate the samples. The control signal sequence samples can be generated from a theoretical input signal. The tool provides an estimate of the input signal that can be used to verify the SystemVerilog module.

The analog frontend must be initialized when simulating the system with the Python tool. The function *get_leap_frog* with parameters OSR, N, and bandwidth is used for this. This version is

not available in the main branch of the CBADC library at the time of writing this thesis. Thus the version contained from the *veriloga_v2* branch is used. To get the filter coefficients, the digital estimator is initialized as an FIR filter with the analog system, digital control, and filter length as inputs. The analog system and the digital control need to be simulated to get the control signal sequence. That is done by creating a simulator with the *get_simulator* function and running it sample by sample with *next* function. When running simulations on the system an input signal must be used. The standard input when testing oversampling converters is a sinusoidal signal [2]. With the Python toolbox, this can be done by using the function *Sinusoidal* to initialize the analog signal.

2.11.2 RISC-V Toolchain

The RISC-V toolchain is a C and C++ cross-compiler [20]. The toolchain can compile C or C++ files with gcc into assembly code or machine instructions. The compiled assembly code can also be converted to different memory formats depending on the use case. When this is done together with a linker script, the memory file can be used directly on a CPU. Before compiling the C code the toolchain needs to be set up to target the same parameters as the CPU the code should be run on i.e., what extensions that can be used.

2.11.3 Cadence Xcelium

The simulation tool Xcelium (Cadence Xcelium Logic simulator) is used to simulate the digital system. The RTL (Register Transfer Level) simulations and the GLS (Gate Level Simulation) simulations are performed using Xcelium. The simulator tool receives SystemVerilog files as input and parameter values. The SystemVerilog files are compiled with the parameter values set to the corresponding parameter input. One of the SystemVerilog files is the top-level module, a testbench. The testbench reads the hex file (memory), which is compiled C code as explained in Section 2.11.2. When running GLS, the SystemVerilog design file is replaced with a single file, a Verilog gate level representation of the circuit [21].

Xcelium prints messages during simulation to be used for debugging. Another debugging method is by probing signals and showing the waveforms of the signals; see Section 2.11.4. The memory can be dumped after the simulation. From the dumped memory, all calculated samples can be extracted. Correct execution of the filter is checked by comparing these samples with the reference calculations provided by the Python tool [19].

All signal transitions can be written to a VCD (Value Change Dump) file. This file is used for time-based analysis in the power estimation tool. Alternatively, a SAIF (Switching Activity Interchange Format) file can be dumped to perform an average power estimation by the power estimation tool.

2.11.4 Cadence SimVision

Cadence SimVision is used for debugging. SimVision is a graphical tool that can show waveforms of signals in the circuit. By storing the switching activity of some signals while running Xcelium, the waveform can be studied to find errors or verify the correct behavior of the circuit at all points in the simulation. Specific signals or all signals in a block can be shown. In SimVision, bus signals can be displayed as hexadecimal, decimal, or binary numbers to ease the reading of signal values [22].

2.11.5 Synopsys Design Compiler

The synthesis tool used is Synopsys DC (Design Compiler). DC compiles the SystemVerilog RTL code into a Verilog gate level file. While compiling, DC optimizes the design by leaving out unused registers and blocks of code. If the design does not meet the timing requirements, DC alters the

circuit to provide the necessary delay optimization, if this is possible. This can lead to a higher area, and thus, the circuit will probably consume more power [23].

It compiles the RTL files using the library files for a 28nm FDSOI technology. The library files contain physical implementation details on all the logic cells that can be used. This information gives the synthesis tool enough knowledge to estimate the area of the circuit, the power consumption, and the timing of the circuit. These estimates, however, are not precise estimation results. The design must be placed and routed for the synthesis tool to find the correct area. The synthesis tool lacks the switching activity of the circuit. Thus, it cannot calculate a precise result, as it uses default switching activity for all the nets in the circuit.

2.11.6 Synopsys PrimeTime

Synopsys PT (PrimeTime Static Timing Analysis) is used for power estimation. The tool receives the synthesized design, switching activity as a VCD file, and information about the specific library from the library files to estimate how much power the circuit consumes. The cell library used can be used in a low voltage threshold configuration and a regular threshold configuration. The supply voltage's lowest configuration is 0.80 V. The library is only used in the regular threshold mode, with a supply voltage of 0.80 V, as the design meets the timing requirements for all configurations.

Because the switching activity of the circuit is analyzed, the calculated results are more accurate than the results provided by the synthesis tool. The power consumed by the circuit can be estimated for individual scenarios, like if the circuit can run in various modes [24].

Chapter 3

Implementation

3.1 FIR Filter

In Equation 2.13 $\boldsymbol{s}$ is the control signal given from the digital control, and the individual values of $\boldsymbol{s}$ can only be -1 or 1. Due to the nature of the control signal, no multiplications are needed to calculate the estimate $\hat{u}$. The estimate can be calculated by checking if the $\boldsymbol{s}$-values are -1 or 1 and do subtraction or addition, respectively. Since a digital bit only can have the values 0 and 1, the value -1 will be represented by a 0, i.e., if there is a 0, a subtraction should be performed, and if there is a 1, an addition should be performed.

Each of the samples $\hat{u}(t_k)$ is calculated independently of each other but with the same $\boldsymbol{h}_l$. The difference between the samples is that $\boldsymbol{s}$ is updated (a new set of bits are shifted in) for each calculation of a new sample. With no dependence between the samples, the task can easily be parallelized. The number of additions done per sample is dependent on the size of the backward recursion (*K1*) and forward recursion (*K2*). The total recursion size, denoted as K, represents the sum of *K1* and *K2*. Throughout this thesis, *K1* and *K2* are always equal, and only *K1* will be used.

The filter coefficients, denoted as $\boldsymbol{h}_l$, are constrained to the range between -1 to 1, rendering them highly suitable for fixed point representation. The benefit of the most common alternative, floating point, is flexibility, but when no flexibility is needed, fixed-point will yield higher performance at a lower cost. The calculations performed are only additions and subtraction of the values in $\boldsymbol{h}_l$, and the result of these additions will never reach values higher than 1 or lower than -1, meaning no integer bits are needed. In the context of a 32-bit CPU, 32-bit fixed point numbers are the most natural choice, with 1 bit reserved for the sign, 0 for the integer part, and the remaining 31 bits allocated to the fractional part.

To reduce the size of the accelerator, fewer bits than 32 are used. Section 4.1 shows that 22 bits are the fewest bits needed for calculation without reducing the SNR substantially. If it is set lower, the SNR will be affected. Using one sign bit and none for the integer part of the number will result in the smallest possible increment between adjacent values being $2^{-21} = 4.768 \cdot 10^{-7}$. Using fewer bits for the numbers will reduce the size of the registers and the width of the adders, which means a significant reduction of area compared to a 32-bit implementation.

3.2 CPU Core

When choosing which CPU core should be used for an application, what operations the CPU should perform is crucial to know. For the digital estimation filter of the CBADC, only a few instructions are used, but they must be performed several times per calculated sample. Only add, load, store, and branch instructions are used in the filter calculations. Consequently, when using

a RISC-V core, no extension is needed to run efficiently, and the E extension can be used without slowing down anything. For a system that only should perform a specific task, benchmark tests of cores might not be very helpful as cores perform some tasks faster than others. It is more helpful to look at how fast it can do the specific tasks it is meant to do.

The chosen CPU core for this project is Picorv32. This RISC-V core is selected for its simplicity, low area, and power usage. The reason for using RISC-V compared to other ISAs is that the standard is open source. In [25], Picorv32 is the core using the least resources and has the lowest power consumption of the tested CPU cores. The Picorv32 core is also suitable for simple operations as it can be implemented using only the E extension. It is implemented with the simplest configurations to reduce the area and power consumption. The measured cell area of the CPU is 4519 μm^2 with 3972 logic cells.

Another benefit of the core is its built-in functionality for adding external logic via Pico Co-Processor Interface (PCPI) [4]. The signals of the interface can be seen in Table 3.1. The PCPI is made such that every time the core gets an instruction that it does not recognize the *pcpi_valid* signal is asserted. The instruction is written to the *pcpi_insn* signal, and all data the external logic needs must be at the *pcpi_rs1* and *pcpi_rs2* signals. Once the external logic completes the instruction, the result is written to the *pcpi_rd* signal, and *pcpi_ready* is activated, allowing the CPU to proceed with the following instruction. If no module has signaled it will complete the instruction, the CPU will continue with the next instruction.

Table 3.1: The PCPI interface signals [4].

Name	I/O	Width	Explanation
pcpi_valid	O	1	When the CPU receives an unsupported instruction, pcpi_valid is asserted.
pcpi_insn	O	32	When pcpi_valid is asserted, the unsupported instruction is broadcasted on pcpi_insn.
pcpi_rs1	O	32	Contains CPU-register $rs1$ value.
pcpi_rs2	O	32	Contains CPU-register $rs2$ value.
pcpi_wr	I	1	If the coprocessor is returning a value, pcpi_wr must be asserted.
pcpi_rd	I	32	The return value is written to pcpi_rd.
pcpi_wait	I	1	Is asserted if a PCPI module needs more than 16 cycles to complete the instruction.
pcpi_ready	I	1	Asserted when the coprocessor is finished executing its instruction.

3.3 Accelerator

The Picorv32 CPU core cannot calculate samples with a frequency of 20 MHz alone. One possible solution to this problem is to use multiple CPU cores in parallel, and another solution is to connect one CPU core to a custom-made hardware component to help speed up the calculations. The benefit of using multiple CPU cores in parallel is its flexibility after synthesis. A disadvantage is that it requires more complex memory management if multiple cores access the same memories. In addition: a CPU is not specialized, and some unnecessary logic is added per extra core.

In this thesis, the proposed solution to the CPU core's lack of speed is to use a custom accelerator connected to the Picorv32 core. An accelerator design can exploit the knowledge of the system to utilize features such as parallelism and pipelining, as explained in Section 2.6 and 2.7. Multiple copies of the same module can be placed in parallel to process multiple data elements at the same time. This would enhance performance significantly if the clock frequency is the same. If power is of primary concern, instead of using one unit, five units with clock frequency $\frac{1}{5}$ of the original unit could perform the same task with a significantly lower voltage source. Pipelining can be utilized in combination with parallelism. By having i.e. five pipeline stages, each stage only needs to compute 20% of the original computation. This could allow an increase in frequency and higher computational speeds. Another possibility is to keep the low frequency and lower the voltage to reduce the power consumption.

The digital estimation will, for each sample, add or subtract $N \cdot K$ coefficients together. For a configuration $N = 3$ and $K1 = 128$, 768 coefficients must be summed together. Computing sums of 16 coefficients in a module, 48 modules can calculate sums of all coefficients. The 48 sums can be summed by another three modules, and the sum of the three final intermediate results can be

summed together to the estimate. Three pipeline stages are thus implemented. The first stage uses 48 parallelized blocks, and the second uses three. The final stage is not utilizing any parallelization.

Even though the Picorv32 has lower performance than the other cores it is compared to in [26], it could perform well enough when the main calculation is outsourced to an accelerator. The accelerator is connected to the Picorv32 CPU, which is used as an interface to the memory, instruction fetch unit, and control unit. The CPU decodes some of the instructions and executes them as well. To connect the custom circuit to the CPU, the PCPI is used together with some custom-made instructions. Since the implemented module is used alongside the CPU and gets data from the CPU when used, it could be considered a coprocessor to the Picorv32. However, as the module uses its own optimized memory and is designed to perform a specific task, it is referred to as an accelerator.

Typically, a Picorv32 coprocessor will assert the signal *pcpi_ready* when it has completed its calculations. The CPU will wait for the assertion of *pcpi_ready* before it continues its following instruction. As we want the CPU and the accelerator to work in parallel, the *pcpi_ready* signal is asserted whenever the accelerator recognizes the instruction. The returned result of the calculation is thus the calculation of a delayed sample. The accelerator adds additional delay as the calculation is separated into stages, calculating intermediate results each time *start* is asserted. As long as the accelerator finishes its intermediate calculations sooner than the next custom instruction is issued, the samples will be calculated correctly. If not, the intermediate results will not be correct before passing them on to the following submodules. As the Picorv32 is a simple core executing the instructions in order, using a fixed number of cycles on each instruction, calculating the number of cycles between each issued instruction is straightforward. The number of cycles the accelerator module uses to compute the estimates can be tailored to the CPU core.

The integration of an accelerator with a CPU necessitates the establishment of a mechanism for the CPU to determine when to engage the accelerator. To achieve this, custom instructions are devised, then integrated into the toolchain to enable the compiler to recognize these instructions. The format of the custom instructions is the standard RISC-V instruction format using some unused masks. The instructions implemented are one for issuing the calculation and three different ones to load in data. All of them will, for simplicity, use two registers as input and one as output. To use these instructions together with the C-code, the assembler is modified to understand the instructions, and inline assembly in C is used to write them. Then the compiler chooses which registers to use for the operations in the custom instructions. A comprehensive breakdown of the custom instructions can be found in Appendix A.

3.4 SystemVerilog Modules

At the top level, a module connecting the Picorv32 to the accelerator can be found. While the Picorv32 is an open-source CPU, the accelerator is made by the authors of this thesis. The module consists of several submodules that together calculate the samples of the ADC. A block diagram of the top-level module is shown in Figure 3.1. The figure shows that the Picorv32 core is connected to the accelerator through the PCPI interface. It can also be seen that the memory is connected to the memory interface of the Picorv32. The memory is only simulated in the testbench and is thus outside the top-level module. Hence it will not be synthesized. All modules are implemented using the hardware description language SystemVerilog.

The different versions of the accelerators all consist of modules and submodules. The submodule lowest in the hierarchy is the adder module. This module adds together NUM_ADD number of values. The result is computed in NUM_ADD+1 cycles. One step higher in the hierarchy, multiple of these adder modules are combined to add the results from all submodules together. Highest in the hierarchy, the matrices $\boldsymbol{h}$ and $\boldsymbol{s}$ are read in, stored in the registers $\boldsymbol{S}$ and $\boldsymbol{H}$, and distributed to the lower levels. The top-level accelerator module controls the lower levels. The code for all the modules can be found in [27].

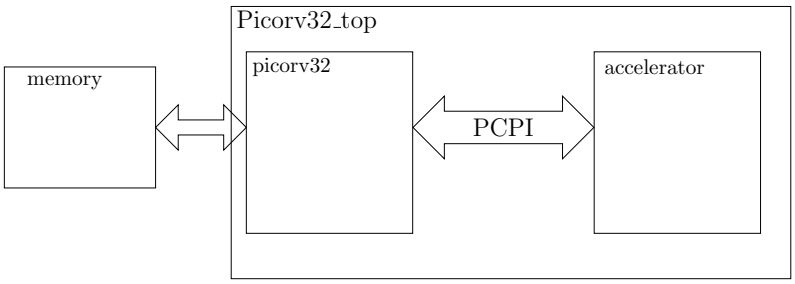


Figure 3.1: A block diagram of the top-level module connecting the Picorv32 CPU with the accelerator.

3.4.1 Accelerator Top-Level Module

The top level of the accelerator is in charge of controlling the accelerator module, as well as storing $\boldsymbol{h}$ and shifting in $\boldsymbol{s}$ when the CPU sends the coefficients and control sequence samples. When the *pcpi_valid* signal is set high, the module checks if the instruction is recognized as one of the custom instructions. If the instruction issued is to push a coefficient to the $\boldsymbol{H}$ matrix, the CPU will assign $rs1$ and $rs2$ (registers 1 and 2 of the CPU) to values of $\boldsymbol{s}$. The coefficient value will be found at $rs1$, while the analog state the coefficient belongs to is at $rs2$. The coefficient will be shifted into $\boldsymbol{H}$, which is a shift register. In Figure 3.2, a block diagram of the accelerator module is shown.

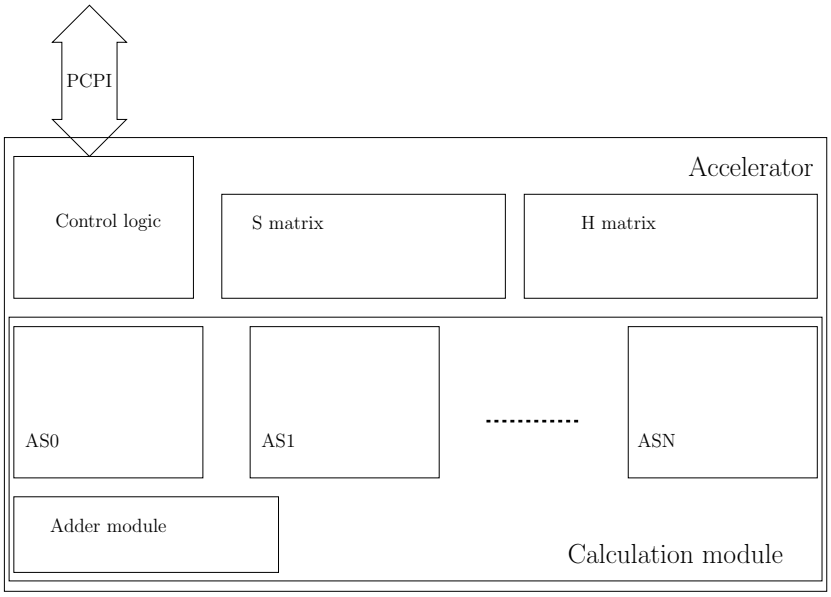


Figure 3.2: Overview of the accelerator module.

When the CPU provides new values of $\boldsymbol{s}$, both registers are used for control sequence samples. For all configurations of the accelerator, DSR is equal to OSR. This means that DSR sets of N control sequence bits are shifted into the shift register for each calculation of a new sample. All these bits are shifted into $\boldsymbol{S}$ in one clock cycle. At the same time, the signal *start* is asserted, signaling to the submodules to start calculating the value of a sample. The delay before the next time *start* is asserted must be at least as long as it takes for the addition submodules to sum all its operands. This will be the case for the filter as long as the implemented CPU fetches new instructions and control sequence values from memory before sending it to the accelerator, and this requires enough time.

At the same time as the calculation of another sample is issued, the accelerator will write the value of the previously calculated sample to the *pcpi_rd* for the CPU to read. The *pcpi_ready* signal will be set to 1 to let the CPU continue the program it is executing.

In the accelerator module, a calculation module is instantiated. This module separates the calculations into separate ones for each analog state, parallelizing the calculation of each analog state. The filter coefficients and control sequence signals are routed into the corresponding module, and an adder module adds the result of each state before it is transferred to the CPU using the PCPI interface.

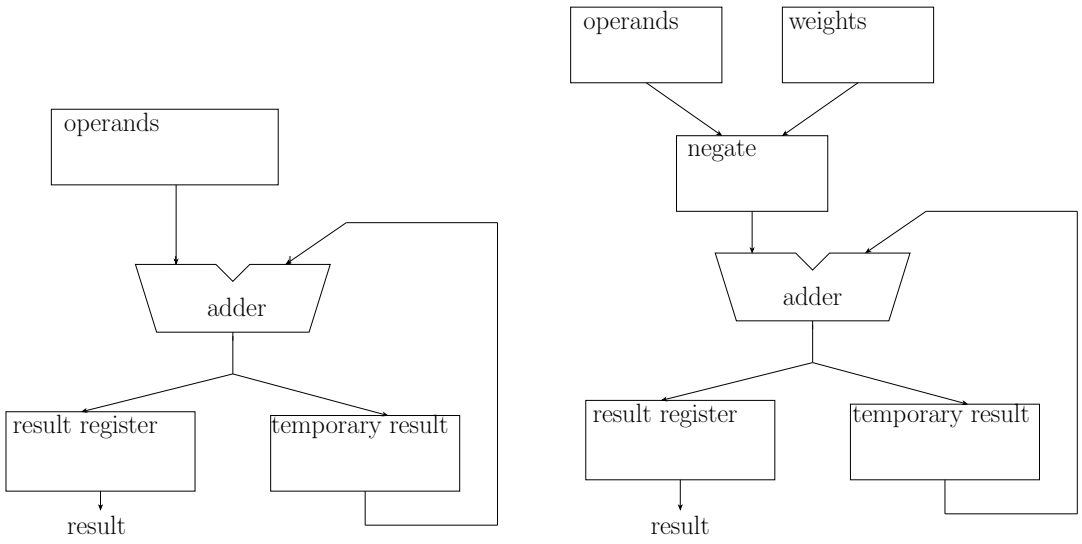
3.4.2 Single Analog State Calculation

This module adds together all coefficients of an analog state. First, the operands and the control signal for that analog state are passed to an array of modules adding together weighted sums of the coefficients, as explained in Section 3.4.3. This is the first step of a pipeline in the single analog state calculation. Each submodule calculates the result of NUM_ADD of the weighted coefficients and returns the sum, as explained in Section 3.4.3. The number of weighted summation submodules must be at least $(N \cdot K)/NUM\_ADD$ to calculate all weighted additions, utilizing parallelism in the accelerator. Since the total number of coefficients is much larger than the number of additions in a single weighted summation module, at least one more pipeline stage of adder modules needs to be implemented. The adders used in this step are non-weighted adders. They add together the results from the previous stage.

3.4.3 Multi-Cycle Adder

The multi-cycle adder is made in multiple versions. A simple adder that only adds numbers together, a weighted adder receiving the $\boldsymbol{s}$-values corresponding to the $\boldsymbol{h}$-values and calculates the weighted sum, and multiple versions using LUTs.

The first version shown in Figure 3.3a receives the operands as input. When the signal, *start*, rises, the module will start calculating the sum. One of the operands will each clock cycle be added to a register containing the temporary result. When all additions are performed, the output *res* is assigned the sum of all the operands. The module receives inputs *clk*, *resetn*, and *enable* as well. When *resetn* is 0, the module will be reset, setting *res* and the temporary result register to 0. The enable signal will stop all calculations in the module if deasserted.



(a) Simple adder module that adds operands together.

(b) Adder module that adds weighted operands together.

Figure 3.3: Two of the adder modules implemented.

The second version shown in Figure 3.3b receives the $\boldsymbol{S}$-values corresponding to the $\boldsymbol{H}$-coefficients. These $\boldsymbol{S}$-values are used as a weight for the addition. As in the first version, one coefficient is added

each clock cycle. But the coefficient is weighted, where the weight 1 means use the coefficient as it is, and 0 means multiply with -1 before adding (same as subtraction). This will add some extra logic to the module.

3.5 Core Optimizations

3.5.1 Reduced Signal Widths

When creating filter coefficients, some observable patterns concerning the size of the numbers are found. These patterns depend on which analog state they represent and the coefficient's proximity to the filter's center. These patterns are illustrated in Figure 3.4. When introducing more analog states in a CBADC, the coefficients of new analog states are mostly lower than the corresponding coefficients in the lower analog states, where $\boldsymbol{h_1}$ are the coefficients corresponding to the lowest analog state. Therefore, a smaller number of bits can be used to represent the coefficients. When the bit-width of the coefficients is reduced, the sizes of signals and adder modules can also be reduced. By sign extending the signals when they are being added together with wider signals, the results will be the same as if all signals had the same wide bit-width, to begin with. This means that the SNR will be identical.

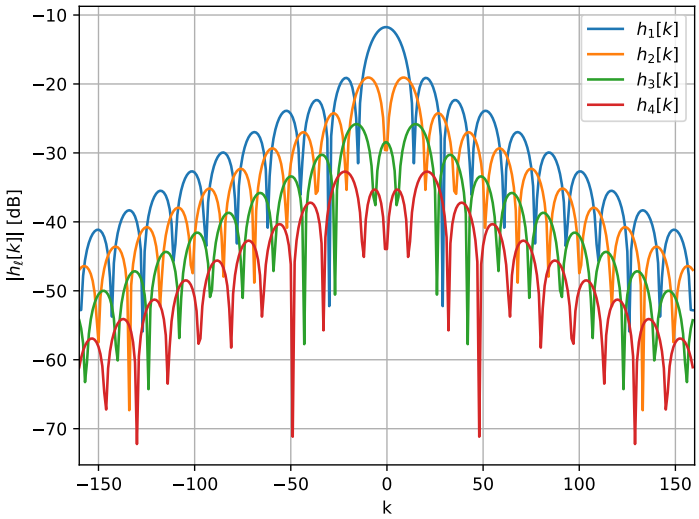


Figure 3.4: Absolute value of the filter coefficients when $N = 4$.

By reducing the size of the coefficients, the removal of the most significant bits is meant. A number that is small enough not to alter the coefficient value or intermediate results must be chosen to preserve the precision of the CBADC. Because the numbers are signed fixed-point numbers only 0s are removed from the most significant bits of positive numbers and 1s from the negative numbers. The potential decrease in area and power consumption could be significant.

From Figure 3.4 we see that for the lowest analog state, $\boldsymbol{h}_1$, the peak coefficient values are larger than for $\boldsymbol{h}_2$, which in turn has a larger peak than $\boldsymbol{h}_3$. Because of this, the number of bits representing the coefficients and bit-width of calculations can be made smaller. The higher the analog state is, the fewer bits need to be used. This implementation will save both area and power by instantiating smaller $\boldsymbol{H}$ registers in the accelerator and smaller single analog state calculation modules.

As mentioned, there is likewise a trend in coefficient magnitude based on how close to the center of the filter they are. Coefficients near the center tend to be larger than those near the edges. Additional area and power can be saved by reducing the coefficients' bit-width closer to the filter's

edges.

How many bits can be removed is individual for the different numbers of analog states. This means each configuration needs to have different numbers of bits removed for each analog state. To find how many are required, a plot of how many bits are used for each of the coefficients is made, and an example of the coefficients when $N = 4$ can be seen in Figure 3.5. Since the optimization is done both to the coefficient registers and in the adder modules, there must be some headroom to store the results of the additions, which will be larger than a single coefficient. For this reason, the dotted line is 4 bits higher than the minimum needed for only the registers.

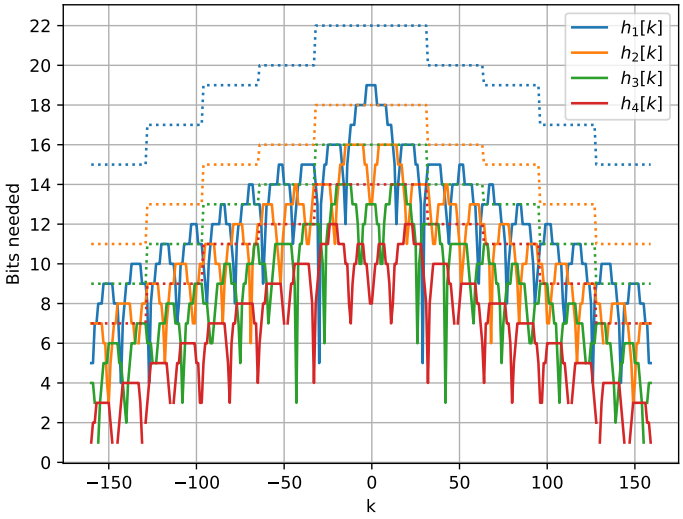


Figure 3.5: The number of bits needed to store each coefficient of $\boldsymbol{H}$.

How many extra bits are needed on the specific system is found through trial and error. The system with the highest number of bits removed, which does not give any changes in the calculations compared to the reference version, is used. First, the optimization for each analog state is implemented. This removes a number of bits for all coefficients and adders in a single analog state. The number of bits in the state would be set to the value at $k = 0$ of the dotted line in Figure 3.5. This exploits that the coefficients for the higher analog states are smaller than those for the lower states. Table 3.2 displays how many bits can be used in the modules with reduced coefficient sizes compared to the reference version without optimizations. The number of bits in the table is the total number used in the registers to represent the coefficients. The number of bits in the $\boldsymbol{H}$ matrix register without optimization is calculated by $N \cdot K \cdot$ FPB, i.e., $3 \cdot 256 \cdot 22 = 16896$ for $N = 3$.

Table 3.2: Number of bits used to store the filter coefficients with different optimizations.

N	$K1$	FPB	Bits with no optimization	Bits with optimization per analog state	Bits with both optimizations	Decrease with both optimizations
3	128	22	16896	14592	12288	27.3%
4	160	22	28160	22400	18048	35.9%
5	160	22	35200	27520	22400	36.4%
6	224	22	59136	44800	35584	39.8%
7	224	22	68992	54208	43456	37.0%
8	256	22	90112	71168	55808	38.1%

The calculations in the accelerator are done in blocks of 32 values. Each of these blocks are optimized to use fewer bits the farther away from the center of the filter the coefficients are. There is no optimization for the coefficients of $\boldsymbol{H}$ from -32 to 31. From the following 32 coefficients on each side of them, the number of bits is reduced, and the size of the reduction is more significant

at the edges of the filter. The number of bits used for each block is chosen by an offset to the number of bits in the largest coefficient. The reduction at the filter's edges can be more significant with a larger K or steeper decline of the absolute value of $\boldsymbol{h}$.

3.5.2 Lookup Tables

One method of reducing the number of additions that needs to be performed by the estimation filter is by implementing LUTs (Lookup Tables) that would replace the regular $\boldsymbol{H}$ matrix. Adding together all possibilities of sums for a group of $\boldsymbol{h}$ values and keeping them in memory can remove many summations from the execution of the program. This could speed up the execution of the program, but a disadvantage of this method is an increase in area, as $\boldsymbol{H}$ matrix as the LUT becomes larger.

The smallest possible LUTs could be made by storing the negated value of all $\boldsymbol{H}$ matrix values and the non-negated number together. If the address of the negated version ends with a 0-bit, the non-negated address should be the same only with a 1 as the LSB (Least Significant Bit). By ORing the corresponding $\boldsymbol{S}$-value with the last bit of the address of the non-negated coefficient, the resulting address could be added as it is without negating it when the $\boldsymbol{S}$-value is negative. This would double the memory consumption of $\boldsymbol{H}$.

Taking it one step further, one could add combinations of two coefficients together. The negated coefficients are added together for two 0s in the corresponding $\boldsymbol{S}$ matrix. For one 0 and one 1 in the $\boldsymbol{S}$ matrix, one number is negated, and the other is not; for two 1s, none are negated. An example of this is shown in Table 3.3. For this LUT, 2 input bits must be used for choosing what element in the LUT should be used. The memory of $\boldsymbol{H}$ would be twice the size of an implementation with no LUTs, i.e., equal to that of the simplest LUT version. The reason for this is that for each two $\boldsymbol{S}$ values, there are four possible combinations of the bits, but when combining two and two coefficients, the length of $\boldsymbol{H}$ will be reduced to half of what it was.

The size of $\boldsymbol{H}$ will be LUT_SIZE times larger than $\boldsymbol{H}$ in the reference version for all implementations of the LUTs except for the first implementation, where it doubles. The reason for this is the increase $\boldsymbol{H}$s width by a 2^{LUT_SIZE} but reduces length by a factor LUT_SIZE. The size of the LUTs can be increased much more at the cost of more memory consumption. The LUT configuration is implemented in the first step, with the additional negated $\boldsymbol{H}$ matrix coefficients, with two, three, and four values combined into LUTs. All of these variations are implemented to explore how area and power are affected by using more registers to reduce switching activity caused by additions.

Table 3.3: Implementation of LUT combining 2 samples.

Input	Output
2'b11	$\boldsymbol{h}_l[1]$ + $\boldsymbol{h}_l[2]$
2'b10	$\boldsymbol{h}_l[1]$ - $\boldsymbol{h}_l[2]$
2'b01	-$\boldsymbol{h}_l[1]$ + $\boldsymbol{h}_l[2]$
2'b00	-$\boldsymbol{h}_l[1]$ - $\boldsymbol{h}_l[2]$

3.5.3 Post-Synthesis Parameterizability

In some cases, having a post-synthesis parameterizable module can be interesting. In an implementation where the input signal $\hat{u}$s strength varies greatly, the need for a high number of states and large filter length is not always necessary. If the noise floor is stable, the ADCs accuracy does not need to be as high when the signal is strong as when it is not to achieve the same SNR. Another case is if the signal is only interesting in some periods and must be monitored to find when. If the monitoring does not need the same SNR, the filter length or number of used analog states can be changed to save power.

By having the possibility to change the number of analog states used, or the width, K, of the filter, the properties of the filter will change. N and K can either be changed one at a time or both at the same time. With a reduction of these parameters, power can be saved. This power saving comes at a cost of an increase in area and some more control logic that will increase the area of the circuit. For the parameterizable version, the DSR equals the OSR for the largest system it can be configured to, i.e., $N = N\_MAX$. The SNR for a system with different OSRs can be seen in figure Figure 4.1.

Before the synthesis of the design, the maximum number of analog states N_max and the maximum width of the filter, K_max, must be set. N_max can be between 3 and 8, and K_max between 64 and 512 and must be a multiple of 32. The gate level design will be able to change the number of analog states between these maximum values and smaller values. This is accomplished by using a custom instruction to change the registers N_en and K_en. The register N_en is N_max bits wide, and K_en is $K\_max/32$ bits wide. Each bit of the vector N_en corresponds to an analog state. If a bit is 1, its corresponding adders are turned on, and if it is 0, the adders are turned off. This is similar for K_en, where each bit corresponds to 32 filter coefficients, 16 are from the first half of the filter, and 16 are from the second half. If, for example, K_max is 320, K_en is 10 bits wide. If the first five bits are 1 and the last five are 0, only the 160 coefficients from the middle of the filter are used. The rest is not used for calculations. Thus, the circuit will have less switching activity, and power will be saved. In addition, the K_en register controls how much of the shift register $\boldsymbol{S}$ is used. If only the first five bits of K_en is 1, $\boldsymbol{s}$ will only be shifted into 160 of the 320 register blocks of $\boldsymbol{S}$s analog states.

3.5.4 Remove Low-Pass Filtering from Digital Estimator

Another way of optimizing the area and power consumption is to remove the low-pass (LP) filtering part of the DE. This means that higher-frequency noise will be carried through the DE, and the filtering needs to be done at a later stage. The reason this could be interesting is that it changes the filter coefficients such that the filter length could be shorter. To make the coefficients for a system without LP filtering, the DE is initialized with eta2=1 in the Python toolbox instead of eta2 being calculated dependent on the bandwidth. This will make the system the same but with no filtering.

By looking at Figure 3.6, we can see that the power spectral density (PSD) for the version with and without the LP filtering is about the same for frequencies below 20 MHz. This is where the noise generated in the analog system starts to be significant. This clearly shows that when the LP filtering is removed from the DE, it must be implemented at a later stage.

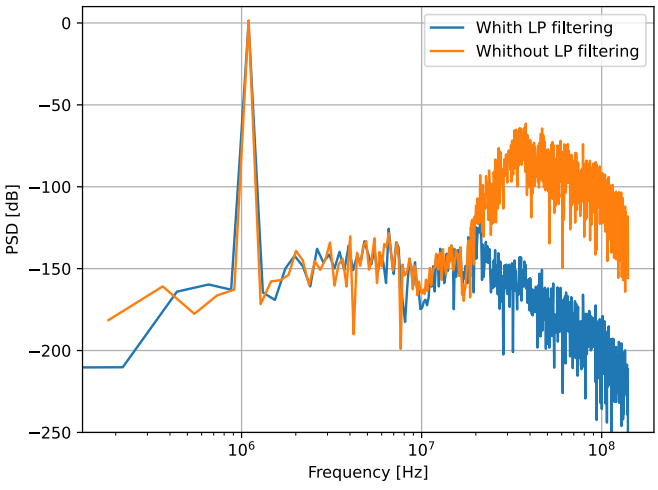


Figure 3.6: PSD for $N = 5$ OSR $= 7$ with and without LP filtering in the digital estimator.

3.6 Memory

When running simulations of the system a simplified memory system is used. When the testbench receives a request for a memory block, it waits for 1 ns before the data is valid, and the CPU can read it. This is because a typical latency for a level 1 cache is 1ns [18]. The constant memory delay is added to get a more realistic memory read time, even though it is not that simple in the real world.

In a realistic implementation, a fast level 1 cache would be implemented. In addition, another larger cache, between the level 1 cache and the slow main memory, would most likely be implemented. Since the calculation is done sequentially on the data from the analog part of the CBADC, the memory could be very efficient using a prefetcher that preloads data from the main memory to the faster cache. As only theoretical memory is used, there has not been a focus on optimizing the memory access patterns of the accelerator.

Chapter 4

Results and Discussion

The results and discussion chapter is divided into two main parts. To begin with, we explore the quality of the results of the digital estimation. This means examining how the coefficients and parameters of the filter affect the SNR. The results of these analyses are used to choose parameter values suitable for achieving the target SNR while consuming small amounts of power. The results of these analyses are found in Section 4.1.

In the second part of the results and discussion section, we explore how the circuitry can be tweaked to reduce area and power consumption while maintaining a high sampling frequency and meeting the target specifications. Multiple variations of the accelerator are implemented, and the version of the filter with a single $\boldsymbol{h}$, no LUTs, and no reduction of bit widths will be the reference which all other versions will be compared against. The different versions have used the reference version's RTL code as a base, and their extra functionality is added to this as it is the simplest version.

4.1 Choosing Filter Parameters

When designing an A/D converter, SNR is one of the parameters used to measure performance. As this thesis aims to design a system capable of sampling with an SNR of at least 70 dB, the analog system must have an SNR not smaller than 70 dB. The SNR of the analog system should not be substantially higher either, as this will consume more power. The digital estimator should not significantly impact the SNR as that would lead to an SNR lower than 70 dB. The analog system's performance should set the CBADC's SNR, and the digital system should calculate the samples without losing precision.

Later chapters will explain why the DE will use less power if $K1$, N, and bit width are as low as possible. A comparison between different values of N is wanted, and N is thus varied between 3 and 8. The parameters used when designing the analog system are N and OSR. Figure 4.1 shows how OSR and N impact the SNR when the digital system is more extensive than it needs to be. The SNR of the systems with a high number of analog states is reduced when the OSR becomes too large. Even if $K1$ is 1024, the digital estimation filter will still reduce the SNR of these filter implementations. However, this happens well after the 70 dB line is crossed. As we do not want such high OSRs, this decline in OSR is thus irrelevant. Figure 4.1 shows that for a lower N, the OSR needs to be higher to reach the target SNR. The OSR chosen for the system is, for each N, the lowest OSR where the SNR has reached more than 70 dB. This means for $N = 4$, OSR is set to be 15 since that is where the yellow line crosses the pink one.

SNR is calculated for various values of $K1$ to find the lowest one needed for a given N and its corresponding OSR. The accelerator groups together coefficients in groups of 32 when calculating the samples. Thus, the parameter $K1$ is chosen as a multiple of 32. Figure 4.2 shows how the SNR is impacted by a higher value of $K1$ when N is 4. The plot shows that for OSR = 15, $K1$ needs to be at least 160 not to reduce the SNR of the system compared to higher values of $K1$.

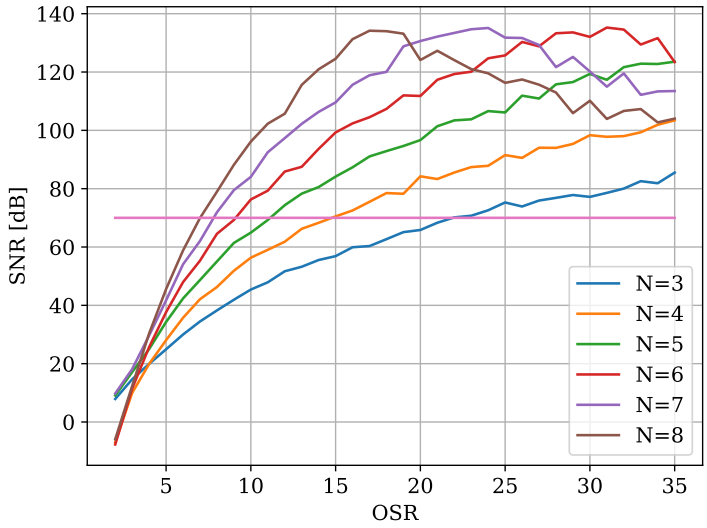


Figure 4.1: SNR vs. OSR for different numbers of analog states with $K1 = 1024$.

The K1 needed for the other numbers of analog states is found through similar plots located in Appendix B.

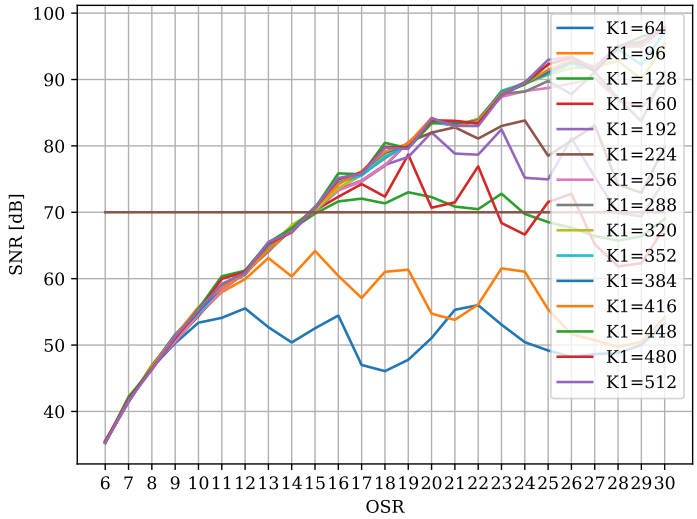


Figure 4.2: SNR with different values of $K1$ and OSR for $N = 4$.

Another parameter that can be optimized for lower area and power consumption is the bit width of the coefficients. In addition to reducing the size of the values that must be stored, the width of the adders can be reduced. The values used in the calculation are always numbers between -1 and 1, which means the MSB is used as sign bits, and the rest are used to store the number. To find how many bits are necessary, each value of N is reviewed individually.

Figure 4.3 is used to determine how many bits are needed for $N = 4$. The figure shows that the SNR rises when the number of fixed point bits (FPB) used increases before stabilizing around the dotted line. The dotted line is the SNR when FPB is 32 and is hence the target to reach. The place the line with OSR = 15 is equal to the dotted line is when FPB is 22. The figures in Appendix C show that FPB must be 22 for all the versions. When FPB is 22, one bit is used as a sign bit, and 21 bits are used for the decimal part, which means the smallest difference between two numbers

that can be represented is $4.768 * 10^{-7}$

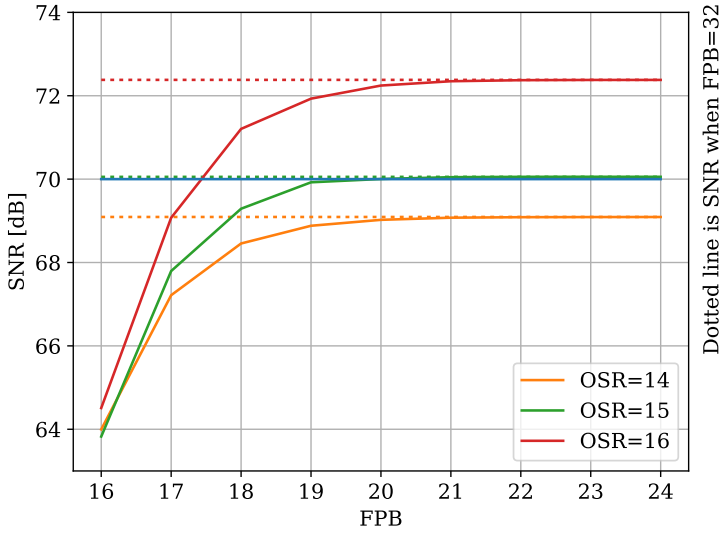


Figure 4.3: SNR for different numbers of bits used for $N = 4$.

When looking at the different graphs in Figure 4.3, it can be seen that a higher FPB is needed when increasing the SNR target. This is because higher SNR requires more precision to detect smaller variations. This shows that if the system is designed with a higher SNR target, FPB must be higher, and FPB can be lower if the target SNR is lower.

Table 4.1 shows the OSR, $K1$, and FPB needed to obtain an SNR of at least 70 dB. The lowest OSR possible is chosen. Then the lowest $K1$ and FPB that satisfy the specifications are chosen. The values in the table were obtained from Figure 4.1, 4.2 and 4.3, in addition to the plots in Appendix B and C. Table 4.1 show if more analog states are used, $K1$ needs to be higher and OSR can be lower. This means that for the DE, a lower N is better to reduce the area. A lower N requires a higher OSR and more bits from $\boldsymbol{s}$ must be shifted into $\boldsymbol{S}$ for every sample. Compared to a digital system with the same number of analog states and filter length, the switching activity and, thus, power consumption of a system with a higher OSR will be larger than that of a lower OSR. A higher OSR might have a negative impact on the analog system's power consumption.

Table 4.1: Requrements for the system to get 70 dB.

N	OSR	$K1$	FPB
3	20(23)	128	22
4	15	160	22
5	12	160	22
6	9	224	22
7	8	224	22
8	7	256	22

For $N = 3$, an OSR of 20 is chosen despite 23 being required for an SNR of 70 dB. This is because when loading control sequence samples from memory into the accelerator, the PCPI interface cannot transfer 23 samples per analog state in one transmission. The PCPI interface is restricted by using two 32-bit registers, which means 64 bits can be transferred simultaneously. And when OSR and, therefore, DSR is 23, that means $3 * 23 = 69$ bits should be transferred. It could be possible to load more bits per calculation, but the CPU uses more time on memory reads, leading to a lower sampling frequency. The highest OSR supported without decreasing the sampling frequency is thus 21. Implementing OSR = 21 required more complex implementation than OSR = 20, and the implementation would still not reach the target SNR of 70 dB. An implementation

of OSR = 20 was chosen as the insight gathered would be the same as for OSR = 21.

4.2 Frequency

In [9], it is stated that the main problem of using a general-purpose CPU for the DE in the CBADC was the speed of calculation. A simple general-purpose CPU can only execute one instruction at a time. For an unoptimized FIR filter in a CBADC, the number of additions to be computed for each sample is equal to $N \cdot K - 1$, which for the smallest filter configuration is 765 ($N = 3$ and $K = 256$). With a target sampling frequency of 20 MHz, the time to calculate all these additions is 50 ns per sample. This will not be possible with a simple CPU as it would require a frequency of addition of more than 15 GHz. The number of additions per sample will increase with a larger N or K. This thesis proposes a solution where an accelerator calculates many additions in parallel to achieve the target sampling frequency. This will enable a much larger number of calculations per clock cycle, thus creating a higher possible sampling frequency.

When calculating samples without downsampling, each sample can be computed faster since less data is loaded to the CPU for each sample. An issue is that more samples must be calculated to get the same bandwidth. In Equation 2.1, it can be seen that when the OSR increases, the bandwidth decreases for the same sampling frequency. The signal is downsampled to get the highest bandwidth for a given sampling frequency. When combining Equation 2.1 and Equation 2.2, we can see that the optimal downsampling is DSR = OSR, which is used in this thesis. When OSR and DSR are equal, the maximum possible bandwidth is half the achieved sampling frequency.

The achieved sampling frequency is given by the number of samples calculated divided by the calculation time. For all simulations in the thesis, the sampling frequency, F_s, is approximately 21 MHz, which is above the target of 20 MHz. Despite the significantly higher number of additions to be executed for configurations with larger values of N and K, the execution time does not noticeably increase in those cases. The reason for this is that the additions are performed in parallel. The CPU will execute the same number of instructions for all versions. For every sample calculated, the CPU will fetch control signal values $\boldsymbol{s}[k]$ that will be input to the accelerator.

The clock frequency of a system can be increased to achieve a corresponding increase in sampling frequency. However, this is only possible if the peripherals, such as memory, can deliver instructions and data at a higher rate. This is because the Cycles Per Instruction (CPI) remains constant, and with a faster clock, instructions are completed more quickly. If the circuit operates close to its constraints, increasing the clock frequency may only be possible by modifying the circuitry. The synthesis tool can increase the area of the circuit to reduce negative slack and meet timing requirements. Increasing the supply voltage can help a design meet its timing requirements without increasing the circuit's area. However, it is essential to note that increasing the clock frequency will result in a higher switching power, as we see from Equation 2.8. As discussed in Section 2.3, increasing the supply voltage results in a squared increase in power consumption. Consequently, it is desirable to use the lowest possible supply voltage.

The main limiting factor of the sampling frequency is the CPU core. The CPU must read two 32-bit data elements from memory for each calculated sample, containing the control sequence samples $\boldsymbol{s}[k]$. The CPU must read instructions and perform branching. As the CPU uses some cycles per instruction, the total time to complete these instructions is close to the maximum permissible time per sample. A CPU able to reduce the CPI at the same clock frequency would increase the possible sampling frequency. This could be enabled by pipelining or parallelism in the processor. However, this requires additional control logic, which would increase the size and power consumption of the circuitry.

If the system is designed for a higher target SNR than 70 dB, the OSR must be increased. To have the same sampling frequency, the DSR must also increase. However, since more control signals are used per sample, and the PCPI bus is restricted to transfer 64 bits simultaneously, the OSR cannot be raised significantly before the CPU restricts the sampling frequency. The solution to transfer more than 64 bits is to use an extra instruction on the CPU to send some of the control signals before the calculation. The problem when the CPU issues more loads from memory is that

the time between the estimation of each sample increase, and thus, the frequency will decrease.

4.3 Limitation of the Cell Library

The power estimates presented in this thesis are all generated using Synopsys PrimeTime. As explained in Section 2.11.6, the power is estimated using the synthesized design, a switching activity file, and information about the cell library. The cell library configuration of 0.80 V, the library's lowest possible supply voltage level, is used in the synthesis. The slack of the most critical path is close to 0, and thus the synthesized design will not have the capability of a lower supply voltage. However, if the design had been synthesized with a lower supply voltage, the tool would change the netlist to find faster but more power-hungry logic blocks. This way, a lower supply voltage might lead to a larger design. The increased area increases the power consumption, and the reduced voltage reduces it. There is a tradeoff between supply voltage and area. The increased area will increase the cost of the chip as well.

By reducing the device threshold voltage as well as the supply voltage level, the dynamic power consumption would decrease, while the static power consumption would increase. As seen in Appendix E, the static power of all designs is below 2% of the total power. Because the static power is small compared to the dynamic power, we can tolerate a much more significant increase of it than a reduction of the dynamic power. Because a reduction of dynamic power consumption by 3% would reduce total power consumption even if static power is doubled. The supply voltage level and device threshold voltage should thus be lowered to the point where reducing them would result in higher power consumption.

4.4 Reference Version

The reference version of the digital estimation filter is implemented as explained in Chapter 3. The reference version has no optimizations implemented, meaning there is no bit width reduction, and LUTs are not implemented. This is to review the impact those optimizations have, compared to not implementing them. The total cell area of the reference design is found through the synthesis of the circuit. The total cell area for the reference versions of the digital estimation filter is shown in Figure 4.4. The clock frequency of the designs is set to 1 GHz, which gives a sampling frequency of just above 20 MHz, as shown in Section 4.2.

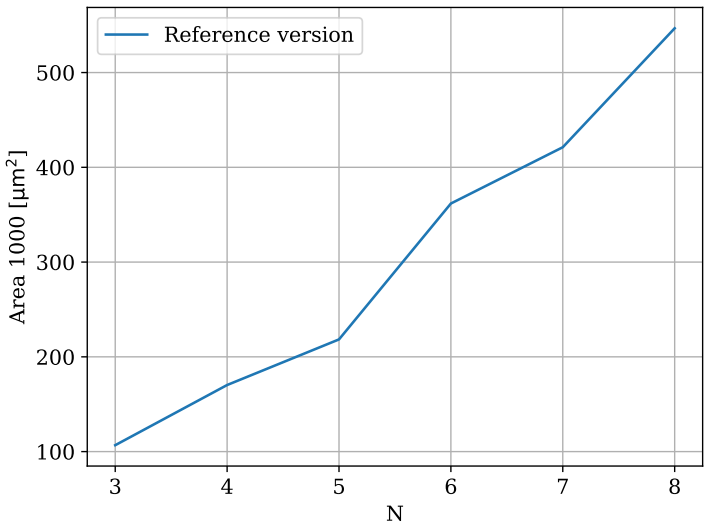


Figure 4.4: The total cell area of the reference versions of the digital estimation filter.

Figure 4.4 shows that the area increases for an increase in N. The rise in the area is not linear, however. The reason for this is the growth of K as N becomes larger. The size of the $\boldsymbol{h}$ and $\boldsymbol{s}$ matrices grows proportionally with the factor $N \cdot K$. This also applies to the number of additions that needs to be computed. By looking at Figure 4.5, which shows the normalized area of the circuit compared with the normalized product of N and K, we see a strong correlation between the area and product. The only parts of the design unaffected by N and K are the Picorv32 core and the accelerator's finite state machine (FSM) logic.

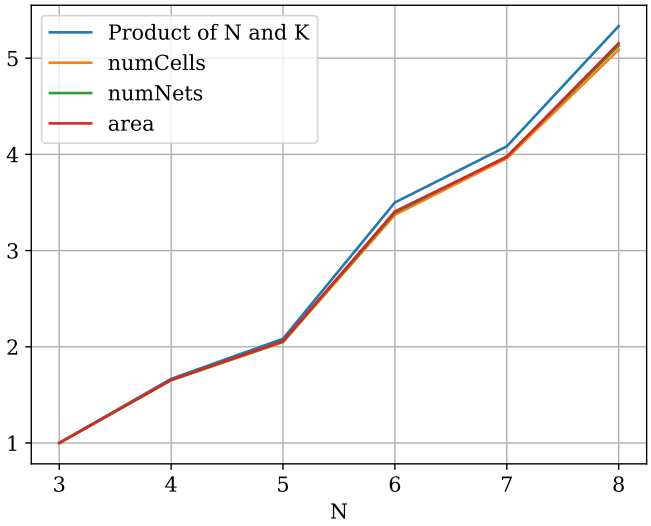


Figure 4.5: The normalized area, cell count, and number of nets of the circuit compared with the normalized $N \cdot K$.

The product of N and K has a slightly larger increase. This is likely due to the area of the CPU and control logic of the circuit not increasing as N and K rise. Because Place and Route have not been performed, the estimates of the areas are not precise. This is why the cell area, number of cells, and number of nets have such a strong correlation between them. Because of this, we will only look at the total cell area of the synthesized designs. The total cell area of the smallest configuration, $N = 3$, is at 106 757 µm^2, and for $N = 4$, the area is 170 255 µm^2. For $N = 8$ the area is significantly larger for $N = 8$ with an area of 546 669 µm^2, approximately five times larger.

Of the $N = 3$ configuration, 4636 µm^2 of the total area is the Picorv32, approximately 4.3% of the total area. The accelerator module is then the remaining 102 121 µm^2. Of these, 52 391 µm^2 is $\boldsymbol{H}$, consuming 49.1% of the area. With $\boldsymbol{S}$ consuming 1629 µm^2 of the accelerator, all the adder logic and control logic is the remaining 48 101 µm^2. When looking at the $N = 8$ module, 279 419 µm^2 is $\boldsymbol{H}$, thus 51% of the total area. $\boldsymbol{H}$ is, by far, the largest component of the system. Reducing the size or number of coefficients in $\boldsymbol{H}$ would thus be beneficial.

In Figure 4.6, we see the power consumption of the reference version for all N in the range of three to eight. The power consumption is closely related to the area, thus N and K. The total power is lowest for the $N = 3$ configuration at 7.36mW and rises to 36.90 for the $N = 8$ configuration. For $N = 4$, the power consumption is 11.70 mW. The total power for the smallest configuration is thus approximately 19.9% of the power of the largest. This is close to the ratio of the total cell area between the configuration, where the $N = 3$ implementation consumed 19.5% of the area of the $N = 8$ implementation.

The leakage power of the digital estimator in the reference version is negligible as they are below 0.5% of the total power for all configurations. It is 0.45% for $N = 3$ and $N = 4$ and 0.46% for the remaining configurations. Not surprisingly, the leakage power grows as the area becomes larger. The ratio between the leakage power and the total power does increase, but barely and does not significantly affect the circuit's power consumption. As explained in Section 2.4, a more balanced proportionality between the static and the dynamic power consumption would be more beneficial.

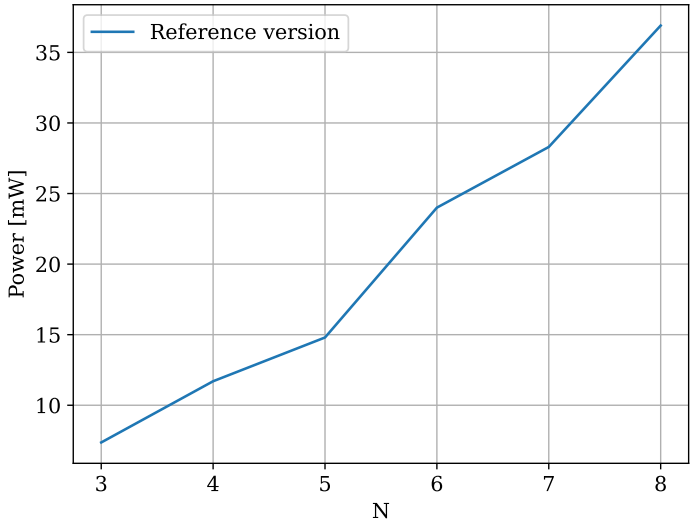


Figure 4.6: Power consumption of the reference version.

The supply voltage and voltage threshold could then be lowered to significantly reduce the dynamic power but then increase the static power. Variations in the leakage power would then be of higher importance.

4.5 Lookup Tables

The accelerator is implemented in four different versions using LUTs. The simplest version, not combining multiple coefficients, only duplicating $\boldsymbol{H}$ and negating it, is having $LUT\_SIZE = 1$. The LUT_SIZE refers to the number of input bits to the LUT. The largest version, with $LUT\_SIZE = 4$, combines four and four coefficients, reducing the number of additions in the accelerator to almost a fourth compared to the reference version, at the cost of a larger area of $\boldsymbol{H}$. Section 4.4 explained how the $\boldsymbol{H}$, with its coefficients, occupied almost 50% of the total cell area of the reference version. When implementing LUTs, the size of $\boldsymbol{H}$ becomes larger, which is the effect we see in Figure 4.7.

The area of $\boldsymbol{H}$ in the versions with LUT_SIZEs one and two is doubled compared to the reference version, and the total cell area increases by up to 43% for $LUT\_SIZE = 1$ and up to 39% for $LUT\_SIZE = 2$. An example is the $LUT\_SIZE = 2$, $N = 4$ version. The total cell area of the design increased to 279 102 µm^2 from the 170 255 µm^2 of the reference version. The area for the $LUT\_SIZE = 1$ version is slightly larger than for the $LUT\_SIZE = 2$ version. The reason for this is that $\boldsymbol{H}$ is twice as long (but half the width), meaning twice as many additions must be performed by the accelerator. This requires additional adder modules.

For the implementation with $LUT\_SIZE = 3$, the area is larger than for $LUT\_SIZE = 2$ and slightly larger than for $LUT\_SIZE = 1$. The area of the accelerator with $LUT\_SIZE = 4$ is significantly larger than the reference version. The area increases up to 140.3% for $N = 3$ and 148.7% for N=4. The other configurations have an area increase between these. The area increase seems to be independent of the size of N. For $LUT\_SIZE = 4$, $N = 4$, the area increased to 423 348 µm^2.

Figure 4.8 shows the power consumption of all implemented accelerators utilizing LUTs. The power consumption increases for a LUT_SIZE of 1 compared to the reference version. This is because the number of additions and subtractions needed to be performed stays the same. The fact that the adder module no longer needs to perform subtractions, only additions, does not seem to reduce the power significantly. The added logic is to decide if the standard coefficient or the negated one

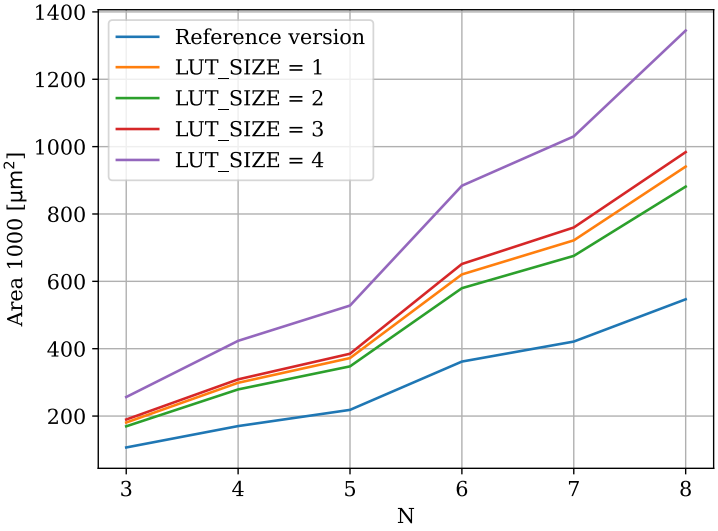


Figure 4.7: Total cell area of all DE filters with LUTs.

should be used, and the increase in static power increases the total power consumption. Thus, This module has a larger area and higher power consumption than the reference version.

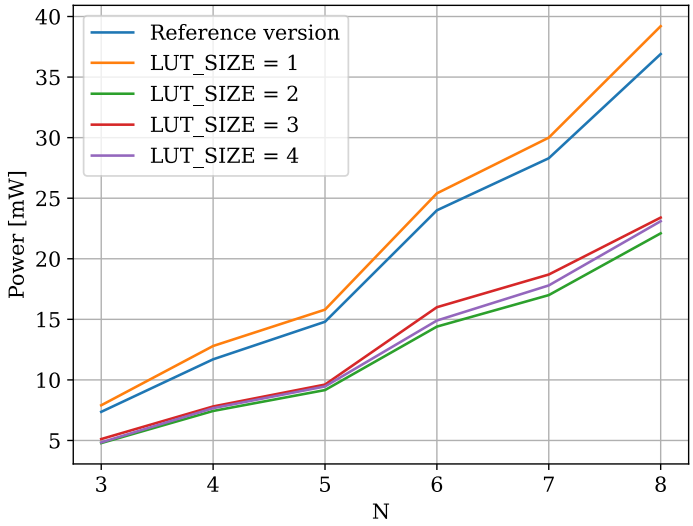


Figure 4.8: Power consumption of all DE filters with LUTs.

The other accelerators utilizing LUTs, however, consume less power than the reference version. For $LUT\_SIZE = 2$, the power is reduced by 35% for $N = 3$, and 40.1% for $N = 8$. For $N = 4$ the power consumption is 7.44 mW. The reduction seems to increase slightly as N increases. When the LUT_SIZE increase further, the power consumption grows. The power consumption for $LUT\_SIZE = 4$ is slightly higher than for a LUT_SIZE of 2. The power savings are between 34.3% and 37.9%.

The reason for the reduction of power consumption when LUTs are implemented is the reduction of calculations in the filter. For a two-input LUT, the length of $\boldsymbol{H}$ is reduced to half of what it was. The number of coefficients that need to be added together is thus half of the original filter. However, the power is not reduced by 50% as the increased area increases leakage power, and some parts of the circuit, such as the processor, remain the same size.

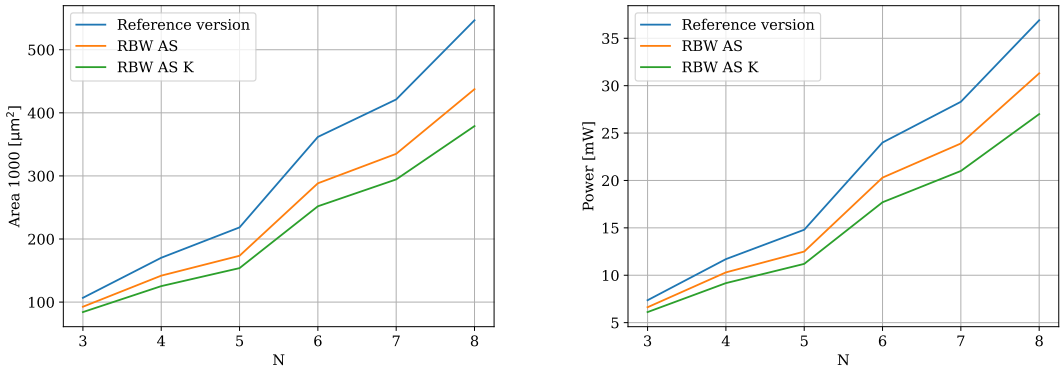
For $LUT\_SIZE = 3$, the power is larger than for $LUT\_SIZE = 2$ and $LUT\_SIZE = 4$. The reason for this is not clear. A small effect can come from the filter length not being dividable by the filter length. This is not optimal for power consumption, but it should not greatly impact it. The power reduction for $LUT\_SIZE = 3$ is between 21.7% and 36.5%, thus not far below having two- or four-input LUTs.

The leakage power is more significant for higher LUT_SIZE and contributes to the total power. However, the leakage power of the circuits is below 2% for all implementations. This means that a doubling of the area of the circuit, without more switching, would not increase the total power significantly. As explained in Section 4.4, the influence a larger area has on power consumption is smaller than it would be in a physical implementation, as another cell library most likely would be used.

Among the variations of LUT implementations, a LUT_SIZE of two is found favorable. While consuming slightly less power than the four-input LUT version and significantly less than the others, it offers a significant advantage in terms of area. The area is smaller than all other LUT versions and is thus the most attractive option for the control-bounded A/D converter's digital estimation filter.

4.6 Reduced Signal Widths

Reducing the bit width of coefficients and their corresponding adder modules, removing unused bits in registers and calculations, is shown to be an effective method for reducing both the area and power consumption of the circuitry. How much is saved compared to a similar version using 22 bits for all the shift registers and adders can be seen in Figure 4.9. The RBW AS has reduced bit width for each analog state, while the RBW AS K version has reduced bit width both for each analog state and for filter taps closer to the filter edges. The large area reduction comes from the fact that $\boldsymbol{H}$ is a large part of the design, as stated in Section 4.4. The adders are also a significant part of the total area of the accelerator. The bit width reduction is targeted at reducing those parts of the design.



(a) Area comparison.

(b) Power consumption comparison.

Figure 4.9: Optimization with reduced bit width (RBW) compared to the reference.

From Figure 4.9a, we see how a reduction of bits reduces the total cell area of the circuit. Reducing the bit width for each analog state reduces the power by 13.2% for $N = 3$. The reduction is higher for $N > 3$, and for $N = 8$, the area is reduced by 20%. The area is further reduced by reducing the bit width for filter taps near the edges of the filter as well. For $N = 3$, the area reduction is 21.2%, and for $N = 8$, 30.7%. The $N = 3$ implementation has the smallest decrease in area, while $N = 8$ has the largest decrease.

The power of the digital estimator with reduced bit width is displayed in Figure 4.9b. The power consumption is clearly reduced as the bit widths are reduced. For $N = 4$, the power of the reference

version is 11.70 mW. For the bit width reduction of analog state calculations, the power is reduced to 19.30 mW, a reduction of 12.0%. Also, reducing the bit width towards the edges of the filter reduces the power to 9.17 mW, a decrease of 21.6%. For $N = 8$, the corresponding percentages are 15.9% and 27.4%. For all Ns, the power reductions are between 9.1% and 16.7% for the first optimization and between 16.1% and 27.4% for the second one. The savings seem to be more prominent as N increases. This is because the added analog states have smaller coefficients and can thus be represented by smaller numbers. The savings are somewhat arbitrary, as the reductions of bits are very coarse. By fine-tuning the bit-reduction in more detail, the savings could be more significant, and the power reduction would be seen as more systematic.

The reason for the power reduction, even though only the unused part of the numbers are removed, is due to the use of two complements to represent the negative numbers. When adding two small positive numbers, only the bits used to represent the numbers are switched, and the unused upper bits stay 0 the whole time. But when adding a positive and a negative number, all the upper bits on the negative number are 1, and therefore all the unused bits are switched. This means that the power is saved by removing bits that are not necessary at all.

One problem with this optimization is that it is very individual for each filter configuration and is thus not flexible. There is no guarantee that it will work if the filter coefficients are changed. This can be solved by designing with a larger margin using extra bits, but that will lead to a larger area and power. This way of optimizing is most beneficial when designing the system for one specific case, but the principle can still be used even if the exact coefficients are not known at design time. The number of bits removed should then be a lot smaller, which leads to less reduction of area and power consumption.

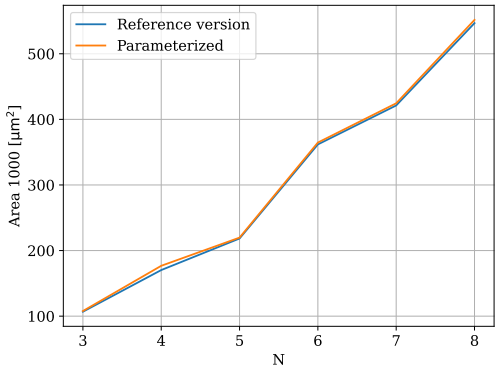
This specific optimization could not be directly converted into a real-world system. One of the benefits of the CBADC is the possibility of calibrating the system with new filter coefficients when the transistors in the AS change properties due to environmental changes and imperfect production. This will lead to changes in the sizes of the coefficients and a need for some margin in the shift registers and adders. The area and power consumption improvements will not be as large as in the synthesized version shown in Figure 4.9. In Chapter 5, some ways to develop this technique even further are presented. With those techniques combined, the area could be improved even more than what is shown in Figure 4.9.

4.7 Post-Synthesis Parameterizability

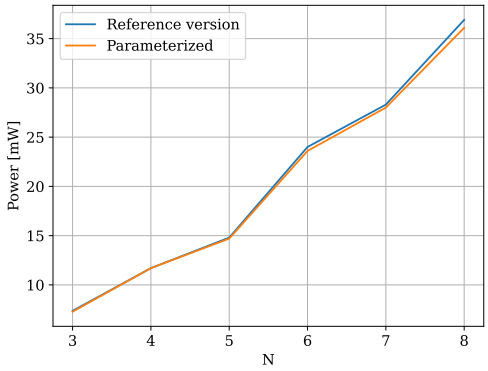
The main reason to explore a solution for a post-synthesis parameterizable system is to have one system usable with multiple SNR targets. To see how it impacts the area and power of the system, the parameterizable system is compared to the reference version with N and K equal to N_MAX and K_MAX; the maximum N and K for the post-synthesizable version. The area and power of the version capable of reaching the target SNR at 70 dB can be seen in Figure 4.10. The figure shows that the area increases insignificantly when $N = N\_MAX$ and $K = K\_MAX$ correspond to N and K of the reference version. The power is essentially the same for the two versions for the same configurations. This means there are no notable disadvantages regarding area and power by adding the possibility to change to a lower SNR configuration.

The synthesized system with $N\_MAX = 7$ and $K\_MAX = 448$ will be used for this comparison. When only N changes on this system, the SNR will change depending on what N becomes. As stated in Section 4.1, SNR will not increase by using a higher K for the given N and OSR combo than the minimum required. As shown in Figure 4.2 for $N = 4$ and OSR = 8, the SNR will not be higher than 46 dB. This means there is little reason to use the system with $N = 4$, OSR = 8, and $K = 448$. When the number of analog states used in the digital part is lowered, K should also be reduced not to do unnecessary calculations. As shown in Figure 4.11, it is also beneficial for power since the power decreases when K decreases in the same configuration.

Reducing only K to run a lower SNR configuration is also possible. When N equals N_MAX, the change in K will impact the SNR as shown in Appendix B. In Figure 4.11, the lower analog states have SNR about the same for all K. This is because the OSR is eight, and with such a low OSR,



(a) Area of parametrizable implementations with $N\_MAX = N$.



(b) Power consumption of parametrizable implementations with $N\_MAX = N$.

Figure 4.10: Comparison between reference version and parametrizable version.

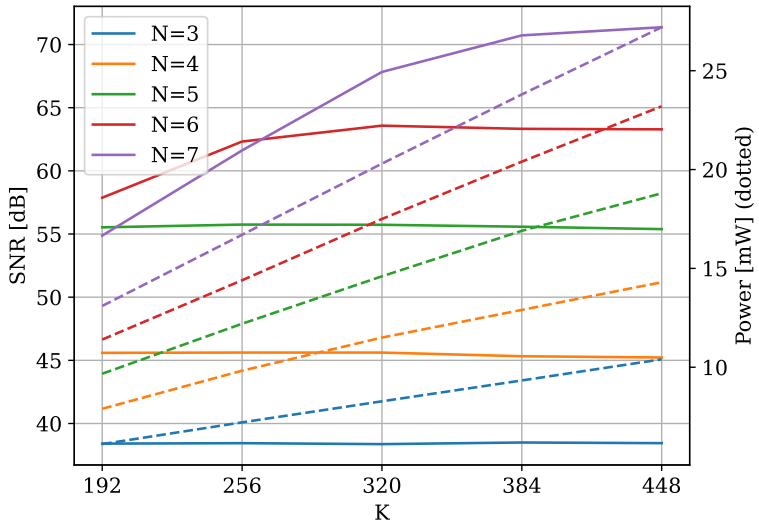


Figure 4.11: Power and SNR of $N\_MAX = 7$ and $K\_max = 448$.

the analog system will limit the performance of the ADC.

Figure 4.11 shows that if the required SNR is a couple of dB lower, the way to do that is by lower the K by some steps. But the more realistic is to lower the required SNR by a bit more. If the target is 60 dB, the figure shows that some configurations get a bit higher SNR than that. For example $N = 6$ with $K = 256$ and $N = 7$ with $K = 256$. These configurations get about the same SNR, but the power for the design with $N = 6$ is lower. That means it would be better to use the version with $N = 6$.

When lowering the target SNR the best way to use the least amount of power is to reduce the number of analog states used. It will lower the power more for the same SNR than just lowering K. To optimize it the most, K should be reduced as well. This correlation can be found for all the configurations when looking at the simulation results in Appendix E

Figure 4.11 show that the power goes down when less of the circuit is used. This effect is caused by a lower activity factor of the circuit when some parts are not switching. Since the clock network is a large part of the power consumption, the unused portions of the circuit are clock gated. When clock gating a circuit, it still consumes static power. One way to lower the static power is to use

power gating. Power gating a part of the circuit would reduce the supply voltage at that part to near zero. As both static and dynamic power is almost eliminated, power consumption of the power gated part becomes approximately zero.

The power given in this section is affected by the fact that the static and dynamic power is not balanced to the optimum. This will drastically change the result in this version, where more of the circuit is unused when running lower SNR configurations. When using clock gating and not power gating, the static power should be around the same for all the estimates for the same circuit. Appendix E show that the leakage power differs by 0.01 mW between the estimates with the same K_MAX and N_MAX, as expected. If a better voltage configuration is used to balance the static and dynamic power, the power saving shown in Figure 4.11 should be less significant as the leakage does not go down.

4.8 Combination of Optimizations

To make a better system, all the optimizations made in Section 4.5 4.6 and 4.7 can be combined. This means that the power and area reduction from reducing the number of bits used can be combined with the power reduction from using LUTs. To optimize the power the most, Section 4.5 shows that the optimal size of the LUT is 2. Section 4.7 also shows that parameterizability can be added to the system without significantly increasing area or power consumption.

If a system with $N = 4$ is used, the power reduction from reducing the bit width of some parts of the accelerator is 21.6%, and the reduction of power from implementing LUTs is 36.4%. If both optimizations are implemented and have the same power reduction contribution as found, the total power reduction could be up to 50.2% for $N = 4$. For the same configuration, the area would increase by 20.8%. The exact amount of power reduction and area increase for the system will vary from this depending on the configuration and how well the optimization techniques can be utilized together. Especially how many bits that are reduced will vary much depending on the system.

A disadvantage of combining the parameterizability with the reduction of bit width is that the reduction of bits is specific and different for each configuration. As mentioned in Section 4.6, it is possible to implement the optimization with some more headroom to reduce the likelihood of overflow. When combined, the area and power gain will be less than if the system is optimized for only running one configuration, but it will be more flexible.

If the optimization of the number of bits is combined with LUT, the area reduction could be significant, especially if the optimization of bits used in the registers and adders are separated. Since the adders need some more bits than the coefficient registers. The example shown in Figure 3.5 shows that the system needs four more bits than the coefficients need. When the LUT size increase, the number of adders decreases, and the size of $\boldsymbol{H}$ increases. If the bit width is reduced more on the registers than the adder module, the area of the versions with LUTs implemented could be reduced by more than the 35.9% found earlier.

4.9 Remove Low-Pass Filtering from Digital Estimator

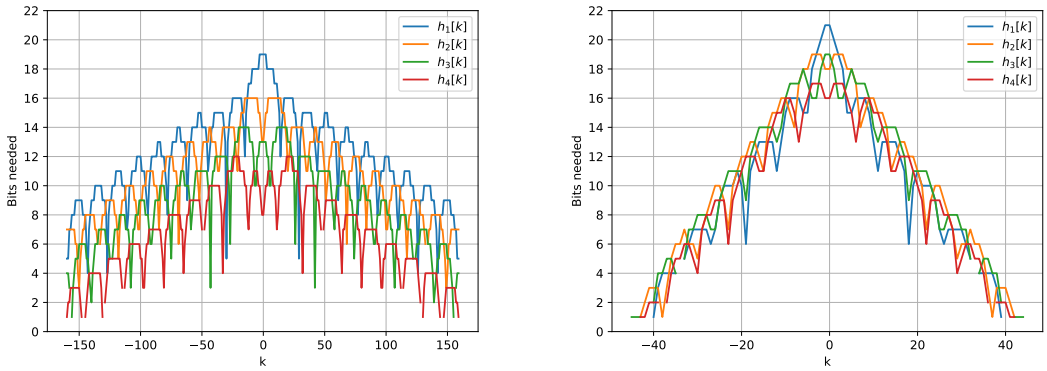
A different approach to implementing the digital estimator is to remove the low-pass (LP) filtering from the DE. The main reason to consider removing the LP filtering part of the DE is that the length of the filter can be significantly reduced. As shown in Section 4.4, the area and power decrease when the filter length is reduced.

The first issue discovered when testing the reference version without LP filtering is the fall in SNR as downsampling is implemented. This limitation arises because the noise contains frequencies higher than the bandwidth of the system, and because there are frequency components above the bandwidth, downsampling will result in aliasing. This means the DE must calculate OSR samples before the low-pass filter can downsample the signal. Thus, the number of calculations might

not be lowered, as multiple estimates must be calculated before downsampling them. Equation 2.1 provides insight into the scenario when no downsampling is utilized, indicating that the bandwidth is determined by $BW = F_s/2OSR$. Since the sampling frequency of the DE is the same as with downsampling, the solution will only fit applications with lower bandwidth. When adding an LP filtering stage after the DE, the signal can be downsampled there, but it will not help the sampling frequency of the DE, and the BW will still be limited.

Another weakness of the system described in this section is that the target SNR is not reached. This is caused by an unknown effect in the system that makes the DE limit the SNR to about 60 dB. This effect is not wanted and should be investigated further. Even though the tested system does not meet the target SNR, the results show a promising method of optimization if the issues can be resolved. The system without LP filtering is compared to the 70 dB SNR configurations found in Section 4.1 even though 70 dB is not reached.

Figure 4.12 shows an example of a system with $N = 4$ without LP filtering compared to a system with LP filtering. The system without filtering has only 45 coefficients in each direction that can be represented, whereas the system with LP filtering in the DE has 160 in each direction. The system with filtering is stopped at 160 since Section 4.1 shows that it does not need more than that to reach 70 dB SNR.



(a) With LP filtering

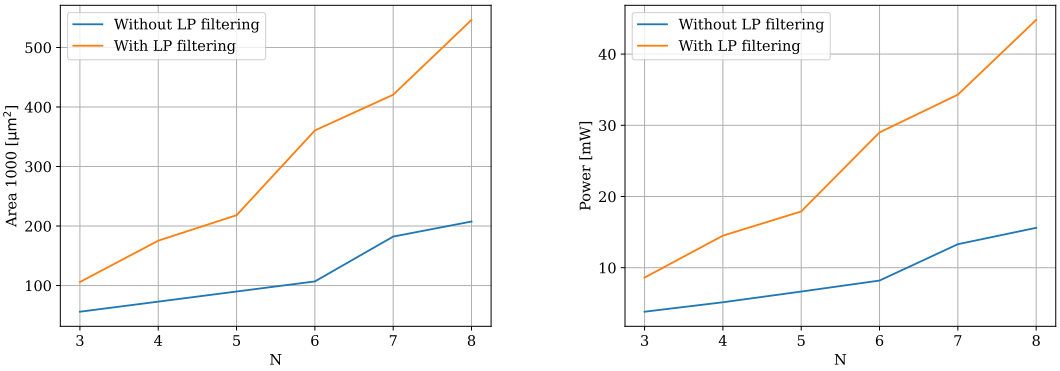
(b) Without LP filtering

Figure 4.12: Number of FPBs needed to store the filter coefficients with and without LP filtering.

Figure 4.12b shows that using $K1$ higher than 45 when 22 bits are used will only add coefficients too small to be represented. When making a system with lower $K1$, the area and thus power will decrease drastically. The adders are made to add blocks of 32 numbers together, which means $K1$ needs to be a multiple of 32. When $N = 4$, the lowest $K1$ can be, is 45. However, since the adders are made in blocks of 32, $K1$ at 64 is used. However, changing the modules to support another size of the blocks can better adapt it to certain implementations. From Appendix D, it can be seen that $K1$ needs to be a bit higher when N increases, and when N is above seven, more than 64 values are valid in each direction, which means $K1$ needs to be 96 to use all the coefficients that are large enough to be represented.

When $K1$ can be lowered as drastically as shown in Appendix D, the area and power should also be reduced drastically. When $K1$ is reduced from 128 to 64, the area of the accelerator should be almost halved. As stated in Section 4.4, the CPU area is not changed if the parameters change and is always 4519 µm^2. The total cell area is thus not expected to be entirely halved when halving $K1$. Figure 4.13a shows that the area is reduced by 47.1% compared to the reference system, when $N = 3$. The reduction is larger for higher N. This is because $K1$ is larger for all $N > 3$ for the reference version. The area can be seen in the figure to grow linearly with N when $K1$ is the same and got a jump when $K1$ increases from 64 to 96 between N equal to 6 and 7.

Figure 4.13b shows the power drawn from the system with and without the LP filtering. The figure shows that the power follows the same pattern as the area where it is directly correlated to the rise in N and $K1$. This is the same trend that is shown in the other versions. The reference system



(a) area

(b) power

Figure 4.13: Area and power with and without LP filtering in DE.

used with LP filtering is a version without downsampling and thus is the power not precisely the same as the reference version with downsampling. The trend where the power follows the area and the area follows the number of coefficients used means that it is beneficial to change the shape of the filter to use as few coefficients as possible. In this section, it is presented a solution that reduces the power drastically, but it does not reach the target SNR. The optimal solution is to use a filter that uses the least amount of coefficients that still fulfill the SNR requirements.

Chapter 5

Future Work

5.1 Reduction of Area and Power Consumption

In Section 3.5.1, a method of reducing the area and power of the digital estimator using fewer bits to store filter coefficients and reducing the size of the adder modules is described. To further improve the method, the number of register widths of the filter coefficients can be further decreased. The register does not have the same need for headroom as the adder module, as the adder module must be large enough for the sum of multiple coefficients. In the earlier proposed solution, the numbers of bits used in the filter coefficient's registers are the same width as in the adders. The size of the whole adder module must stay the same as in the already implemented configuration to prevent overflow during addition. Because the adder bit width is larger than the coefficients, the coefficients must be sign-extended before additions, some extra logic is necessary for each adder. Therefore, a solution to reduce the area even more without changing the output signal is to use fewer bits for the coefficients than the associated adders.

In Section 4.9, estimates are calculated without LP filtering in the DE. The sections show that there is much power to save by changing the shape of the coefficients in that way. The problem with that solution is the reduced SNR. It is possible to explore this change in coefficients in another way that can have at least some of the power gains and still get the wanted SNR. This is by changing the filter type and shape, instead of removing it. Malmberg have developed a new version of the Python toolbox which make it possible to make the DE with different filter types. This makes it possible to test different filter shapes and find out if they need lower filter depth to get the same SNR. If a better solution is found, power and area can be reduced without changing the design of the accelerator.

It is possible to explore the possibility of removing even more bits of the additions and filter coefficients than shown in Section 4.6 by removing bits of the useful data. This will impact the SNR of the DE, but if it is small enough changes, it could be used. The thought is that the coefficients for the lowest analog states and the closes to the middle are the ones that have the largest impact. If some bits in the outermost values for the high analog states are changed, the idea is that such a small change will not significantly impact the result. This could possibly be easier to do with the version of the Python toolbox that can change the shape of the filer.

As explained in Section 2.4 and Section 4.3, the power can be drastically reduced by reducing the supply voltage level. To maintain performance, the threshold voltage must be lowered as well. Using another library capable of lowering the threshold voltage and supply voltage would be crucial when implementing the digital estimator on a physical chip. The cell library used can be used with a lower threshold voltage, but without the opportunity to reduce the supply voltage the static power will not be reduced. The dynamic power, however, will increase significantly.

5.2 Sampling Frequency and Bandwith

The target for sampling frequency is in this thesis set to be 20 MHz. Some applications would require even larger sampling frequencies than that. With the system proposed in this thesis, this can be done in multiple ways. One way to produce samples at a higher rate is to run the system on a higher clock frequency. This will change the synthesis, so the area will be affected as well. Section 2.3 says that a higher frequency will give higher switching power. This optimization is also limited by the maximum clock frequency that is possible.

The current system design's limiting factor for the sampling frequency is the CPU's ability to read and write data to and from the accelerator module. One possible way to increase the speed of these operations is to use a faster CPU. However, this would require developing a new communication mechanism between the CPU and the accelerator, as the current PCPI is specific to the Picorv32 CPU. With a faster CPU, operations could be performed more quickly, and memory reads could also be optimized using a prefetcher.

The CPU does not have to perform its operations faster to increase the speed of the DE filtering. By, for example, using a 64-bit processor, the amount of data received per fetch instruction would be doubled. By doubling the registers connected to the accelerator module, the transfer rate would double at the same frequencies.

Another optimization approach to improve the system's sampling rate is to remove the CPU's involvement in memory access and delegate it entirely to the accelerator using the Direct Memory Access (DMA) technique. This technique transfers most of the CPU's workload to the accelerator. This is a more traditional way of implementing an accelerator, where the CPU and accelerator can communicate over a shared bus. The CPU will still be used to control the accelerator and can be used for tasks like changing parameters and starting and stopping the filter computations. When no changes must be made to the estimation filter, the processor is free to do other tasks.

When the accelerator itself is responsible for fetching data, the data can be stored differently. If the samples of the control sequence are stored in a FIFO (First In, First Out) array, the reading process for the accelerator could be simple, with no memory indexing. This type of memory will have a low latency.

Another way to get a higher sampling frequency is to use a CPU core with wider registers. By implementing a 64-bit core, data from the memory could be fetched in less time. The Picorv32 fetches a 32-bit number for each memory request. With a 64-bit core, and thus a 64-bit bus, all the control signal bits, $\boldsymbol{s}[k]$, needed for one calculation can be fetched in a single transmission on the bus. Another possibility with a 64-bit core is to increase the OSR and DSR without reducing the sampling frequency.

Chapter 6

Conclusion

In this thesis, the digital estimator of a single input CBADC has been implemented using a RISC-V CPU with a custom-made accelerator. The area and power consumption of configurations with 3 - 8 analog states have been compared to each other with the target SNR of 70 dB. Multiple solutions on how to optimize the area and power consumption of the accelerator are presented.

A reference version of the system has been described, and optimizations have been made to improve its power consumption. All implementations are described in the hardware description language SystemVerilog, with the CPU and accelerator being synthesized while excluding memory from the area and power consumption estimates. The reference version of the accelerator with 4 analog states is estimated to use 11.70 mW with an area of 170 255 µm^2.

The results show that the circuitry's area and power consumption are closely related to the number of analog states and the filter length. Notably, a lower number of analog states necessitates fewer filter coefficients to achieve the desired SNR, resulting in significantly reduced power consumption for such configurations.

By exploiting the trends in filter coefficient sizes based on their analog state and filter tap, register sizes and module bit widths can be reduced significantly. The power savings seems to get more prominent for a higher number of analog states, ranging between 16.1% and 27.4% for the implemented filters, while area reductions were between 21.2% and 30.7%. For 4 analog states, the estimated power consumption and area were reduced to 9.17 mW and 125 300 µm^2.

By precalculating combinations of coefficients and storing them in registers, the power consumption of the circuitry can be reduced. A variation of LUTs with different input sizes is implemented, and a two-input LUT is found to be the most efficient, resulting in power consumption reductions between 35% and 40.1%. However, this optimization led to an increase in area by up to 61.9%. The power consumption and area were thus reduced to 7.44 mW 279 102 µm^2.

The impact post-synthesis parametrization has on the circuit is found as well. By enabling the possibility to change the number of analog states or filter length at runtime, the power consumption can be reduced at the cost of reduced SNR. For a version where the number of analog states and the filter length is the maximum allowed by the design, the area and power consumption are approximately the same as for a non-post-synthesis parameterizable filter.

Furthermore, the impact of the removal of low-pass filtering from the digital estimator is investigated. This optimization reduced the required filter length, consequently lowering both the area and power consumption. However, downsampling in the digital estimator introduced aliasing, making it incompatible with the same bandwidth requirements Although this optimization did not meet the target SNR, it presents an intriguing concept for further development.

The 28 nm FDSOI technology node has a significantly lower leakage power consumption than dynamic power consumption. The increase or reduction of the area thus has a low impact on the total power consumption of the circuit. By using a technology capable of a lower threshold voltage

and supply voltage, the results would be different. The large area increase of the LUT versions would then result in a significant increase in power consumption compared to what is found in this thesis. However, lower supply voltage reduces the dynamic power consumption.

Bibliography

[1] Neil H. E. Weste and David Money Harris. *CMOS VLSI design: a circuits and systems perspective*. Addison Wesley, Boston, 4th ed edition, 2011. ISBN 978-0-321-54774-3. OCLC: ocn473447233.

[2] Hampus Malmberg. *Control-Bounded Converters*. PhD thesis, ETH Zurich, 2020. URL http://hdl.handle.net/20.500.11850/469192. Artwork Size: 260 p. Medium: application/pdf Pages: 260 p.

[3] Andrew Waterman, Krste Asanovic, and CS Division. *The RISC-V Instruction Set Manual*, volume I: User-Level ISA. RISC-V Foundation, December 2019.

[4] PicoRV32 - A Size-Optimized RISC-V CPU, May 2023. URL https://github.com/YosysHQ/picorv32. original-date: 2015-06-06T11:52:27Z.

[5] Hans-Andrea Loeliger, Lukas Bolliger, Georg Wilckens, and Jonas Biveroni. Analog-to-digital conversion using unstable filters. *IEEE*, page 4, 2011. doi: 10.1109/ITA.2011.5743620.

[6] Hans-Andrea Loeliger and Georg Wilckens. Control-based analog-to-digital conversion without sampling and quantization. In *2015 Information Theory and Applications Workshop (ITA)*, pages 119–122, San Diego, CA, USA, February 2015. IEEE. ISBN 978-1-4799-7195-4. doi: 10.1109/ITA.2015.7308975. URL http://ieeexplore.ieee.org/document/7308975/.

[7] Fredrik Feyling, Hampus Malmberg, Carsten Wulff, Hans-Andrea Loeliger, and Trond Ytterdal. High-level Comparison of Control-Bounded A/D Converters and Continuous-Time Sigma-Delta Modulators. In *2022 IEEE Nordic Circuits and Systems Conference (NorCAS)*, pages 1–5, Oslo, Norway, October 2022. IEEE. ISBN 9798350345506. doi: 10.1109/NorCAS57515.2022.9934426. URL https://ieeexplore.ieee.org/document/9934426/.

[8] Fredrik Esp Feyling. *Design Considerations for a LowPower Control-Bounded A/D Converter*. PhD thesis, NTNU, 2021.

[9] Andreas Bjørsvik and Sevat Mestvedthagen. *Optimization of CBADC estimation filter algorithms for RISC-V implementations*. PhD thesis, NTNU, 2022.

[10] David André Bjerkan Mikkelsen. *A study of control-bounded estimation filter architectures*. PhD thesis, NTNU, 2022.

[11] Roger Woods, John McAllister, Gaye Lightbody, and Ying Yi. *FPGA-based Implementation of Signal Processing Systems*. Wiley, 2008. ISBN 978-0-470-03009-7. URL https://d1.amobbs.com/bbs_upload782111/files_30/ourdev_564364C2IZCF.pdf.

[12] Peter Marwedel. Optimization. In *Embedded System Design*, pages 349–379. Springer International Publishing, Cham, 2021. ISBN 978-3-030-60909-2 978-3-030-60910-8. doi: 10.1007/978-3-030-60910-8_7. URL http://link.springer.com/10.1007/978-3-030-60910-8_7. Series Title: Embedded Systems.

[13] Xinpeng Xing, Peng Zhu, and Georges Gielen. *Design of Power-Efficient Highly Digital Analog-to-Digital Converters for Next-Generation Wireless Communication Systems*. Signals and Communication Technology. Springer International Publishing, Cham, 2018. ISBN 978-3-319-66564-1 978-3-319-66565-8. doi: 10.1007/978-3-319-66565-8. URL http://link.springer.com/10.1007/978-3-319-66565-8.

[14] Wai-Kai Chen, editor. *The electrical engineering handbook*. Elsevier Academic Press, Amsterdam ; Boston, 2005. ISBN 978-0-12-170960-0. OCLC: ocm56953259.

[15] Biagio Peccerillo, Mirco Mannino, Andrea Mondelli, and Sandro Bartolini. A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives | Elsevier Enhanced Reader, 2022. URL https://www.sciencedirect.com/science/article/pii/S1383762122001138.

[16] Wajahat Qadeer, Rehan Hameed, Ofer Shacham, Preethi Venkatesan, Christos Kozyrakis, and Mark Horowitz. Convolution engine: balancing efficiency and flexibility in specialized computing. *Communications of the ACM*, 58(4):85–93, March 2015. ISSN 0001-0782, 1557-7317. doi: 10.1145/2735841. URL https://dl.acm.org/doi/10.1145/2735841.

[17] RISC-V foundation. History, 2021. URL https://riscv.org/about/history/.

[18] John L. Hennessy. *Computer architecture: a quantitative approach*. Morgan Kaufmann Publishers, Cambridge, MA, sixth edition edition, 2019. ISBN 978-0-12-811905-1.

[19] Hampus Malmberg. Control-Bounded A/D Conversion Toolbox Documentation, 2021. URL https://cbadc.readthedocs.io/en/latest/control-bounded_converters.html.

[20] Riscv-tools, April 2019. URL https://github.com/riscv-software-src/riscv-tools.

[21] Cadence. Xcelium Logic Simulator, 2023. URL https://www.cadence.com/en_US/home/tools/system-design-and-verification/simulation-and-testbench-verification/xcelium-simulator.html.

[22] Cadence. SimVision for Debugging Mixed-Signal Simulations, 2023. URL https://www.cadence.com/en_US/home/training/all-courses/86301.html.

[23] Synopsys. Design Compiler, 2023. URL https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html.

[24] Synopsys. Gold Standard in Static Timing Analysis - PrimeTime, 2023. URL https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html.

[25] Dennis Agyemanh Nana Gookyi and Kwangki Ryoo. Selecting a Synthesizable RISC-V Processor Core for Low-cost Hardware Devices. In *Journal of Information Processing Systems*, 2019. doi: 10.3745/JIPS.03.0129.

[26] Islam Elsadek and Eslam Yahya Tawfik. RISC-V Resource-Constrained Cores: A Survey and Energy Comparison. In *2021 19th IEEE International New Circuits and Systems Conference (NEWCAS)*, Toulon, France, June 2021. IEEE. ISBN 978-1-66542-429-5. doi: 10.1109/NEWCAS50681.2021.9462781. URL https://ieeexplore.ieee.org/document/9462781/.

[27] Andreas Bjørsvik and Sevat Mestvedthagen. Github, June 2023. URL https://github.com/Bjorsvik98/Digital_Estimator_CBADC.

Appendix A

Custom Instructions

To instruct the CPU when to use the accelerator, custom instructions are made. These are made in the standard RISC-V format presented in [3] and are added to the toolchain to make the compiler understand when to use them. For simplicity, all instructions that are added are made in the R format which groups the bits as shown in Figure A.1. The instruction uses two input registers, one output register, and no immediate values. It could be possible to optimize the instructions by using immediate values for the instruction for loading in the filter coefficient.

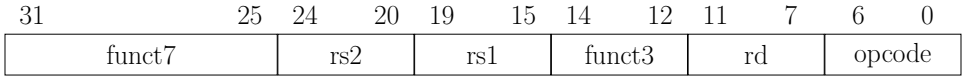


Figure A.1: The format of the RISC-V R instructions [3].

To separate the custom instruction from those already implemented in the instruction set, some bits of the opcode must be changed. When doing that it is important to understand the encoding rules, and thus avoid choosing some combinations used by other instructions. The instructions presented here are only made not to conflict with the RV32E format which is used for this thesis. No guarantee can be granted that the instruction will not conflict with other extensions. The two most important rules to consider are that all 32-bit instructions have their two least significant bits set to 11 and bits 2-4 should never be 111.

The four instructions that are used are *changevar*, *calculate*, *loadh*, and *loads*. *Changevar* is used to load in constant values such as N and K. Input register 1 ($rs1$) is used to load the value, and input register 2 ($rs2$) is used to determine which constant that is loaded. The *calculate* instruction takes in values from the $\boldsymbol{S}$ matrix in $rs1$ and $rs2$, and it is the only one that uses the return register (rd) to receive the calculated value. The last two instructions load values of $\boldsymbol{H}$ and $\boldsymbol{S}$ respectively into their registers in the accelerator. The $\boldsymbol{H}$ matrix is loaded one coefficient at a time, while the $\boldsymbol{S}$ matrix is loaded in multiple matrix elements at a time. For each *loads* command, DSR $\cdot$ AS number of bits are loaded. The bit patterns for the instructions can be seen in Table A.1.

Table A.1: Decoding patterns for the instructions.

Instruction name	funct3	opcode	32-bit match [hex]
changevar	110	0100111	00006027
calculate	010	0100111	00002027
loadh	011	0100111	00003027
loads	100	0100111	00004027

Appendix B

Finding Filter Length

Simulated SNR of the complete CBADC system with varying $K1$ and OSR are seen in Figure B.1 to Figure B.6. The figures show how oversampling ratio and filter depth affect the SNR of the system. This is used to choose parameters OSR and $K1$ that give an SNR of above 70 for the implemented versions.

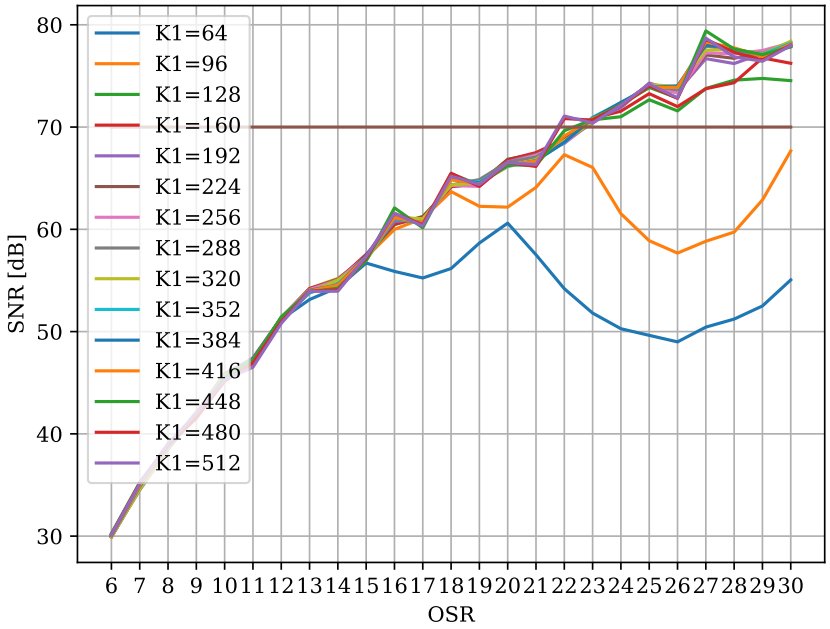


Figure B.1: SNR VS OSR to find $K1$ for N=3.

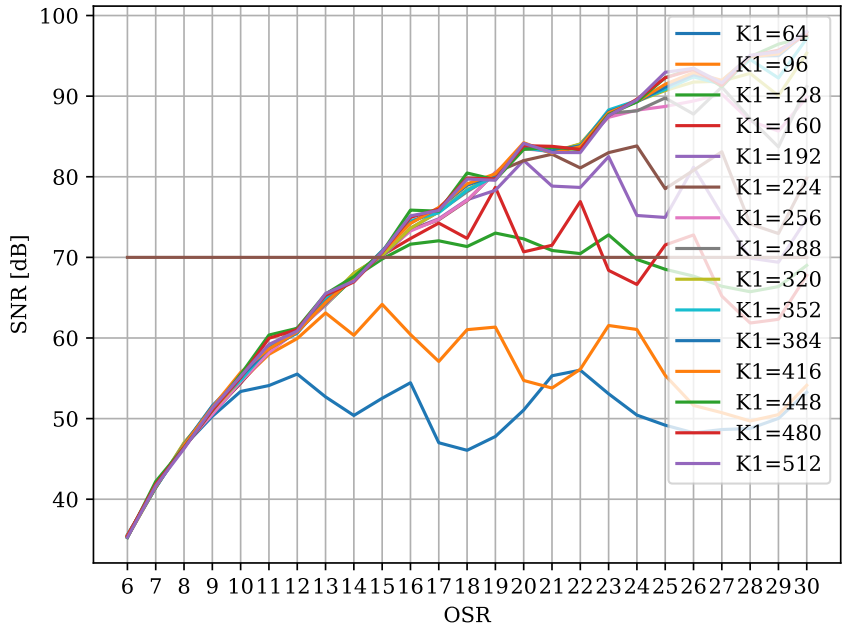


Figure B.2: SNR VS OSR to find $K1$ for N=4.

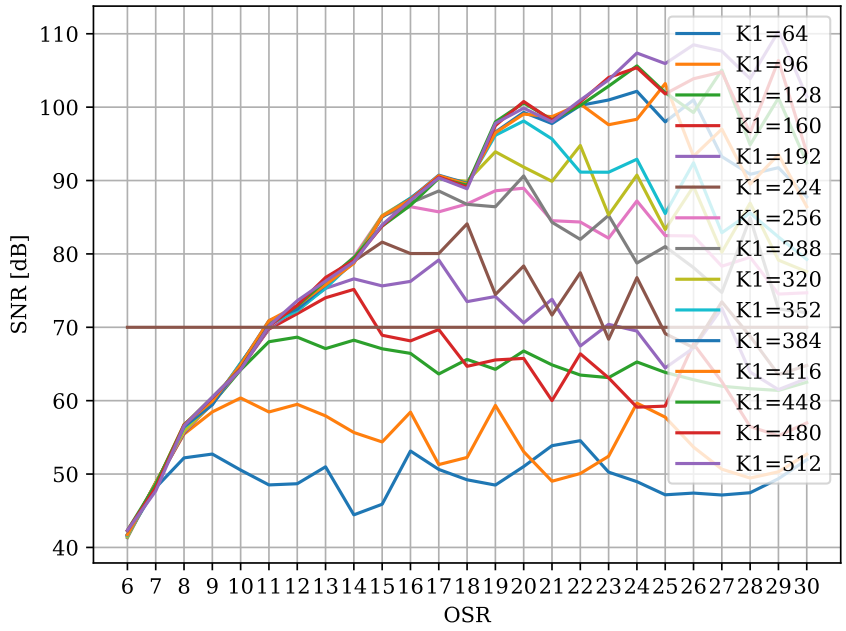


Figure B.3: SNR VS OSR to find $K1$ for N=5.

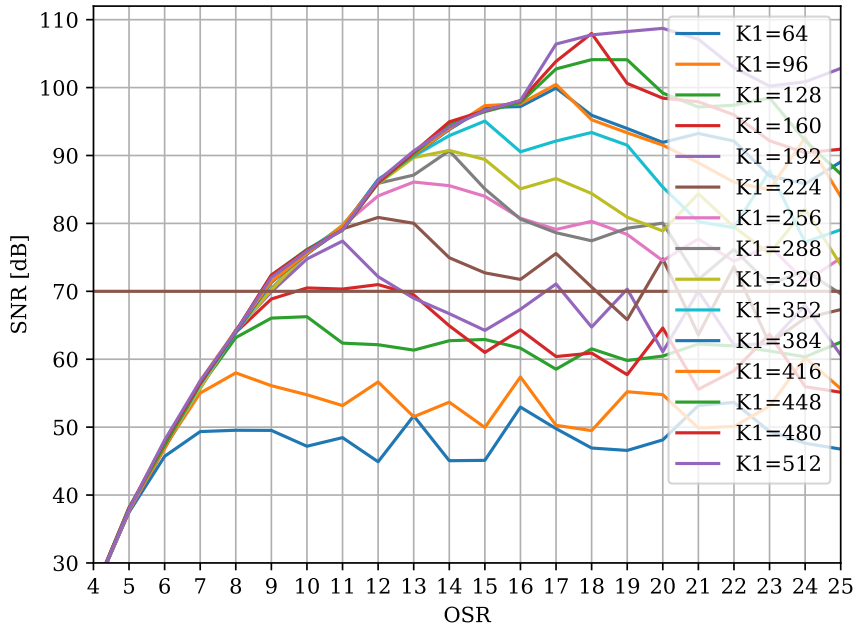


Figure B.4: SNR VS OSR to find $K1$ for N=6.

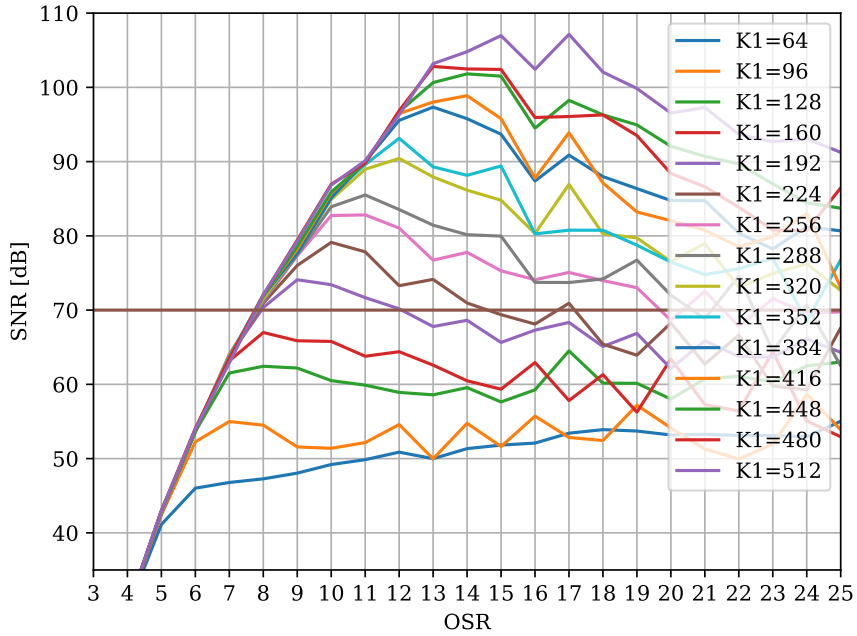


Figure B.5: SNR VS OSR to find $K1$ for N=7.

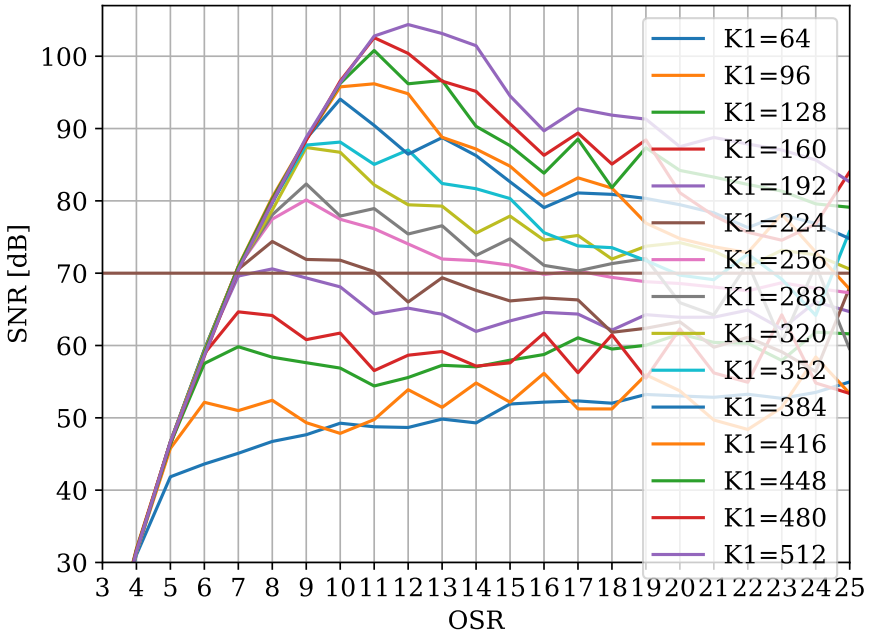


Figure B.6: SNR VS OSR to find $K1$ for N=8.

Appendix C

Finding FPB

When investigating the number of bits needed to be used in the calculation, it is checked when the SNR is impacted. The goal is to find the lowest number of bits that can be used where the SNR is the same as if 32 bits are used. This is checked for each N individually for the OSR that gives SNR at 70 dB. All the results can be seen in Figure C.1 to Figure C.6. The dotted lines show the SNR for the configuration with FPB=32, which is the SNR the solid line should be the same as.

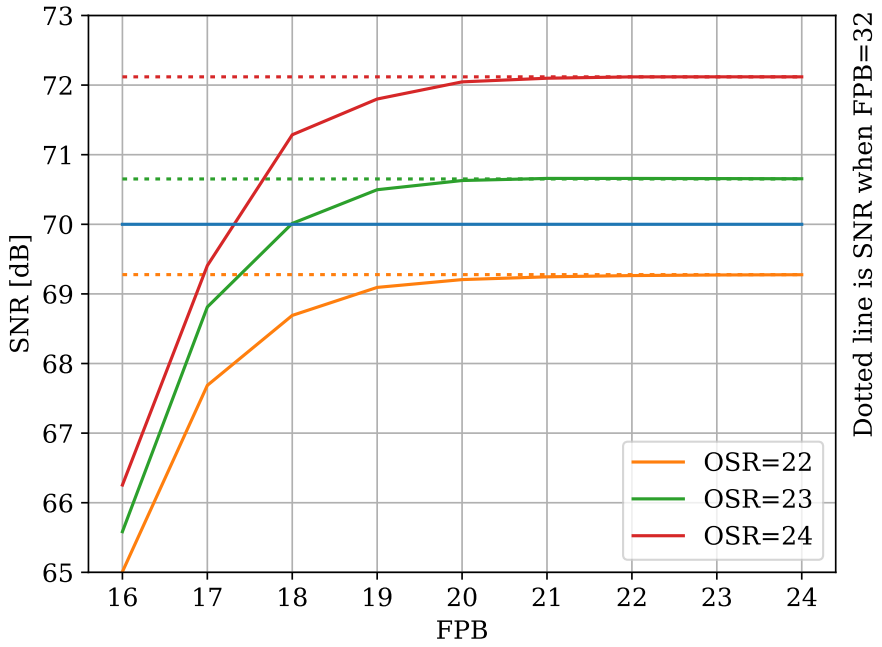


Figure C.1: SNR VS OSR to find FPB for N=3.

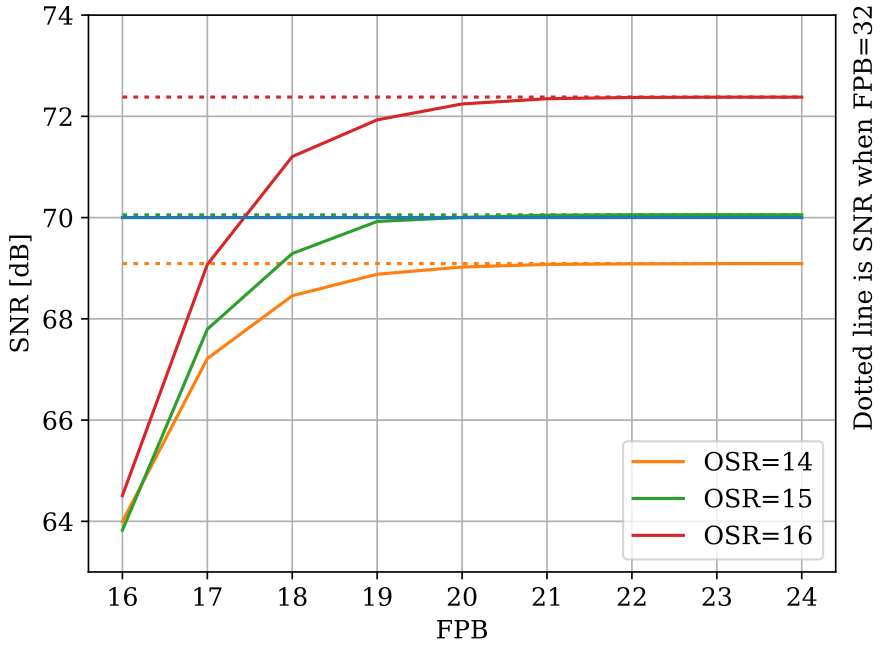


Figure C.2: SNR VS OSR to find FPB for N=4.

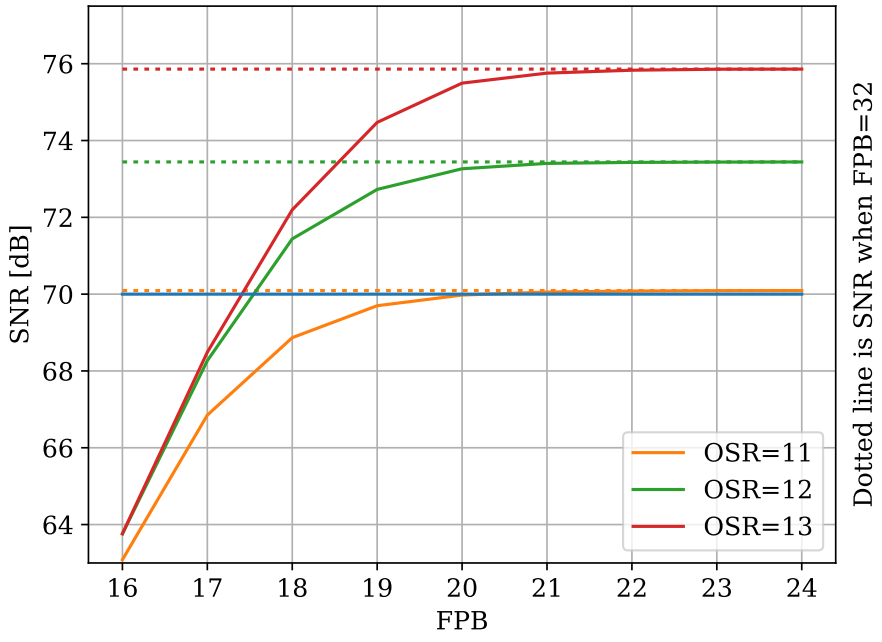


Figure C.3: SNR VS OSR to find FPB for N=5.

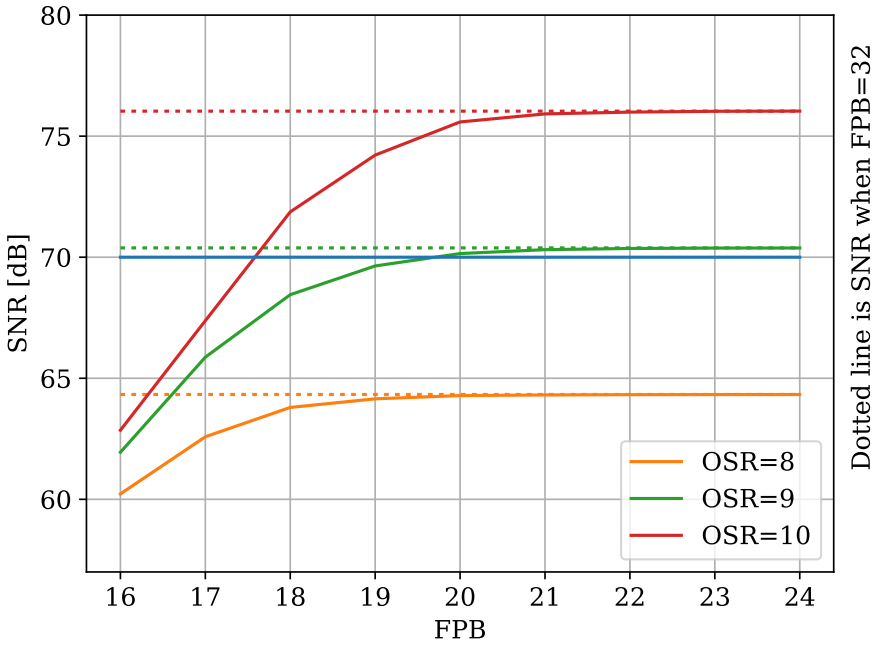


Figure C.4: SNR VS OSR to find FPB for N=6.

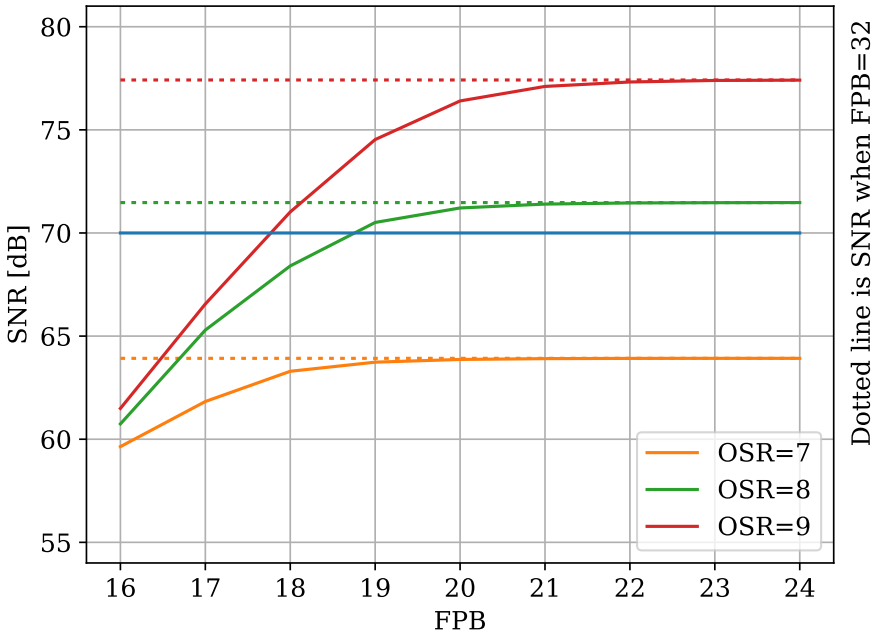


Figure C.5: SNR VS OSR to find FPB for N=7.

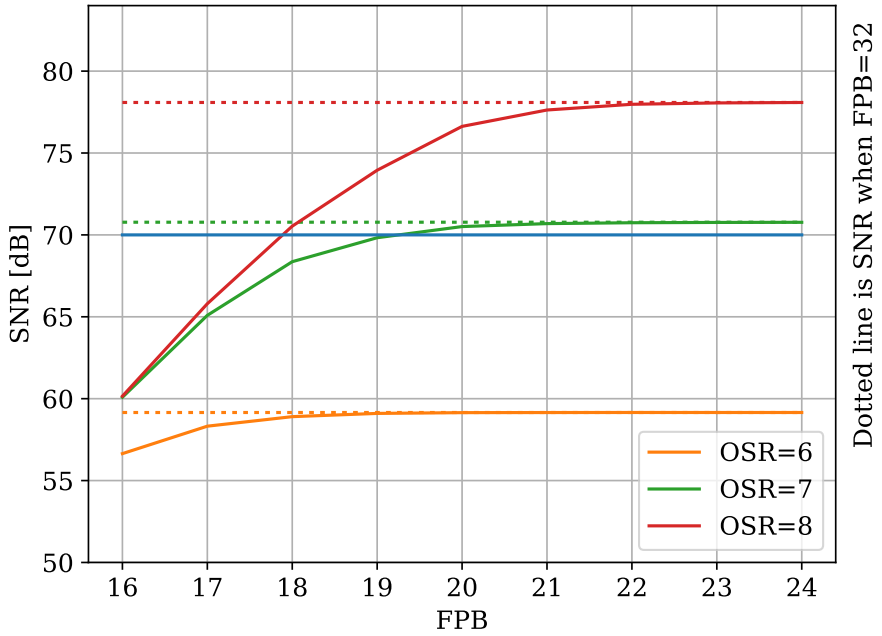


Figure C.6: SNR VS OSR to find FPB for N=8.

Appendix D

Filter Coefficients Without Low-Pass Filtering

When testing the system without doing the LP filtering in the DE, the shape of the coefficient plot will be much steeper than Figure 3.5. The bits needed to represent coefficients for all the different systems can be seen in Figure D.1 to Figure D.6 The figures show that when N is larger than seven, there must be more than 64 coefficients in each direction, and 64 is enough when N is between three and six.

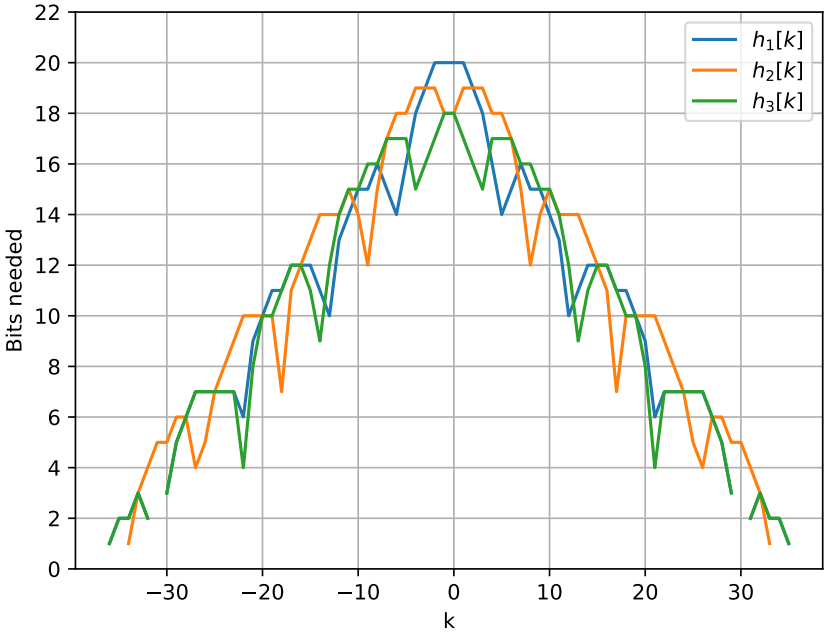


Figure D.1: No filtering N3.

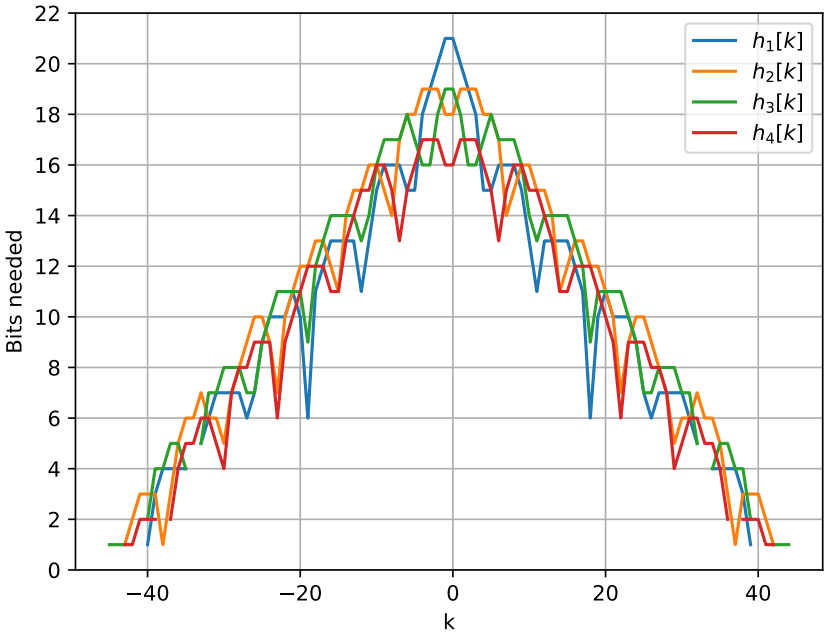


Figure D.2: No filtering N4.

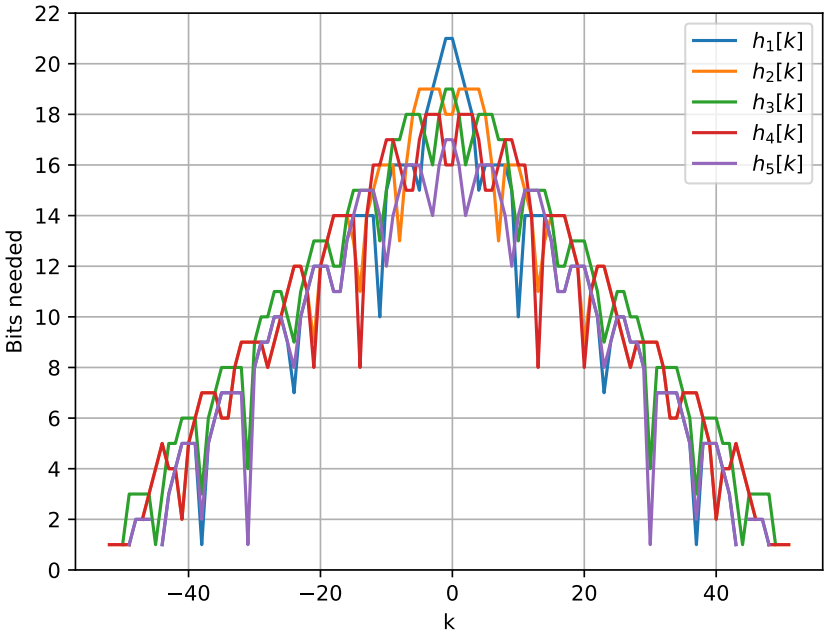


Figure D.3: No filtering N5.

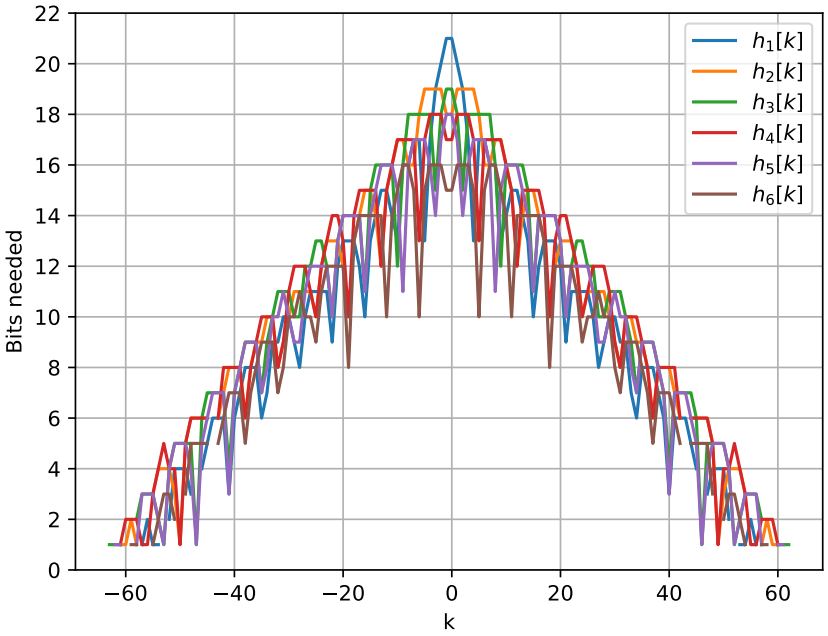


Figure D.4: No filtering N6.

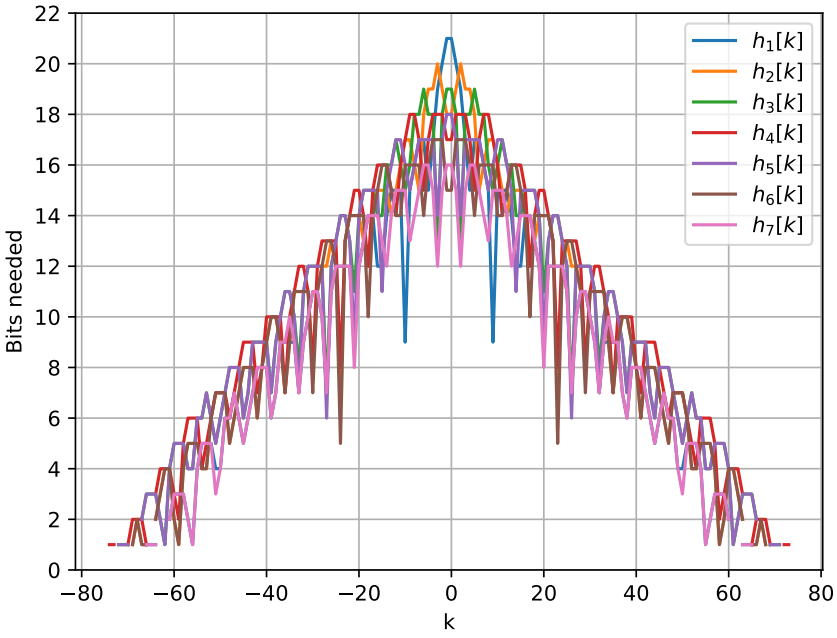


Figure D.5: No filtering N7.

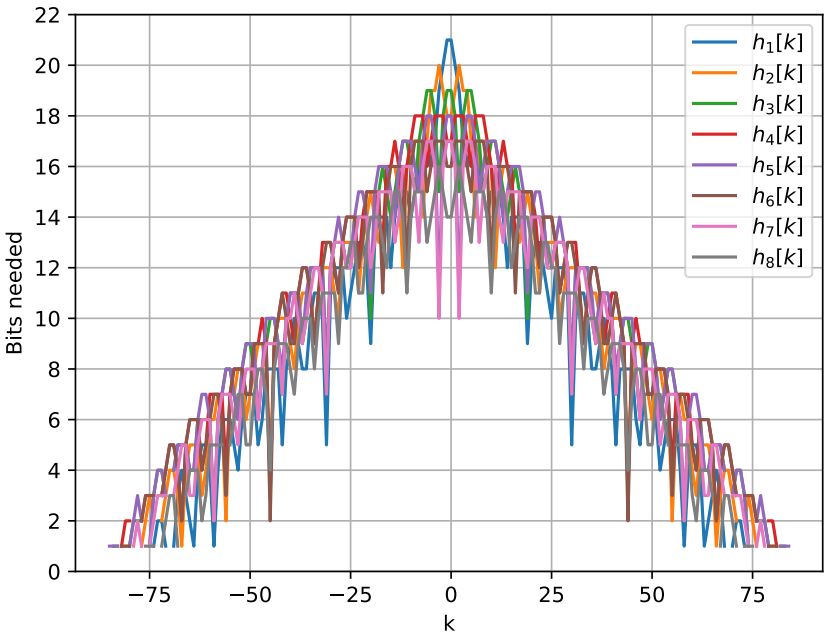


Figure D.6: No filtering N8.

Appendix E

Result Table

The results of the synthesis, simulation, and power estimate for all the different configurations on all the different versions are shown in Table E.1.

Table E.1: Results from all simulations.

Version	N	K	Total area [µm^2]	Switching power [mW]	Internal power [mW]	Leakage power [mW]	Total power [mW]	SNR [dB]
Reference	3	256	106757	2.47	4.86	0.03	7.36	66.61
Reference	4	320	170255	3.70	7.97	0.05	11.70	69.41
Reference	5	320	218391	5.01	9.76	0.07	14.80	73.40
Reference	6	448	361795	8.05	15.90	0.11	24.00	68.88
Reference	7	448	421191	9.43	18.70	0.13	28.30	71.37
Reference	8	512	546699	12.20	24.50	0.17	36.90	69.36
Reference no downsampling	3	256	105883	2.91	5.67	0.03	8.61	66.85
Reference no downsampling	4	320	175361	4.84	9.56	0.05	14.50	70.27
Reference no downsampling	5	320	218124	6.04	11.80	0.07	17.90	71.86
Reference no downsampling	6	448	360564	9.71	19.20	0.11	29.00	70.25
Reference no downsampling	7	448	420600	11.40	22.80	0.13	34.30	70.76
Reference no downsampling	8	512	546325	14.80	29.80	0.17	44.80	70.26

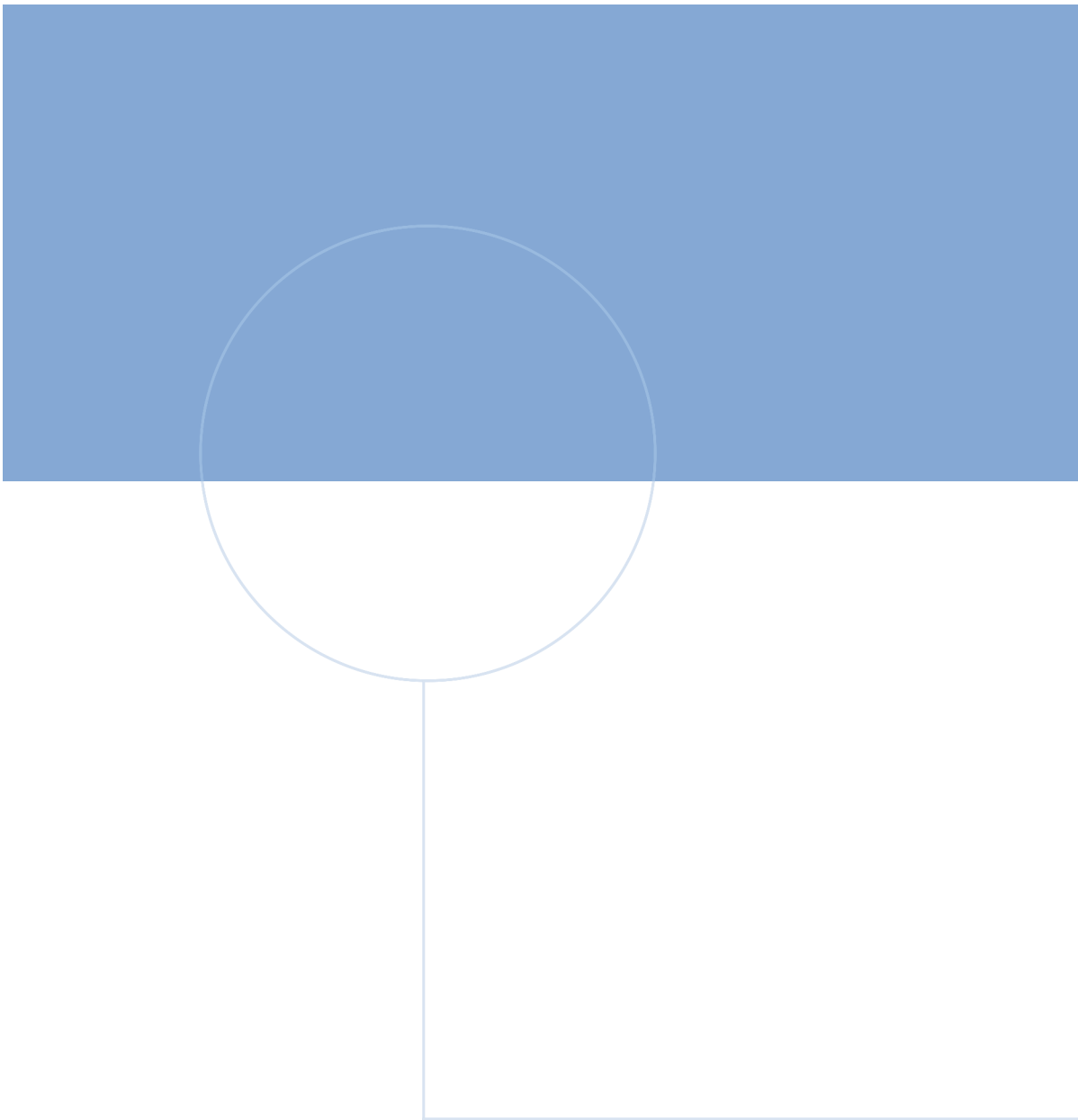
Version	N	K	Total area [µm^2]	Switching power [mW]	Internal power [mW]	Leakage power [mW]	Total power [mW]	SNR [dB]
LUT 1	3	256	180738	2.76	5.09	0.06	7.91	66.61
LUT 1	4	320	298786	4.49	8.26	0.09	12.80	69.41
LUT 1	5	320	372197	5.57	10.10	0.12	15.80	73.40
LUT 1	6	448	620520	8.92	16.30	0.19	25.40	68.88
LUT 1	7	448	721665	10.50	19.30	0.22	30.00	71.37
LUT 1	8	512	940737	13.60	25.30	0.29	39.20	69.36
LUT 2	3	256	169539	1.61	3.12	0.05	4.78	66.61
LUT 2	4	320	279102	2.67	4.68	0.09	7.44	69.41
LUT 2	5	320	347550	3.31	5.75	0.11	9.16	73.40
LUT 2	6	448	579688	5.24	8.95	0.18	14.40	68.88
LUT 2	7	448	675595	6.20	10.60	0.21	17.00	71.37
LUT 2	8	512	881306	8.01	13.80	0.27	22.10	69.36
LUT 3	3	256	189748	1.87	3.19	0.06	5.12	66.61
LUT 3	4	320	308980	3.04	4.68	0.09	7.81	69.41
LUT 3	5	320	385037	3.78	5.72	0.11	9.61	73.40
LUT 3	6	448	651543	6.39	9.39	0.20	16.00	68.88
LUT 3	7	448	759908	7.49	10.90	0.23	18.70	71.37
LUT 3	8	512	983714	9.42	13.70	0.30	23.40	69.36
LUT 4	3	256	256505	1.81	2.95	0.08	4.84	66.61
LUT 4	4	320	423348	3.06	4.47	0.13	7.67	69.41
LUT 4	5	320	528105	3.79	5.51	0.17	9.46	73.40
LUT 4	6	448	883852	6.12	8.55	0.28	14.90	68.88
LUT 4	7	448	1030405	7.27	10.20	0.32	17.80	71.37
LUT 4	8	512	1344595	9.44	13.20	0.42	23.10	69.36
Without LP filter	3	128	55969	1.29	2.51	0.02	3.82	56.17
Without LP filter	4	128	72911	1.75	3.37	0.02	5.15	56.51
Without LP filter	5	128	89920	2.28	4.34	0.03	6.65	52.34
Without LP filter	6	128	106877	2.80	5.37	0.03	8.19	72.80
Without LP filter	7	192	182226	4.60	8.67	0.05	13.30	63.41
Without LP filter	8	192	207534	5.39	10.20	0.06	15.60	57.39

Version	N	K	Total area [µm^2]	Switching power [mW]	Internal power [mW]	Leakage power [mW]	Total power [mW]	SNR [dB]
RBW AS	3	256	92622	2.19	4.40	0.03	6.62	66.61
RBW AS	4	320	141900	3.35	6.89	0.04	10.30	69.41
RBW AS	5	320	173448	4.11	8.38	0.05	12.50	73.40
RBW AS	6	448	288277	6.62	13.60	0.09	20.30	68.88
RBW AS	7	448	335015	7.75	16.00	0.10	23.90	71.37
RBW AS	8	512	437484	10.10	21.10	0.14	31.30	69.36
RBW AS K	3	256	84159	1.97	4.12	0.03	6.11	66.61
RBW AS K	4	320	125300	2.92	6.21	0.04	9.17	69.41
RBW AS K	5	320	153873	3.58	7.55	0.05	11.20	73.40
RBW AS K	6	448	251846	5.64	12.00	0.08	17.70	68.88
RBW AS K	7	448	294427	6.63	14.30	0.09	21.00	71.37
RBW AS K	8	512	378970	8.45	18.40	0.12	27.00	69.36

Version	Max N	Max K	N	K	Total area [µm^2]	Switching power [mW]	Internal power [mW]	Leakage power [mW]	Total power [mW]	SNR u_hat [dB]
Parametrizable	3	256	3	192	107821	1.95	3.84	0.03	5.83	62.53
Parametrizable	3	256	3	256	107821	2.48	4.77	0.03	7.28	66.61
Parametrizable	4	320	3	192	176750	2.00	3.97	0.05	6.02	57.45
Parametrizable	4	320	3	256	176750	2.52	4.89	0.05	7.47	57.46
Parametrizable	4	320	3	320	176750	3.01	5.75	0.05	8.81	57.41
Parametrizable	4	320	4	192	176750	2.62	5.10	0.05	7.77	63.61
Parametrizable	4	320	4	256	176750	3.33	6.34	0.05	9.72	68.56
Parametrizable	4	320	4	320	176750	4.06	7.57	0.06	11.70	69.41
Parametrizable	5	320	3	192	219744	2.00	3.99	0.06	6.05	51.73
Parametrizable	5	320	3	256	219744	2.50	4.87	0.06	7.44	51.58
Parametrizable	5	320	3	320	219744	2.93	5.62	0.06	8.62	51.45
Parametrizable	5	320	4	192	219744	2.63	5.11	0.06	7.80	60.53
Parametrizable	5	320	4	256	219744	3.34	6.36	0.07	9.77	61.66
Parametrizable	5	320	4	320	219744	4.04	7.60	0.07	11.70	61.76
Parametrizable	5	320	5	192	219744	3.23	6.19	0.07	9.49	59.70
Parametrizable	5	320	5	256	219744	4.15	7.80	0.07	12.00	66.63
Parametrizable	5	320	5	320	219744	5.04	9.55	0.07	14.70	73.40
Parametrizable	6	448	3	192	364630	2.01	3.98	0.10	6.10	42.57
Parametrizable	6	448	3	256	364630	2.46	4.75	0.10	7.31	42.55
Parametrizable	6	448	3	320	364630	2.85	5.43	0.10	8.38	42.54
Parametrizable	6	448	3	384	364630	3.25	6.10	0.10	9.45	42.46
Parametrizable	6	448	3	448	364630	3.64	6.76	0.10	10.50	42.50
Parametrizable	6	448	4	192	364630	2.67	5.15	0.10	7.92	51.01
Parametrizable	6	448	4	256	364630	3.38	6.39	0.10	9.87	51.79
Parametrizable	6	448	4	320	364630	4.03	7.52	0.11	11.70	51.66
Parametrizable	6	448	4	384	364630	4.58	8.45	0.11	13.10	51.71
Parametrizable	6	448	4	448	364630	5.12	9.32	0.11	14.60	51.63
Parametrizable	6	448	5	192	364630	3.28	6.23	0.10	9.61	59.63
Parametrizable	6	448	5	256	364630	4.19	7.82	0.11	12.10	60.62
Parametrizable	6	448	5	320	364630	5.08	9.41	0.11	14.60	60.70
Parametrizable	6	448	5	384	364630	5.93	10.90	0.11	16.90	60.93
Parametrizable	6	448	5	448	364630	6.71	12.20	0.11	19.00	60.96
Parametrizable	6	448	6	192	364630	3.91	7.35	0.11	11.40	56.05

Version	Max N	Max K	N	K	Total area [µm^2]	Switching power [mW]	Internal power [mW]	Leakage power [mW]	Total power [mW]	SNR u_hat [dB]
Parametrizable	6	448	6	256	364630	5.02	9.29	0.11	14.40	65.73
Parametrizable	6	448	6	320	364630	6.12	11.20	0.11	17.50	68.08
Parametrizable	6	448	6	384	364630	7.19	13.10	0.11	20.40	68.81
Parametrizable	6	448	6	448	364630	8.12	15.40	0.11	23.60	68.88
Parametrizable	7	448	3	192	424496	2.01	3.98	0.12	6.11	38.41
Parametrizable	7	448	3	256	424496	2.42	4.67	0.12	7.21	38.44
Parametrizable	7	448	3	320	424496	2.82	5.34	0.12	8.28	38.37
Parametrizable	7	448	3	384	424496	3.21	6.00	0.12	9.33	38.50
Parametrizable	7	448	3	448	424496	3.61	6.65	0.12	10.40	38.44
Parametrizable	7	448	4	192	424496	2.65	5.13	0.12	7.90	45.59
Parametrizable	7	448	4	256	424496	3.35	6.36	0.12	9.83	45.62
Parametrizable	7	448	4	320	424496	3.95	7.39	0.12	11.50	45.62
Parametrizable	7	448	4	384	424496	4.49	8.27	0.12	12.90	45.32
Parametrizable	7	448	4	448	424496	5.03	9.16	0.12	14.30	45.23
Parametrizable	7	448	5	192	424496	3.29	6.26	0.12	9.67	55.53
Parametrizable	7	448	5	256	424496	4.21	7.87	0.12	12.20	55.75
Parametrizable	7	448	5	320	424496	5.09	9.42	0.12	14.60	55.73
Parametrizable	7	448	5	384	424496	5.91	10.80	0.13	16.90	55.58
Parametrizable	7	448	5	448	424496	6.65	12.10	0.13	18.80	55.39
Parametrizable	7	448	6	192	424496	3.91	7.37	0.12	11.40	57.87
Parametrizable	7	448	6	256	424496	5.01	9.30	0.12	14.40	62.32
Parametrizable	7	448	6	320	424496	6.11	11.20	0.13	17.50	63.58
Parametrizable	7	448	6	384	424496	7.18	13.10	0.13	20.40	63.33
Parametrizable	7	448	6	448	424496	8.21	14.90	0.13	23.20	63.29
Parametrizable	7	448	7	192	424496	4.51	8.44	0.12	13.10	54.88
Parametrizable	7	448	7	256	424496	5.82	10.70	0.13	16.70	61.62
Parametrizable	7	448	7	320	424496	7.11	13.00	0.13	20.30	67.83
Parametrizable	7	448	7	384	424496	8.38	15.30	0.13	23.80	70.72
Parametrizable	7	448	7	448	424496	9.59	18.30	0.13	28.00	71.37
Parametrizable	8	512	3	192	551621	2.05	3.95	0.15	6.16	34.33
Parametrizable	8	512	3	256	551621	2.45	4.63	0.15	7.24	34.37
Parametrizable	8	512	3	320	551621	2.84	5.31	0.15	8.31	34.36
Parametrizable	8	512	3	384	551621	3.24	5.99	0.15	9.38	34.33

Version	Max N	Max K	N	K	Total area [µm^2]	Switching power [mW]	Internal power [mW]	Leakage power [mW]	Total power [mW]	SNR u_hat [dB]
Parametrizable	8	512	3	448	551621	3.64	6.67	0.15	10.50	34.33
Parametrizable	8	512	3	512	551621	4.03	7.34	0.15	11.50	34.43
Parametrizable	8	512	4	192	551621	2.72	5.15	0.15	8.01	41.07
Parametrizable	8	512	4	256	551621	3.39	6.33	0.15	9.87	41.07
Parametrizable	8	512	4	320	551621	3.96	7.29	0.16	11.40	41.07
Parametrizable	8	512	4	384	551621	4.50	8.19	0.15	12.80	40.90
Parametrizable	8	512	4	448	551621	5.04	9.10	0.15	14.30	40.88
Parametrizable	8	512	4	512	551621	5.58	9.99	0.16	15.70	40.82
Parametrizable	8	512	5	192	551621	3.36	6.30	0.15	9.82	49.52
Parametrizable	8	512	5	256	551621	4.26	7.88	0.16	12.30	49.62
Parametrizable	8	512	5	320	551621	5.12	9.39	0.16	14.70	49.62
Parametrizable	8	512	5	384	551621	5.89	10.70	0.16	16.70	49.69
Parametrizable	8	512	5	448	551621	6.58	11.80	0.16	18.60	49.73
Parametrizable	8	512	5	512	551621	7.28	13.00	0.16	20.40	50.15
Parametrizable	8	512	6	192	551621	4.00	7.42	0.16	11.60	55.57
Parametrizable	8	512	6	256	551621	5.10	9.35	0.16	14.60	56.59
Parametrizable	8	512	6	320	551621	6.19	11.30	0.16	17.60	56.69
Parametrizable	8	512	6	384	551621	7.24	13.10	0.16	20.50	56.65
Parametrizable	8	512	6	448	551621	8.23	14.80	0.16	23.20	56.47
Parametrizable	8	512	6	512	551621	9.14	16.30	0.16	25.60	56.44
Parametrizable	8	512	7	192	551621	4.60	8.50	0.16	13.30	54.43
Parametrizable	8	512	7	256	551621	5.90	10.80	0.16	16.80	61.16
Parametrizable	8	512	7	320	551621	7.20	13.10	0.16	20.40	62.69
Parametrizable	8	512	7	384	551621	8.46	15.30	0.16	23.90	62.89
Parametrizable	8	512	7	448	551621	9.70	17.40	0.17	27.30	62.95
Parametrizable	8	512	7	512	551621	10.90	19.50	0.17	30.50	62.91
Parametrizable	8	512	8	192	551621	5.22	9.61	0.16	15.00	51.17
Parametrizable	8	512	8	256	551621	6.71	12.20	0.16	19.10	59.51
Parametrizable	8	512	8	320	551621	8.20	14.80	0.16	23.20	64.40
Parametrizable	8	512	8	384	551621	9.67	17.40	0.17	27.20	68.53
Parametrizable	8	512	8	448	551621	11.10	20.00	0.17	31.30	69.52
Parametrizable	8	512	8	512	551621	12.50	22.40	0.17	36.10	69.36



NTNU

Norwegian University of
Science and Technology