A MASTER THESIS ON

Stochastic Computing and its Application to Sound Source Localization

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

Diplom-Ingenieur

(Equivalent to Master of Science)

in

Institute of Computer Technology (E384)

by

Peter Schober

01355178

Supervisor(s):

Prof. Axel Jantsch

Dr. Nima Taherinejad

M. Hassan Najafi

Vienna, Austria

January 2022

Abstract

Stochastic computing (SC) is an alternative computing paradigm that processes data in the form of long uniform bit-streams rather than conventional compact weighted binary numbers. SC is fault-tolerant and can compute on small, efficient circuits, promising advantages over conventional arithmetic for newer and smaller computer chips. However, SC is primarily used in scientific research, not in practical applications. Digital sound source localization (SSL) locates speakers using multiple microphones in cell phones, laptops, and other voice-controlled devices. SC has not been integrated to SSL in practice nor in theory until now. In this work, for the first time to the best of our knowledge, we replace a conventional computational block of an existing SSL algorithm with an SC-based design and implement a working SC-based sound source localizer. The practical part of this work shows that the proposed stochastic circuits do not rely on conventional analog-to-digital conversion and can process data in the form of pulse-width-modulated (PWM) signals. The proposed SC design consumes up to 39 % less area than the conventional reference design. The SC-based design can consume less power depending on the computational accuracy, for example, 6 % less power consumption for 3-bit inputs. The presented stochastic circuit is not limited to SSL and is readily applicable to other practical applications.

Kurzfassung

Stochastik Computing (SC) ist eine Alternative zu konventioneller binärer Computerarithmetik, die mit langen Bit-Folgen anstelle von kurzen binären Zahlen rechnet. Zurzeit wird SC hauptsächlich in der Forschung verwendet, obwohl es durch hohe Fehlertoleranz und kleine effiziente Schaltungen Vorteile gegenüber konventioneller Computerarithmetik verspricht. Digitale Sound Source Lokalisation (SSL) ortet Audioquellen mit zwei oder mehreren Mikrophonen und kommt in Mobiltelefonen, Laptops und anderen sprachgesteuerten Geräten zum Einsatz. Bisher wurde weder in der Theorie noch in der Praxis versucht SSL mit SC zu kombinieren. In dieser Arbeit ersetzten wir, für einen bestehenden SSL Algorithmus, einen konventionellen Rechenblock durch einen auf SC basierenden. Der praktische Teil der Arbeit zeigt, dass stochastische Schaltungen keine klassischen analog-zu-digital Konverter benötigen, sondern auch mit analog erzeugten Pulsen rechnen können. Unsere Analyse ergibt, dass digitale stochastische Schaltungen bis zu 39 % kleiner sind und je nach Rechengenauigkeit mehr oder weniger Energie benötigen. Beispielsweise einen 6 % geringeren Energieverbrauch bei 3-bit Rechengenauigkeit. Die vorgestellte stochastische Schaltung ist nicht auf SSL begrenzt, sondern kann ohne Mehraufwand in anderen Anwendungen und Bereichen eingesetzt werden, da sie ohne zusätzliche Änderungen im restlichen Rechensystem auskommt.

Acknowledgment

This thesis was written at the Institute of Computer Technology at the Technical University of Vienna. A special thank goes out to:

Univ. Prof. Dipl.-Ing. Dr. techn. Axel Jantsch for his time and feedback.

Dr. Nima Taherinejad for the invaluable knowledge and work he contributed to the project. The door to his office was always open, and he was willing to help me at any time.

M. Hassan Najafi for his warm welcome in America at the University of Louisiana in Lafayette, the starting point for my thesis. I also want to thank him for sharing his expertise and time during this project.

All family members and friends for their continuous support throughout my studies.

Preface

A version of Section 5.2 was published as
Peter Schober, M. Hassan Najafi, and Nima TaheriNejad, "*High-Accuracy Multiply-Accumulate (MAC) Technique for Unary Stochastic Computing*", IEEE Transactions on Computers, pp.1-14, 2021.

Contents

List of Tables

List of Figures

Acronyms

AC	alternating current. 35
ADC	analog-to-digital converter. viii, 5, 28, 29, 32, 33, 35, 36, 37, 38, 46, 47, 58, 60, 61, 63
APC	accumulative parallel counter. 44, 62
ASIC	application-specific integrated circuit. vii, 27, 56, 57, 58, 59, 61, 63
CC	cross-correlation. vii, ix, 23, 24, 25, 28, 29, 30, 31, 32, 41, 42, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 70, 73, 74
CMOS	Complementary metal-oxide-semiconductor. 28, 56, 61
CS	chip select. 36
DC	direct current. 34, 35
DOA	direction-of-arrival. viii, 18, 19, 20, 23, 24, 25, 34
FFT	fast Fourier transform. ix, 31, 53, 54
FPGA	field-programmable gate array. vii, ix, 27, 29, 30, 34, 35, 36, 37, 38, 39, 41, 49, 56, 57, 58, 59, 61, 63
HDL	hardware description language. 27, 56, 58, 63
IIR	infinite impulse response. 49, 50, 54
LCM	least common multiple. 15, 45
LED	light emitting diode. ix, 29, 30, 39, 40, 48, 49, 53
LFSR	linear feedback shift register. 6, 7
LSB	least significant bit. 29, 45
LUT	lookup table. ix, 57, 58, 59, 63
MAC	multiply-accumulate. ix, 31, 32, 33, 41, 42, 43, 44, 45, 46, 48, 49, 51, 52, 58, 60, 61, 62, 63
MAE	mean absolute error. vii, 55, 56, 61
MISO	master in slave out. 36

MOSI master out slave in. 36

MSB most significant bit. 45

MUX multiplexer. 12, 13, 14, 43

OPAMP operational amplifier. 34, 35

PE probability estimator. 6, 7, 43, 46, 62

PMOD Peripheremodule. 34, 35

PWM pulse-width-modulated. ii, viii, ix, 15, 29, 32, 33, 34, 35, 36, 37, 38, 39, 44, 46, 47, 48, 51, 58, 60, 61, 63

RNG random number generator. viii, 6, 15, 16

SC stochastic computing. ii, iii, vii, viii, ix, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 26, 27, 28, 29, 30, 31, 32, 33, 34, 39, 43, 44, 46, 50, 51, 56, 57, 58, 60, 61, 62, 63, 64

SCC stochastic cross-correlation. 8, 9

SCLK serial clock. 36, 59

SN stochastic number. vii, 4, 5, 6, 7, 8, 10, 11, 12, 14, 16, 46, 51, 58, 62

SNG stochastic number generator. viii, 5, 6, 7, 8, 9, 14, 15, 16, 32, 43, 44, 45, 51, 52, 62

SNR signal-to-noise ratio. 2, 52

SPI serial peripheral interface. viii, 29, 34, 36, 37, 38, 47, 57, 58, 59, 60, 61

SPL sound-pressure level. 19, 52

SSL sound source localization. ii, iii, 1, 2, 3, 4, 17, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 39, 41, 44, 50, 51, 52, 53, 54, 61, 62, 63, 64

TDE time delay estimation. ix, 20, 23, 24, 29, 30, 31, 32, 42, 48, 53, 54, 56, 63, 70

VAD voice activity detection. 21, 22, 52

VHDL very-high-speed integrated circuit hardware description language. 27, 61

Chapter 1

Introduction

In the 1960s, researchers in the United States and the United Kingdom developed the fundamental concepts of stochastic computing (SC) [2–5]. Inspired by the human brain, researchers computed basic mathematical operations with sequences of random pulses. In the 1980s, however, weighted binary computations seemed to be a better computation paradigm, and research interest in SC decreased [6,7]. Conventional computing using weighted binary numbers has dominated the digital world since then. Improvements in chip manufacturing have propelled rapid technological advances, and the processing performance of computer chips increased exponentially. Moore's Law has successfully predicted a downscaling of transistor sizes and a constant increase of transistor density. Smaller technology nodes achieve higher performance and clock speeds, while power consumption remains constant [8].

The shrinking size of transistors has imposed a challenge in maintaining reliability because smaller circuits are more vulnerable to soft errors[1] [9]. Eventually, chip manufactures will not be able to further reduce the transistor size and power consumption due to physical constraints. The limitation motivates more research in parallel computing (e.g., multi-core), specialization (custom chips) and alternative computing paradigms such as approximate computing and SC. The alternative computing paradigms focus on reducing power consumption and error tolerance rather than achieving accurate computing [10]. An increasing interest in artificial intelligence and neural networks contributed to more research on SC in the last decade. Nevertheless, SC has not superseded the conventional weighted binary computing in a single practical application. Computing a single mathematical operation with stochastic bit-streams is simple, but multi-stage processing and efficient generation of bit-streams are complex and have several limitations [1]. In addition, most research focus on applying SC to applications such as neural networks for video and image processing [11–14]. Other applications are rarely studied. To the best of our knowledge, this thesis is the first work that applies SC to sound source

[1]Soft errors cause data to change and can occur due to external environmental factors, such as high-energy atmospheric neurons.

localization (SSL).

In recent years, recording and processing audio is gaining more and more attention as modern communication technology has become a part of our daily lives. Voice control as a human-machine interface is widespread. For example, it appears in smart homes and personal assistants (e.g., Amazon Alexa and Google Assistant). Current laptops and smartphones are equipped with multiple microphones to record user voices and ambient sound. A set of microphones allows spatial filtering of the recorded sound, which increases the signal-to-noise ratio (SNR) by suppressing most of the noise outside the direction of interest. The process of digitally listening at the speaker's location is known as beamforming which requires the exact position of the speaker [15]. SSL is part of the audio signal processing whenever the user does not speak directly into a microphone.

In this chapter, we first discuss the motivation of this thesis, following by an explanation of its goals and scope. Finally, we provide an overview of the thesis structure.

1.1 Motivation

Researchers in SC community search for practical applications in which SC could come out ahead and replace conventional binary computing. To date, there has not been sufficient research that examines the application of SC to SSL. This thesis analyzes the accuracy and potential hardware area and power savings. Efficient sound signal processing is an active field of research as many consumer devices are battery-powered. SC promises high power efficiency that can contribute to longer battery lives. SSL is a real-time application, starting with capturing audio data and ending in location estimations. Therefore, analyzing the power consumption of the signal pre-processing and analog-to-digital conversion for SC could lead to further power savings. Developing a sound source localizer using SC can further increase the research and public interest in SC.

1.2 Goal and Scope

This thesis targets the computationally intensive parts to replace their costly conventional binary computations with low-cost SC alternatives. The emphasis is on applying SC to a new application, namely SSL, designing a novel SC-based functional unit, and analyzing the limitations and advantages of SC for SSL. The work focuses on time-discrete SC, although a time-continuous variant of the sound source localizer is also feasible. However, complete design and study of a time-continuous SC-based sound source localizer requires further research and is beyond the scope of this thesis. This study focuses on the recently introduced deterministic SC [16–18] because it allows computations with lower latency compared to the conventional random SC. The thesis also aims to show the challenges, constraints and promises of using SC in this new application domain but does not contribute to the theory behind SSL.

1.3 Overview

This thesis consists of six chapters within three main parts. In PART I (Chapters 2 and 3), we provide the required background on SC and SSL; Chapter 2 gives an overview of SC. Chapter 3 deals with the theory of SSL and provides an overview of sound source localizers. Chapter 5 elaborates some modifications to the signal processing chain to realize the proposed SC-based design. In PART II (Chapters 4 to 6), we start by formal statement of the research question and hypotheses, followed by an overview of the used methods and research methodology. Chapter 5 discusses the details of the practical implementation. Chapter 6 presents the accuracy and resource consumption results. PART III (Chapters 7 and 8) contains the study's interpretation, discussion and conclusion.

Chapter 2

Stochastic Computing

This chapter introduces stochastic circuits for those who may be new to the subject. It also demonstrates the limitations and implications of using randomness in stochastic computations. The first part of this chapter introduces conventional random SC, stochastic formats (unipolar, bipolar) and SC arithmetic. We focus on single-line (i.e., serial) SC and will not cover multi-line (parallel) SC as it adds more complexity to the proposed circuits. However, multi-line stochastic circuits are feasible and could be used for SSL [12, 19]. The last section of this chapter introduces deterministic SC, which addresses many limitations of random SC.

2.1 Fundamental Concepts

In SC, numbers are coded into stochastic sequences 0s and 1s. A numerical value encoded into a stochastic sequence is called a stochastic number (SN). The logic circuit for doing computations on SNs is called a stochastic circuit [6, 20]. SC is possible in both the time-continuous and the time-discrete domain, but some operations such as delaying and storing stochastic sequences are simpler in the time-discrete domain. The concept of coding a value into a SN is fundamentally different from the concept of conventional computing[1]. Stochastic sequences have voltage level 1 (high) with probability ρ and level 0 (low) with probability $1 - \rho$. In the time-discrete domain, researchers use the term (stochastic) *bit-stream* or *bit-sequence* rather than stochastic sequences. A bit within a random bit-stream is 1 with probability $\mathrm{P}(S_i = 1) = \rho$ and is 0 with probability $1 - \rho$. We use the notation S_i for the ith bit in bit-stream S with length N ($0 \leq i \leq N-1$). A stochastic sequence S with $20\,\%$ 1s and $80\,\%$ 0s is an SN with probability $\rho = 0.2$. The relative position of 1s within a stochastic sequence does not affect the represented value. Since different stochastic sequences can represent the same value, SNs are defined by probability rather than a particular order of bits. Bit-streams are considered as Bernoulli processes.

[1]In weighted binary representation, the value of a number is coded into the position of 1s within the binary number.

Each bit is independent of other bits and is 1 with the same probability of ρ [21]. The expected value (E) of 1s in bit-stream (N_1) is the product of the total length and probability:

$$\mathrm{E}(N_1) = N\rho \tag{2.1}$$

$$\rho = \frac{\mathrm{E}(N_1)}{N} = \lim_{N\to\infty} \frac{N_1}{N}. \tag{2.2}$$

Since in practice the bit-stream length is limited, we need to convert the data from a continuous to a discrete probability. This conversion involves a quantization error similar to the quantization error of analog-to-digital converters (ADCs). The resolution of the bit-stream increases by increasing the bit-stream length. If the probability cannot be represented with a finite bit-stream length, we can write $\hat{\rho}$ to indicate the approximation:

$$\rho \approx \hat{\rho} = \frac{N_1}{N} \tag{2.3}$$

The quantization behavior in unipolar SC is similar to the unsigned weighted binary Q-format (Qi.j) with $i = 0$ integer bits and j fraction bits. However, the bit-stream length is exponentially greater than the number of fraction bits ($N = 2^j$), also denoted as equivalent precision. The relation $N = 2^j$ is a disadvantage of SC compared to weighted binary computing. We need to double the length of bit-streams to gain one extra bit of equivalent precision. For example, a two-bit bit-stream can represent the probabilities $\rho = [0, 0.5]$ and a two-bit weighted binary can represent $[0, 0.25, 0.5, 0.75]$. A two-bit bit-stream can represent two different probabilities, but a two-bit fixed-point weighted binary number can represent four different values. This is generalized to n different numbers with an n-bit bit-stream and 2^n numbers with an n-bit binary number.

In general, a computing system consists of several processing blocks linked to each other. A processing block receives data from the previous block and passes the results to the next one. A stochastic circuit can replace one or more weighted-binary processing blocks of the computing system. Typically, we want to implement a processing block that is computationally intensive in stochastic domain to maximize the savings in resource consumption. Implementing a processing block that requires floating-point or high data precision is impractical in stochastic domain as extremely long bit-streams are needed. Stochastic circuits are best suited for low to medium precision computations. Within conventional computing systems, inputs to a processing block are usually available in weighted binary representation and the results should be passed as a weighted binary number to the next processing block. For that reason, an SC processing system in general consists of three sub-blocks. First, a stochastic number generator (SNG) converts inputs from weighted binary representation to SNs. Second, an

SC circuit that does the actual computation. Third, a probability estimator (PE) transforms the results into weighted binary numbers. The third block is also known as a back-converter or stochastic-to-binary converter. Note that the first and the last block usually have to convert multiple numbers in parallel when the processing block has more than one input and output, respectively. SNGs and PEs are arranged in an array in multi-input stochastic circuits. A block diagram of an SC processing system is illustrated in Figure 2.1.

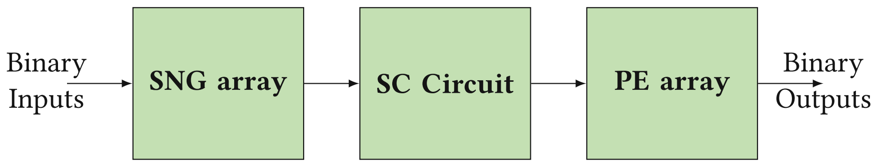

Figure 2.1: Fundamental elements of an SC processing system [1].

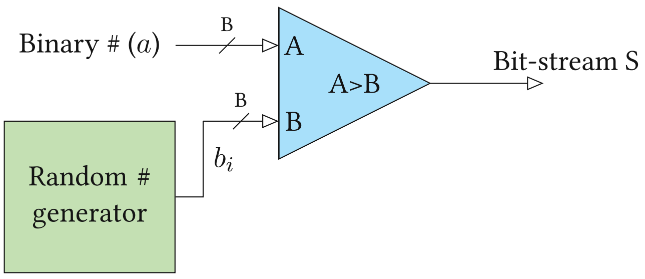

Figure 2.2: An SNG consists of a comparator and an RNG. The inputs are weighted binary numbers with B-bits precision, and the output is a stochastic bit-stream.

An SNG is the first sub-block of an SC processing system that generates stochastic sequences with the desired properties (probability, statistics). Figure 2.2 shows the block diagram of a simple SNG. The illustrated SNG has a data input (binary number), an output (SN), an RNG, and a binary comparator. The stochastic sequence generation works as follows. The comparator output is $S_i = 1$ whenever the input (a) is greater than the random number (b_i) and is $S_i = 0$ otherwise.

$$S_i = \begin{cases} 1, & a > b_i \\ 0, & a \leq b_i \end{cases} \tag{2.4}$$

The statistics of random numbers determine the statistical properties of the stochastic sequence [22]. The distribution of 1s within the stochastic sequence is uniform and random when the random numbers are uniformly distributed and statistically independent. RNGs have been extensively studied in digital and analog implementations for the time-continuous and time-discrete domain [23–27]. Linear feedback shift registers (LFSRs) are commonly used as RNGs because of their simple and efficient design. Still, prior research showed that the RNGs often take up to 90 % of the area of SC designs [28]. A B-bit

LFSR generates a periodic sequence with a maximum period length of 2^B, where B is the bitwidth of the shift register. The output sequence is pseudo-random, rather than truly random, because it is periodic and can be reproduced by having the feedback function and the seed value. However, if B is large, the pseudo-random number can be considered as a random number. For more details on random SNGs, the readers are referred to [7].

The stochastic circuit block performs the arithmetic operations and consists of combinational or sequential circuits (e.g., finite state machines [14]). The stochastic circuit receives SN from the SNG and passes the results to the PE.

A PE converts SNs to weighted binary numbers. In the time-continuous domain, a voltage integrator works as a PE [20]. In the time-discrete domain, the PE often consists of two sub-blocks. A binary counter converts SNs to weighted binary representation in the $[0, 2^B - 1]$ interval, with B being the bitwidth of the counter. The counter value increments whenever the current bit of the bit-stream is 1. The second sub-block scales the counter value to the needed range. The counter value (N_1) needs to be divided by the length of the bit-stream if the result should be a probability estimation ($\hat{\rho}$) in the $[0, 1]$ interval.

$$\hat{\rho} = \frac{1}{N} N_1 = \frac{1}{N} \sum_{i=0}^{N-1} S_i \tag{2.5}$$

Ideally, N is a power of two. In that case, the division equals shift operations and can be implemented with less computational effort. In summary, SNGs generate SNs, and PEs convert the results (i.e., SNs) back to conventional binary. We next discuss correlation and its significance for SC.

2.1.1 Correlation

In the previous section, we discussed how data values are converted from binary to stochastic representation by encoding values into the probability of observing 1s in a stochastic sequence. Correlation and dependency between stochastic bit-streams affect the accuracy and arithmetic of stochastic circuits. Before introducing the arithmetic, we introduce the concepts of positively correlated, negatively correlated and uncorrelated bit-streams. In the standard definition for correlation (i.e., Pearson correlation), two random variables X and Y are uncorrelated when their correlation coefficient $\gamma(X, Y)$ is zero. Two events are independent when their joint probability equals the product of their probabilities.

$$\gamma(X,Y) = \frac{[(X-\mathrm{E}[X])(Y-\mathrm{E}[Y])]}{\sigma_X\sigma_Y} = 0. \tag{2.6}$$

$$\mathrm{P}(X_i=1, Y_i=1) = \mathrm{P}(X_i=1)\,\mathrm{P}(Y_i=1) = \rho_x\rho_y \tag{2.7}$$

Where E is the expected value, and σ_X, σ_Y are the standard deviations of X and Y, respectively. $\mathrm{P}(X_i = 1) = \rho_x$ is the probability of observing 1s. When two bit-streams are independent, they are also uncorrelated. Generally, this is not true the other way around with the definitions in Equations (2.6) and (2.7). The traditional definition for correlation is unsuitable for SC because it imposes constraints on the expected value of the stochastic sequences [29, 30]. In SC, we want an indicator (a metric) not affected by the number of ones in the bit-stream. For that reason, a new correlation measure, namely stochastic cross-correlation (SCC) [31], has been proposed for SC. The SCC lies in the $[-1, 1]$ interval and is defined as follows:

$$SCC(X,Y) = \begin{cases} \frac{\rho_{\mathrm{AND}(X,Y)}-\rho_X\rho_Y}{\min(\rho_X,\rho_Y)-\rho_X\rho_Y} & \text{if } \rho_{\mathrm{AND}(X,Y)} > \rho_X\rho_Y \\ \frac{\rho_{\mathrm{AND}(X,Y)}-\rho_X\rho_Y}{\min(\rho_X,\rho_Y)-\max(\rho_X+\rho_Y-1,0)} & \text{otherwise} \end{cases} \tag{2.8}$$

This definition of correlation has the convenient property that the measure does not depend on the encoded values. Further, if $SCC = 0$, the two random numbers (i.e., SNs) are uncorrelated and independent. The SCC gives the value +1 for maximum overlap of 1s and 0s between the two bit-streams. In this work, $SCC(X,Y) = +1$ stands for positively correlated bit-streams. $SCC(X,Y) = -1$ shows minimum overlap of 1s between the bit-streams and is called negatively correlated. Table 2.1 gives three examples for uncorrelated, negatively correlated and positively correlated bit-streams.

Table 2.1: Examples for correlation between bit-streams.

Stochastic Bit-stream S_1	Stochastic Bit-stream S_2	SCC(X_{S_1}, Y_{S_2})
11110000	11001100	0 -> uncorrelated
11110000	00001111	-1 -> negatively correlated
11111100	11110000	1 -> positively correlated

In SC, some operations (such as minimum using AND and maximum using OR gate) require positively correlated input bit-streams, while some operations (such as multiplication) require uncorrelated bit-streams. There are also operations that need negatively correlated inputs (see Section 2.3.2). The block diagram in Figure 2.3 shows an SNG that generates negatively correlated bit-streams S_1 and S_2 when the switch is connected to SCC$= -1$. The bit-streams S_1 and S_2 are positively correlated when the inverter is bypassed, and the switch is connected to SCC$= +1$. Correlation between bit-streams can also be manipulated after the generation step by using additional circuitry. We refer the readers to [16,30] for more information on correlation manipulating circuits.

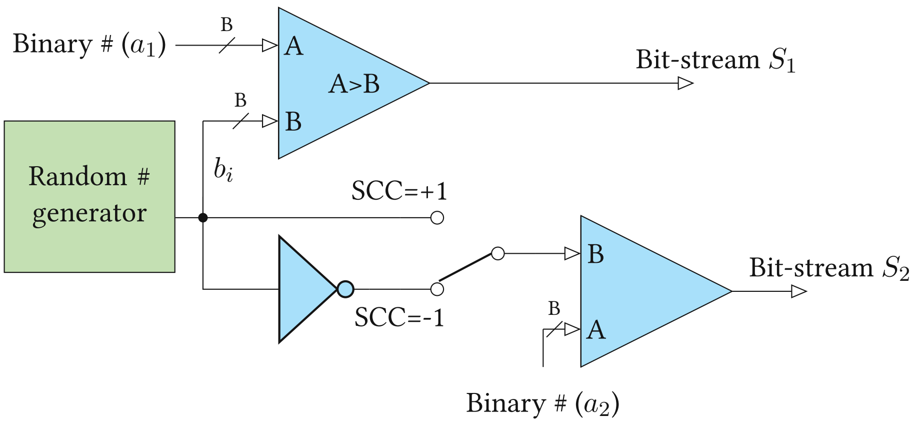

Figure 2.3: The SNG generates negatively or positively correlated random bit-streams depending on the position of the switch.

2.1.2 Errors in Stochastic Computing

SC is a computation paradigm that trades higher energy efficiency for lower accuracy and precision. The primary sources of errors are 1) correlation error, 2) rounding errors, 3) random fluctuations error, and 4) approximation errors. Other error sources are physical errors from unreliable hardware or bit-flips from environmental effects [6].

1) Correlation Error. Bit-streams are ideally correlated or uncorrelated after generation. In stochastic circuits with successive operations, the SCC between bit-streams is often unknown or a mix between perfectly correlated and ideally uncorrelated. The arithmetic of stochastic circuits (introduced in Section 2.3) is inaccurate if the level of correlation between bit-streams is not as needed.

2) Rounding Error. Errors from quantization reflect that an N-bit bit-stream can only represent N different numbers. For unsigned SC, the representable values are $0, \frac{1}{N}, \frac{2}{N}, \ldots \frac{N-1}{N}, 1 \in [0,1]$. Any other probability needs to be rounded to the nearest representable value. For example, an SNG that generates a bit-stream with $N = 16$ bits cannot encode the probability $\rho = 0.1555$. The bit-stream can have two 1s and represent $x = \frac{2}{16} = 0.125$ or three 1s and represent $x = \frac{3}{16} = 0.1785$. In both cases, the SNG produces an error that can only be reduced by increasing N. When the bit-stream length increases, the computation latency increases. Longer bit-streams increase the processing time and the power consumption of arithmetic operations. In general, SC is limited to smaller precisions, and the quantization error is one of the main sources of error.

3) Random Fluctuations Error. Inherent randomness in generating random bit-streams causes random fluctuations error particularly when generating short bit-streams. In such cases, the number of ones in the bit-stream fluctuates depending on the RNGs used resulting in representing different values. In Section 2.4, we discuss that the deterministic methods of SC [32] can eliminate the error caused by random fluctuations.

4) Approximation Error. In SC, some arithmetic functions are approximated. A common example is approximation of the addition operation by union ($z_{approx} = x_1 + x_2 - x_1x_2$). The error ($|x_1x_2|$) might be acceptable because computing the union instead of addition is more efficient in SC.

2.2 Stochastic Numerical Formats

In SC, values are encoded into the probability of observing 1s in stochastic sequences. So far, we assumed that a stochastic sequence with probability (ρ) represents a number (a) with a simple strategy of $a = \rho$. This mapping is called unipolar coding. Other formats are required when SN should represent negative numbers or numbers greater than 1. In what follows, we introduce the most common formats of single-line SC that were introduced in the literature [6, 33, 34].

Unipolar Format: In the unipolar format, the probability of observing 1s in the stochastic sequence is equal to the represented value. The SN is the ratio of the bit-stream length to the number of 1s in the bit-stream (N_1). The ratio $\frac{N_1}{N}$ is the approximation of the probability from Equation (2.3).

$$a = \mathrm{P}(S_i = 1) = \rho \tag{2.9}$$

$$x = \frac{N_1}{N} = \frac{1}{N}\sum_{i=0}^{N-1} S_i \tag{2.10}$$

The value of the SN in Equation (2.10) is equivalent to the function implemented by the probability estimator in Equation (2.5). The SNs (and probabilities) are limited to the $[0, 1]$ interval. Compared to the weighted binary representation, the range of the unipolar format is equal to the unsigned fixed-point Q-format Qi.j with $i = 0$. For example, the bit-stream $S = 0101000$ has the probability $\rho = \frac{2}{6}$. The SN in the unipolar format is $x = \rho = \frac{2}{6} = \frac{1}{3}$. When we take the difference of two successive, representable values, the resolution is $\frac{1}{N}$. To reach an equivalent resolution of 2^{-B} (as in a comparable fixed-point representation), a bit-stream of $N = 2^{B+1}$ bit length is needed.

Bipolar Format: The bipolar format enables signed computations. It maps the probabilities to the $[-1, 1]$ interval. The represented value is the difference in probabilities of observing 1s and observing 0s.

$$a = \mathrm{P}(S_i = 1) - \mathrm{P}(S_i = 0) \tag{2.11}$$

$$= \rho - (1-\rho) = 2\rho - 1 = \frac{2\,\mathrm{E}(N_1) - N}{N} \tag{2.12}$$

$$x = \frac{2N_1 - N}{N} = \frac{2}{N}\sum_{i=0}^{N_1} S_i - 1 \tag{2.13}$$

The binary input values and the result can be in the $[-1, 1]$ interval. The representable interval equals the signed Q-format[2] (Qi.j) with $i = 0$. To give a simple example of an SN, the bit-stream $S =$ 010100 has the probability $\rho = \frac{2}{6}$. The corresponding SN in the bipolar format is $x = 2 \times \frac{2}{6} - 1 = -\frac{1}{3}$. We can also calculate the value by replacing all 0s with -1s and computing the mean. The bit-stream S = 0101100 becomes S' = -1 1 -1 1 -1 -1 with $x = \frac{1}{N}\sum_{i=0}^{N-1} S'_i = \frac{1}{6}(2 \times 1 - 4 \times -1) = -\frac{1}{3}$. The bipolar format doubles the representable range compared to the unipolar format. The difference between two successive, representable values is $\frac{2}{N}$. The required bit-stream length to resolve 2^{-B} is doubled compared to the unipolar format ($N = 2^{B+1}$).

2.3 Stochastic Arithmetic

In the previous sections, we discussed the conversion from and to stochastic sequences, introduced different mappings between probability and SN, and the concept of correlation. The numerical format and correlation of bit-streams affect the arithmetic of stochastic circuits. This thesis focuses on the arithmetic for the linear formats and standard digital components. We first assume that the distribution of 1s in the bit-streams is independent and uncorrelated ($SCC = 0$). In the second part, we show how correlations between bit-streams change arithmetic. The illustrated examples are all in unipolar format for uncorrelated inputs.

2.3.1 Uncorrelated Stochastic Computing

SC can perform arithmetic operations with simple standard logic gates. We show a series of examples that illustrate the most basic SC operations and highlight the strengths of SC. The functions are explained with simple probability theory. For shortened notation, we write ρ_S instead of $\mathrm{P}(S_1 = 1)$ for the probability of observing 1 in bit-stream S_1. We also write $x = \rho = \frac{N_1}{N}$ for SNs in the unipolar format and $x = 2\rho - 1 = \frac{2N_1 - N}{N}$ for SNs in the bipolar format. We neglect that the resolution is generally not sufficient to exactly represent the probability. For a derivation of the arithmetic, we refer

[2]The sign is separately counted and not included in i.

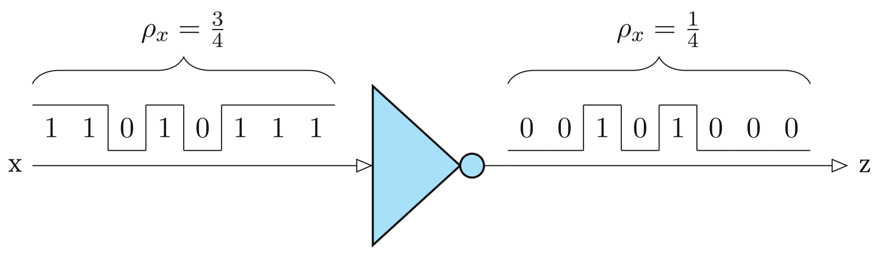

Figure 2.4: The NOT-gate is the stochastic $\rho_z = 1 - \rho_x$ operation

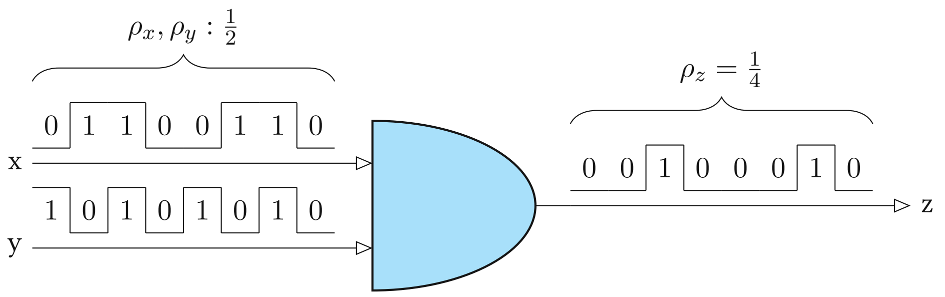

Figure 2.5: The AND-gate is the stochastic $\rho_z = \rho_x \rho_y$ operation

the readers to the literature [6, 33].

NOT Operation The NOT operation is the simplest stochastic operation. A NOT-gate has one input and inverts all bits of the input bit-streams. The output probability is $\rho_z = 1 - \rho_x$. The arithmetic depends on the SN format and is shown in Equation (2.14) of Table 2.2. A computation example for a bit-stream with probability $\rho = \frac{3}{4}$ (SN $x = \frac{3}{4}$ in the unipolar format) is shown in Figure 2.4. The probability of observing 1s in the output bit-stream is $\frac{1}{4}$.

AND Operation If and only if both inputs are 1 at the same time, the output is 1. The output bit-stream has a probability of $\rho_z = \rho_x \rho_y$ for two input bit-streams with uncorrelated probabilities ρ_x and ρ_y. AND gate is the stochastic circuit for multiplication in the unipolar format. A graphical example for an AND-gate multiplication for $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$ is shown in Figure 2.5.

OR Operation The OR operation computes the union of the input probabilities $\rho_z = \rho_y \cup \rho_x = \rho_y + \rho_x - \rho_y \rho_x$. If at least one of the random bits at the inputs of OR-gate is 1, the output is 1. The arithmetic for SNs in bipolar and unipolar formats is listed in Equation (2.16). The OR operation can approximate the addition in the unipolar format if the input values are small ($xy << x + y$). Figure 2.6 shows an example for the inputs $\rho_x = \frac{1}{2}$ and $\rho_y = \frac{1}{4}$. The output is $z = x \cup y = \frac{5}{8} < x + y = \frac{6}{8}$.

MUX Operation A multiplexer (MUX) unit has at least three inputs. Two of them are data inputs, and one is the select input. The select input is not correlated to the two inputs and usually has a probability

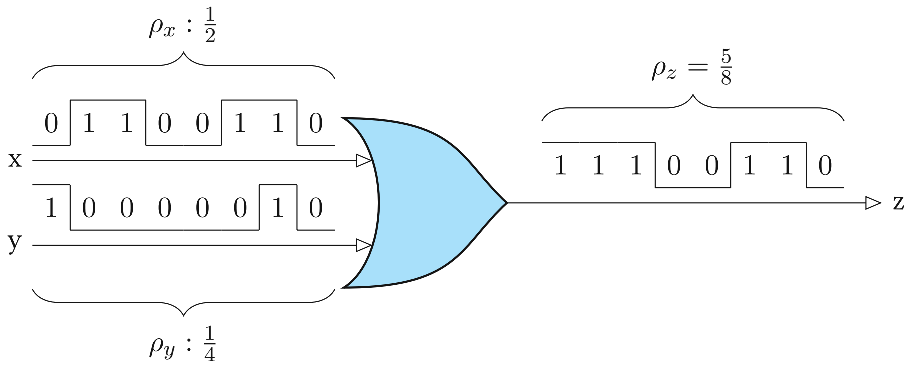

Figure 2.6: The OR-gate is the stochastic $\rho_z = \rho_x + \rho_y - \rho_x\rho_y$ operation

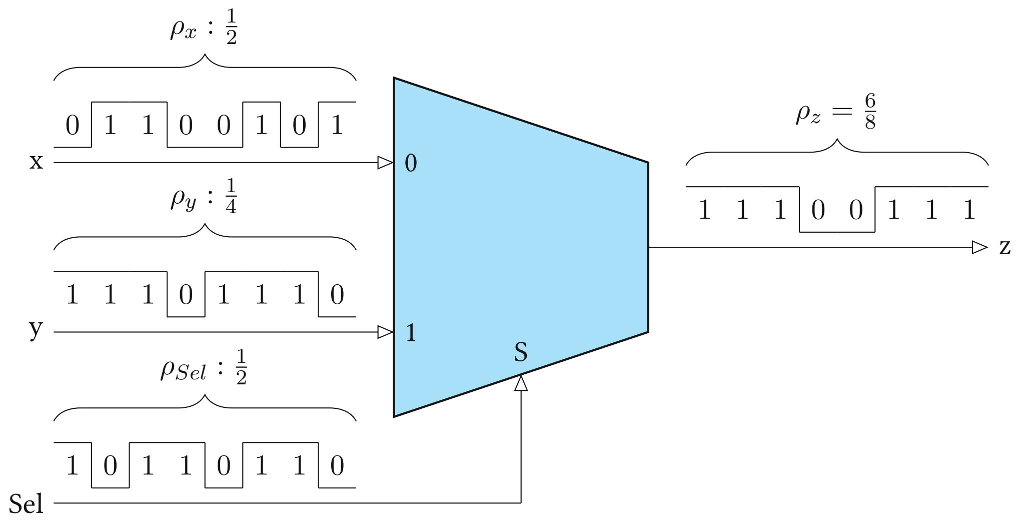

Figure 2.7: The MUX-gate is the stochastic $\rho_z = (1 - \rho_{Sel})\rho_x + \rho_{Sel}\rho_y$ operation

of $\rho_s = 0.5$. The current bit at the select input determines which of the two inputs is forwarded to the output. A MUX unit randomly mixes the two input bit-streams and computes a scaled addition of the two inputs for both unipolar and bipolar formats. In terms of probabilities, the output probability is $\rho_z = (1 - \rho_{Sel})\rho_x + \rho_{Sel}\rho_y$. Equation (2.15) in Table 2.2 lists the equation, and Figure 2.7 shows an example of scaled addition with a MUX unit.

XNOR Operation XNOR performs multiplication on bipolar bit-streams. Throughout this thesis, whenever we discuss AND-gates for unipolar representation, it also applies to XNOR-gates for bipolar bit-streams.

2.3.2 Exploiting Correlation in Stochastic Computing

The accuracy of arithmetic from Section 2.3.1 highly depends on the statistical properties of the input bit-streams. This section considers SC with positive ($SCC = +1$) or negative ($SCC = -1$) correlation

Table 2.2: SC arithmetic with two input SNs (x, y) and the output (z). The input bit-streams are uncorrelated.

Operation	Unipolar	Bipolar	
NOT	$z = 1 - x$	$z = -x$	(2.14)
AND	$z = xy$	$z = \frac{x+y+xy-1}{2}$	(2.15)
OR	$z = x + y - xy$	$z = \frac{x+y-xy+1}{2}$	(2.16)
MUX	$z = \frac{x+y}{2}$	$z = \frac{x+y}{2}$	(2.17)
XNOR	$z = 1 - x - y + 2xy$	$z = xy$	(2.18)

between bit-streams. Table 2.3 lists the arithmetic for correlated unipolar SNs for MUX units and OR-gates. An OR-gate can either compute the maximum value function or a saturating addition [31]. The MUX operation is unaffected by the correlation between data inputs but requires an uncorrelated select input.

Table 2.3: Unipolar SC Arithmetic with two correlated SNs (x, y).

Operation	Negatively Correlated $SCC(X,Y) = -1$	Positively Correlated $SCC(X,Y) = +1$	
OR	$z = \min(1, x + y)$	$z = \max(x, y)$	(2.19)
MUX	$z = \frac{x+y}{2}$	$z = \frac{x+y}{2}$	(2.20)

2.4 Deterministic Stochastic Computing

Truly random or pseudo-random numbers have been used since the early days of SC. Each bit of an N-bit SN is randomly chosen to be 1 with some probability. The results are usually not exact due to random fluctuations. This section discusses the deterministic methods of SC with the same arithmetic as random SC from Section 2.3. Recent research on deterministic SC showed that computation using SC constructs can be completely accurate. Besides quantization errors, running the computation for an exact number of clock cycles is the only constraint to produce completely accurate results with these deterministic methods. Different approaches have been proposed that compute on the same stochastic circuits but use different SNGs. SNGs based on low-discrepancy sequences (e.g., Sobol sequences) and unary bit-streams, are two common variants of deterministic SNs [16, 18].

In so-called *unary bit-streams*, the 1s are grouped at the end or beginning of the bit-stream. We can generate unary bit-streams by replacing the RNG (Figure 2.2) with an up-counting (or down-counting) counter. In the first clock cycles, the counter value will be less than the weighted binary input of the SNG, and the output bit-stream is 1 followed by a cluster of 0s because the counter value is greater than the weighted binary input. Usually, we generate more than one period of 1s followed by 0s, letting the counter overflow and repeat. The time-continuous equivalent of a periodic unary bit-stream is a pulse-width-modulated (PWM) signal. The similarities are shown in Figure 2.8. A PWM signal is defined by a duty cycle (D) and a frequency ($f = \frac{1}{period}$). The duty cycle is the fraction of time in which the signal is high and is equal to the probability of observing 1s in the time-discrete domain [20].

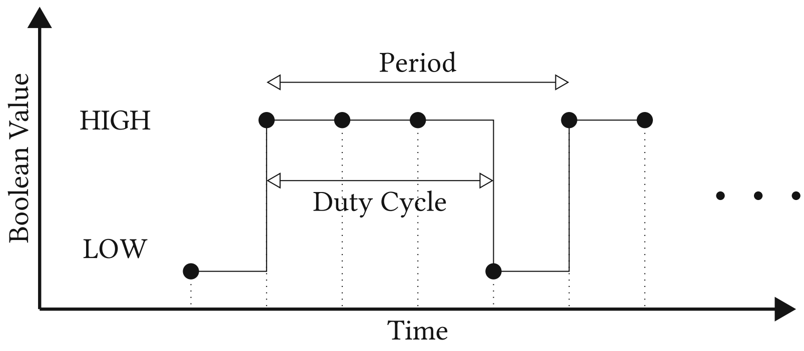

Figure 2.8: Encoding values into the duty-cycle of PWM signals and. Repeated deterministic unary bit-streams are equal to sampled PWM signals

Deterministic SC works with periodic, correlated or uncorrelated bit-streams. Each bit of a period of the first bit-stream should see each bit of a period of the second bit-stream exactly one time to fulfill the $SCC = 0$ property of Equation (2.8) (alternative to the SNG in Figure 2.3) [32]. The operation must run for the correct number of clock cycles, the product of both period lengths. Running the computation for more (fewer) cycles causes that any two bits meet each other multiple (zero) times. In this work, we focus on *relatively prime bit-streams* [32] when computing with uncorrelated bit-streams. We also discuss the *clock division* method because it has approximately the same hardware complexity as the relatively prime method [6, 16, 18].

The **relatively prime method** computes with unary bit-streams with relatively prime period lengths. In the example below, we generate one bit-stream with a period of 4-bits, the second with a period of 3 bits, and connect them to an AND-gate. The important point is that both periods are relatively prime (with $\mathrm{GCD}(4, 3) = 1$) and that the AND operation runs for the least common multiple (LCM) of both period lengths ($LCM(4, 3) = 12$). Remind that the AND operation in Equation (2.21) computes the multiplication for uncorrelated bit-streams [18].

$$\begin{aligned}\frac{1}{3} &= 100100100100\\ \frac{3}{4} &= 111011101110\\ \frac{1}{3}\times\frac{3}{4} = \frac{3}{12} &= 100000100100\end{aligned} \tag{2.21}$$

The **clock division method** with unary bit-stream uses two counters as RNGs with the same resolution but different increment rates. The first counter increments every clock cycle, and the second counter increments after N clock cycles, with N being the period of the first counter. Equation (2.22) shows the multiplication with the clock division method for $N = 4$

$$\begin{aligned}\frac{1}{4} &= 1000\,1000\,1000\,1000\\ \frac{3}{4} &= 1111\,1111\,1111\,0000\\ \frac{1}{4}\times\frac{3}{4} = \frac{3}{16} &= 1000\,1000\,1000\,0000.\end{aligned} \tag{2.22}$$

2.5 Summary

SC can be summarized as computing with probabilities encoded in discrete bit-streams or analog pulses. The distribution of 1s can be random, pseudo-random, low-discrepancy, or unary. The examples show that randomness is not mandatory for SC, and simple counter-based SNGs can be used to perform exact computations. The bit-stream lengths can be shorter compared to conventional random SC because there are no random fluctuations with deterministic SNs. The results of deterministic SC are predictable and can be completely accurate.

Chapter 3

Sound Source Localization

The previous chapter introduced the concept of replacing one part of a conventional processing system with an SC implementation. SSL algorithms exist for both frequency and time domain. Search-based SSL algorithms in the frequency domain are superior in locating multiple sources simultaneously but are more computationally intensive. We refer the readers to [15, 35, 36] for the frequency domain algorithms. In this thesis, we use a time domain algorithm without Fourier transforms and assume only one active source. Structurally, this chapter consists of three thematic sections. First, we discuss the concept of locating sound sources with multiple microphones. Next, we address three main components of sound source localizers. Finally, we present a simple SSL algorithm.

3.1 Sound Capture with Microphone Arrays

The microphone array is a set of closely positioned microphones packed in a single device. SSL starts with multiple microphones capturing sound waves. A single microphone cannot deal with 3D-Processes, such as reverberation and ambient noise, whereas multiple microphones can record sound sources spatially selective through beamforming [37, 38][1]. The sound processing system's accuracy, precision, and capabilities improve with the computational complexity of signal processing and the number of microphones in the array. In this section, we introduce technical terms related to sound source localizers.

3.1.1 Coordinate System and Direction of Arrival

The spherical coordinate system describes points and vectors in the three-dimensional space. Microphones and sound sources are defined relative to the center of a microphone array with elevation θ, azimuth φ and radius r. We use the convention that elevation is zero in the x, y plane and increases in

[1]Beamforming allows listening to the sound coming from one direction while suppressing the background noises and reverberation.

the z-axis. The elevation angle is often replaced by the polar angle measured from the z-axis so that the polar angle is $90 - \theta$. Equation (3.1) shows the conversion between the Spherical and Cartesian coordinate systems. *Small microphone arrays* estimate direction-of-arrivals (DOAs), a vector from the center of a microphone array to a sound source, instead of the absolute source locations. Equation (3.2) and Figure 3.1 show the definition and an illustration of a DOA vector.

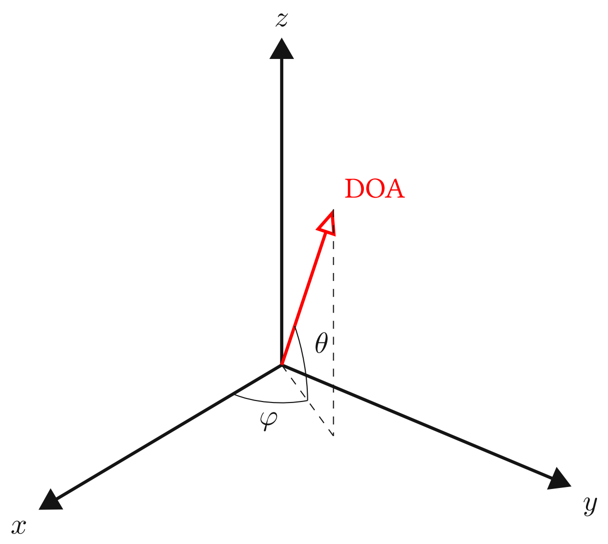

Figure 3.1: Visualization of azimuth and elevation (φ,θ) that define the DOA vector. The DOA points from the center of the microphone array to the sound source.

$$\vec{p} = \begin{bmatrix} r \cdot \cos\theta \cdot \cos\varphi \\ r \cdot \cos\theta \cdot \sin\varphi \\ r \cdot \sin\theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.1}$$

$$\vec{\varsigma} = \begin{bmatrix} \cos\theta \cdot \cos\varphi \\ \cos\theta \cdot \sin\varphi \\ \sin\theta \end{bmatrix} \tag{3.2}$$

3.1.2 Near- and Far-Field

Sound sources generate spherical sound waves that propagate away from sources. An observer close to the source (near-field) can measure the curvature of arriving sound waves. The curvature decreases for an observer far away (far-field), and the sound waves appear planar [35]. Microphone arrays in near-field conditions can estimate the curvature of arriving sound waves by calculating the frequencydependent phase delay at each microphone. The curvature of sound waves is proportional to the distance of sound sources. Generally, SSL with small microphone arrays in far-field conditions is limited to 1D or 2D DOA estimations without estimating the distance. However, DOA estimations of multiple small microphone arrays can be used to compute the absolute position of a sound source with triangulation.

Analyzing sound-pressure levels (SPLs) at each microphone is an alternative or complementary method to estimate distances. The SPL decreases with distance to sources. This method works in near-field and far-field behavior, but the accuracy decreases for sources in far-field. Subsequently, we assume far-field conditions and don't attempt to estimate the distance to sources.

3.1.3 Ambiguity of Sound Source Localization

A microphone array with two microphones can estimate angles in the $[-\pi/2, \pi/2]$ interval. For simplicity, we use the convention that both microphones and source are in the xy-plane, and the microphones are on the y-axis. Using this spatial distribution, the one-dimensional (1D) DOA vector $\varsigma_{1\text{D}}$ forms a right angle with the z-axis:

$$\vec{\varsigma}_{1\text{D}} = \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{bmatrix} \tag{3.3}$$

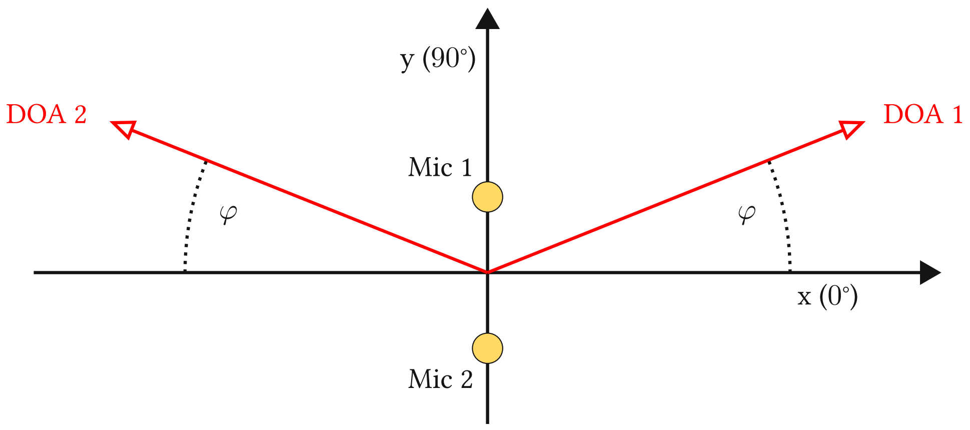

Figure 3.2: The coordinate system and the ambiguity in locating sound sources with two microphones. DOAs, with equal phase delay, form a *cone of uncertainty*.

Figure 3.2 shows a microphone array with two microphones. In the example, sound waves reach microphone one before they reach microphone two. The signal delay is equal for DOA1 and DOA2.

Any DOAs with the same azimuth φ (but any elevation θ) will result in the same time delay estimation (TDE). We can visualize the so-called cone of uncertainty by rotating the DOA vector in Figure 3.2 around the y-axis.

Spatial aliasing also causes ambiguity in SSL results. A microphone array can measure the same signal delay for multiple different DOAs, even if the θ angle is equal and we only consider sound sources in one half-plane. The concept of spatial aliasing is shown in Figure 3.3. The frequency of arriving sound waves is high enough to allow spatial aliasing. The microphones measure the same delay for $\varphi = 0$ and $\varphi = \varphi_1 \neq 0$.

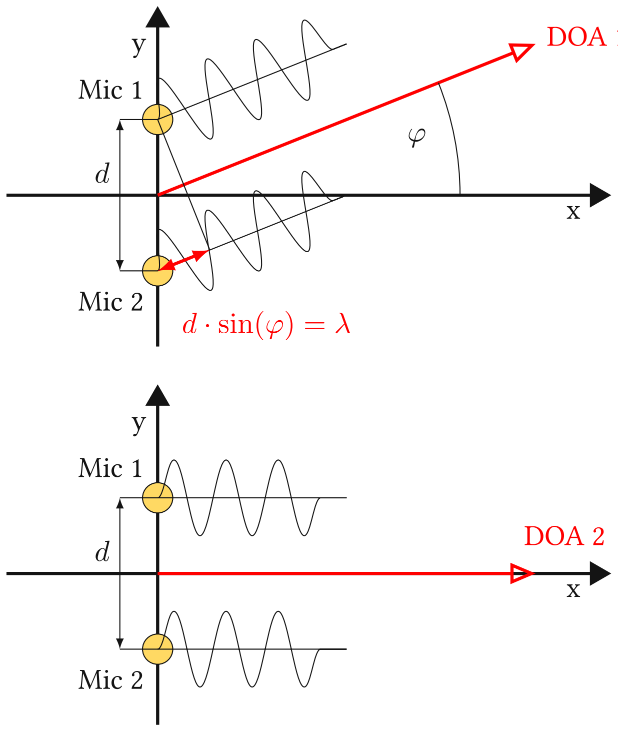

Figure 3.3: Spatial aliasing for high frequencies. DOA 1 and DOA 2 result in the same phase delay.

We observe spatial aliasing for sources that emit a sine wave with frequency $f_1 = \frac{v}{\lambda_1}$, if Equation (3.4) is fulfilled [15]

$$\pi \frac{d}{\lambda_1} \sin(\varphi_1) = \pi \frac{d}{\lambda_1} \sin(\varphi_2) + n2\pi \qquad , \varphi_1 \neq \varphi_2 \tag{3.4}$$

and $v = 343\,\mathrm{m\,s^{-1}}$ is the velocity of sound waves. Equation (3.4) is solvable when $d \leq \frac{\lambda_1}{2}$, as two different azimuth angles yield the same phase shift. We prevent spatial aliasing if the distance between microphones is smaller than half of the maximum wavelength. SSL can tolerate some spatial aliasing if the most dominant frequencies are below that threshold. Digital or analog lowpass filters are used before SSL to dampen high-frequency components and reduce spatial aliasing.

For example, the Amazon Echo Dot has six microphones arranged in a circle around one centered microphone. The distance between two microphones is $d = 77.8\,\mathrm{mm}$. Frequencies above $f = 2.2\,\mathrm{kHz}$ will cause partial aliasing

$$2d < \lambda = \frac{v}{f} \tag{3.5}$$

$$f < \frac{v}{2d} \tag{3.6}$$

$$f < \frac{343}{0.1556} = 2.2044\,\mathrm{kHz}. \tag{3.7}$$

SSL for speech is still possible because the most dominant frequencies of humans are below 2 kHz.

3.2 Sound Source Localizers

A sound source localizer system consists of three subsystems. The preprocessing subsystem usually segments, filters and weights the audio stream. The preprocessing subsystem splits the stream into short data blocks because the SSL algorithm calculates source locations based on short audio frames. Following that, an SSL algorithm processes a single frame when the voice activity detection (VAD) detects a valid signal. By averaging multiple one-frame-estimations or using knowledge about the room geometry and array position, an SSL post-processor can further enhance the precision of estimations [15]. The following subsections introduce the individual blocks in more detail. A specific example for the one-frame SSL algorithm is presented in Section 3.3.

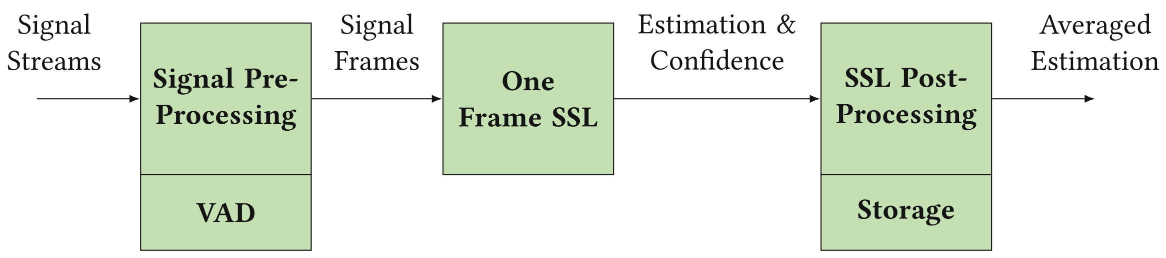

Figure 3.4: Block diagram of a sound source localizer

3.2.1 Preprocessing

The first subsystem of a sound source localizer prepares raw microphone signals for the SSL algorithm. The preprocessing usually includes low-pass or band-pass filtering, framing and VAD. Fourier transform and signal weighing are optional tasks depending on the application and SSL algorithm.

Signal filtering: Most devices locate human speakers. The frequency band for human speech lies below $\approx 6\,\text{kHz}$, but the main signal power is below $2\,\text{kHz}$. We extract the relevant frequency band and avoid disturbance through background noise or localization of other active sound sources with filters. Filtering further reduces spatial aliasing.

Framing: Two reasons for segmenting streams into data blocks are 1) moving sound sources and 2) efficiency. First, when a speaker moves from A to B, the location estimation is only valid for a limited period. Second, many signal processing algorithms are more efficient when using data blocks instead of continuous streams. The window length N_{frame} is a trade-off between responsiveness, computational effort and accuracy. A larger block size increases computational demands and latency of computation [35]. The motion of the source should be negligible during one frame (window length multiplied by the sample period). The simplest segmentation is a multiplication with a rectangular function. The rectangular function function is 1 for N_{frame} samples and 0 otherwise. We can increase the accuracy of the SSL if we avoid the sharp transitions (high-frequency components) by multiplying the audio frames with a window function, for example, the Hanning window

$$w_H[n] = \begin{cases} 0.5[1-\cos(\frac{2\pi n}{N_{frame}-1})] & : 0 \leq n < N_{frame} \\ 0 & : \text{else.} \end{cases} \tag{3.8}$$

Voice Activity Detection: VAD avoids inaccurate source estimations from breaks between words and sentences. Also, some words are quiet and not suitable for correct location estimations. An example VAD algorithm is an averaging filter that averages the absolute value of the microphone signals and compares it with a certain threshold.

Weighting: The phase-transform performs well in realistic and reverberate environments [36]. It is robust to reverberation but sub-optimal under reverberation-free conditions. It is usually computed in the frequency domain, but it can also be calculated in the time domain [39].

3.2.2 One Frame Sound Source Localization

The theory of SSL can be divided into algorithms that locate sources in either one or two processing steps. One-step methods are based on the steered response power. These algorithms form a beam and steer over predefined spatial points, for example, a 5° grid. Algorithms based on the steered response power require a Fourier transform in the preprocessing subsystem because they compute the source location in the frequency domain. Two-step methods rely on TDEs for each microphone pair [15, 36]. This thesis focuses on two-step methods that are further discussed in Section 3.3.

3.2.3 Post Processing

The post-processor is the final subsystem of a sound source localizer that receives the one-frame SSL estimations and optional confidence levels. The post-processor uses application-specific information and previous one-frame location estimations to enhance the accuracy of the sound source localizer. Application-specific information can, for example, include the location of the microphone array. If the microphone array is mounted on a wall, it can filter out false estimations that come from behind. The SSL post-processor has access to previous location estimations and can average over multiple frame estimations. If the microphone array tracks a single speaker, subsequent valid estimations should approximately be in the same area. If a recent location estimation is far from the current average location, it can be identified as false.

3.3 Sound Source Localization Based on Time Delay Estimation

Algorithms based on TDE use the propagation speed of sound waves to localize sources [40]. A sound wave reaches each microphone with a relative delay. The time delay is calculated pairwise for each microphone pair. Note that an M-microphone array has $N_{pairs} = \frac{M(M-1)}{2}$ unique pairs. Selecting two microphones m_1, m_2 (that lie on the y-axis), a sound wave in the far-field reaches microphone m_2 with a delay that depends on the azimuth angle φ of the DOA:

$$\tau_d = \frac{d}{v} \sin \varphi, \tag{3.9}$$

where d is the distance between microphones.

3.3.1 Cross-Correlation

An intuitive computing approach for TDEs is the cross-correlation (CC) function. CC indicates the similarity signals. The function has a sharp peak at a position proportional to the time delay. For discrete inputs, CC is defined as

$$c[n] = (\mathbf{x_1} \star \mathbf{x_2})[n] \overset{\text{def}}{=} \sum_{i=-\infty}^{\infty} x_1^*[i] x_2[i+n] \tag{3.10}$$

where x_1^* is the complex conjugate of signal x_1. Since microphone signals have no imaginary component, we can set $x_1^* = x_1$. If $c[i_{max}]$ is the maximum value of the CC function and i_{max} is different from zero, then audio waves reached microphone 1 before or after microphone 2. If the time delay between the microphone signals increases, then $|i_{max}|$ increases. The maximum time delay is reached when both microphones and the source are lined up. The TDE τ_d is used in Equation (3.9) to calculate the azimuth (φ) of the DOA:

$$\tau_d = \frac{i_{max}}{f_s} \tag{3.11}$$

$$\varphi = \arcsin\frac{\tau_d \cdot v}{d} = \arcsin\frac{i_{max} \cdot v}{f_s \cdot d} \tag{3.12}$$

Equation (3.12) results from of a simple one-frame SSL algorithm that uses two microphones. Spatial aliasing causes a broader peak in the CC function. The CC function always has one clear peak if dominant frequencies are below the spatial aliasing threshold ($d \leq \frac{\lambda}{2}$) and only damped frequencies cause spatial aliasing. A larger microphone array can calculate source locations in a closed-form [41] or search-based. For search-based algorithms, the final location estimation is the position with the minimum root-mean-square error of calculated and measured TDEs. For a detailed analysis, see [15, 35, 42].

3.3.2 Resolution

The resolution of CC is determined by the distance between the microphones and the sampling frequency. For example, $d = 0.066\,\text{m}$ between two microphones and a sample rate of $f_s = 15.6\,\text{kHz}$ results in $2 \times N_{MaxLag} + 1 = 7$ different possible time delays

$$\tau_{d,max} = \frac{d}{v} = 192\,\mu\text{s} \tag{3.13}$$

$$N_{MaxLag} = \tau_{d,max} * f_s = 3 \tag{3.14}$$

with $\tau_{d,max}$ being the maximal time delay. A maxima of the CC at $c[i_{max} = N_{MaxLag}]$ corresponds to a time delay of $\tau_d = i_{max}/f_s = 192\,\mu\text{s}$, whereas a maxima at $c[i_{max} = N_{MaxLag} - 1]$ means a time delay of $128\,\mu\text{s}$. The resolution is limited to $64\,\mu\text{s}$. Interpolating the CC function increases the resolution with additional computational effort. We can interpolate the maximum between two samples instead of using the closest sampled value $c[i_{max}]$.

3.4 Summary

This chapter showed that we need at least two microphones to locate sound sources. For two microphones, the SSL is synonymous with one-dimensional DOA estimation. The one-frame SSL processing block is the core of the sound source localizer and computes the CC function of both microphone signals. Next, the SSL algorithm searches the peak of the CC result. The peak index (i_{max}) is forwarded to the SSL post-processing block.

Chapter 4

Research Method

The previous two chapters introduced important concepts of SC and SSL. In summary, SC is an alternative computing paradigm, and SSL is one specific signal processing task. So far, both research fields have not been brought together in practical applications or theory. In this chapter, we explain the research contribution of applying SC to SSL. This chapter starts with the research question, followed by an explanation and justification. We then discuss the hypotheses emerging from the research question and my research methods to test them. In the third section, we explain the target SSL architecture for this thesis and discuss the relevant decisions made during research design. In the final section of this chapter, we argue for the contribution and summarize the novelties of this thesis.

4.1 Thesis Statement

This thesis aims to apply SC to the field of SSL. Can we gain an advantage by considering SC during the design of a sound source localizer? The consideration of SC should result in measurable and practical advantages over solely using conventional computation and thus should motivate further research for more practical applications of SC.

Justification: SC is a promising computing paradigm that currently is little used in practical applications. The consideration of SC in a new application could be a research question on its own given the multiple variants of SC that can be used. Given the limited scope of a MSc thesis, we choose one specific SSL algorithm and one variant of SC that sounded most promising. The focus on a representative example allows a detailed analysis of advantages and disadvantages and can answer whether it is worth the additional design effort.

4.2 Research Hypotheses

Not all properties and characteristics of a computing system are equally important. A single disadvantage can outweigh multiple advantages, so it is essential to consider a broad spectrum and justify their relevance. The analysis and evaluation in later chapters are based on the hypotheses below. We first state the individual hypothesis, followed by a justification and description. Below each hypothesis, we describe and justify the research methods used to test the statements.

4.2.1 Resource Usage

The first hypothesis asserts that a sound source localizer's power and area consumption can be reduced using SC. Saving resources in computation systems is an active field of research and crucial for modern battery-powered devices. Mobile phones, laptops and personal assistants require the speaker location to enhance their functionality.

First, we implement the SC and conventional designs in a standard hardware description language (HDL) (very-high-speed integrated circuit hardware description language (VHDL)). We then use the synthesis reports of VIVADO (Xilinx field-programmable gate array (FPGA) development tool) and Synopsys Design Compiler (Development tool for application-specific integrated circuits (ASICs) to compare the resource consumption. Two independent tools with two different technologies (FPGAs and ASICs) allow an unbiased comparison as both technologies have strengths and weaknesses.

4.2.2 Precision and Computation Speed

The second hypothesis claims that being limited to low precision and slow computations (disadvantages of SC) are no drawbacks in SSL. A disadvantage of SC is that higher accuracy requires exponentially longer bit-streams and processing times. Lower computation precision should not limit the accuracy of location estimations.

We will study the effect of low precision computations by analyzing the accuracy of location estimations. We will show accuracy results for 3,4,5, and 8-bit precision inputs. If the results are still accurate in low precision implementations, then the exponential relation between bit-stream length and accuracy of SC is tolerable. Longer bit-streams (higher precision) require faster clock speeds of the stochastic circuit to keep up with the sampling rate of the audio data. Faster clocks, however, increase the dynamic power consumption, which will be studied and evaluated during power analysis.

4.2.3 Approximate Computing

We further argue that we can use approximate functions in SC and still get accurate source estimations. Some operations in SC are exact, and others are approximate. In this thesis, we analyze if SSL can tolerate approximate computing. Approximate circuits are often smaller and consume less power than exact implementations.

One version of the sound source localizer with SC uses an OR-gate to accumulate products of the CC function. We will compare the localization accuracy of approximate implementations with exact results. The OR-gate as an approximate adder in unipolar SC is a well-known example of saving resources. It is of interest if approximation negatively affects the accuracy of SSL estimations. SSL requires the maximum position in the CC result but not the exact value unless we interpolate in a post-processing step.

4.2.4 Near-Sensor Processing

We further test if redesigning the analog signal processing path for SC results in power savings and lower complexity. SC requires some pre-processing, which could be more efficient in the analog domain, for example generating bit-streams. ADCs are not mandatory if the bit-streams are generated in the analog domain.

Potential advantages in near-sensor processing will be analyzed by building a custom analog board, discussed in Chapter 5. We will do power measurements (Chapter 6) and theoretical analysis in Chapter 7.

4.2.5 Design Complexity

The fifth hypothesis is that adding SC to SSL does not necessarily increase the design complexity and general usability. The design complexity and constraints with SC are comparable with conventional only designs. Highly custom and optimized designs are generally more efficient than generic solutions. Additional effort or design constraints should not outweigh the advantages of using SC.

All digital stochastic circuits should use Complementary metal-oxide-semiconductor (CMOS) technology and synchronous circuits. Chapter 7 will evaluate the design from a system perspective and compare general constraints like input counts and scalability. As this thesis analyzes a new practical application for SC, it is essential to be compatible with existing widespread technologies.

4.2.6 Properties excluded from analysis

The following properties and topics are not covered in this thesis.

1. SC is **error-tolerant**. Small fabrication nodes and low power make computations susceptible to bit flips (a faulty transition from zero to one or the other way around). In SC, a bit flip is always as severe as a bit flip in an equivalent conventional computation circuit's least significant bit (LSB). Error tolerance, however, is already well-known, and sufficient research exists. There is no reason to assume that error tolerance is more critical for SSL than for other applications
2. **SC can save power by exploiting progressive precision.** SC approaches the correct result during computation. As the computation time progresses, the result becomes more accurate. This behavior is known as progressive precision. The accuracy can be controlled dynamically when lower precision is required, for example, during the power-saving mode.
   Progressive precision is higher for certain types of SC than others. This thesis focuses on deterministic unary bit-streams with relatively low progressive precision. Preemptively terminating the computation would cause high errors and save relatively little power.

4.3 Research Design and Justification

To answer the research question and test some of the hypotheses from the previous section, we design a simple representative sound source localizer with the SSL architecture based on Figure 3.4. The design uses two microphones, analog signal processing, digital TDE with CC, an average filter as SSL post-processing, and light emitting diodes (LEDs) as outputs. The design diagram is shown in Figure 4.1 and briefly explained in Section 4.3.1. Following that, the justification of the five most important design decisions is outlined in Section 4.3.2.

4.3.1 System Design

Figure 4.1 shows an overview block diagram of the sound source localizer used in this thesis. The thesis focuses on the CC and Custom and Reference Analog processing blocks. The signal processing starts with a custom analog processing that generates the PWMs of two microphone signals. Those signals are sampled by an FPGA and thus converted to periodic unary bit-streams. We use a serial peripheral interface (SPI)-controlled ADC for reference as an alternative path. We will analyze and evaluate both analog paths in more detail in Section 5.1.1. The remaining signal processing happens within an FPGA (Xilinx Zedboard). The development board is equipped with a Zynq-7000 FPGA. In the digital domain, we implement different versions of the CC, which we describe in Section 5.2.1 and analyze regarding accuracy and resource consumption in Chapter 6. The focus lies on comparing the proposed stochastic implementations with the reference implementation that uses weighted binary arithmetic. After the CC, the subsequent computation block searches for the maximum $c[i_{max}]$ in the centered K values of

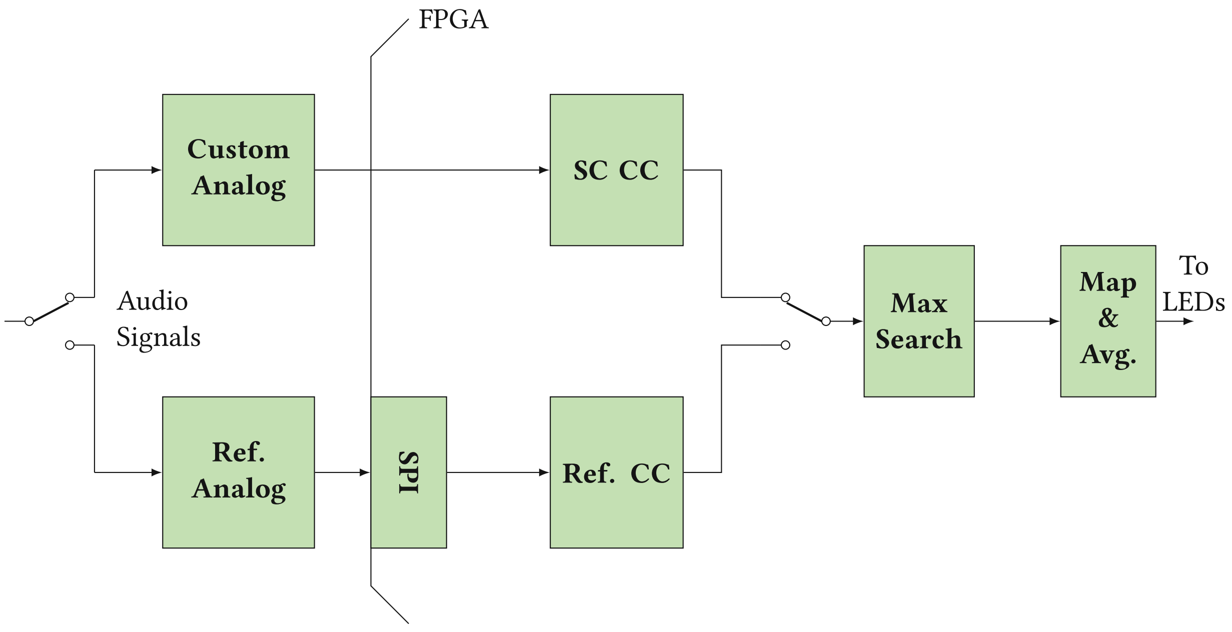

Figure 4.1: Thesis implementation overview. Block diagram of the signal processing chain.

the CC, as i_{max} is proportional to the time delay (see Section 3.3.1). The last processing block computes the average of multiple TDEs. The output of the digital design is an averaged TDE. We linearly map the TDE to 15 individually addressable LEDs on a semicircle, so they point towards the direction of the source location.

4.3.2 Design Decisions

This section explains and justifies my decisions during design. There are multiple variants of SC (random or deterministic) and SSL (SSL based on time-delay or steered response power). The following list of design decisions starts with why SSL is chosen as the target application and ends with the exact variant of SC used in the remainder of this thesis.

1. SSL as the target application.
   (a) Audio data is relatively slow compared to other signal processing tasks. Since human voice is limited to a low-frequency band, we can use 12 kHz as sampling frequency to fulfill the Nyquist criteria. Therefore, the processing can also be slow, making human voice processing a well-suited application for SC.
   (b) SSL is a widespread processing task in our modern devices, with increasing demand.
2. We apply SC to the category of SSL based on TDEs instead of algorithms based on steered response power.

   (a) SSL algorithms based on TDE are usually faster (real-time) but less accurate than algorithms based on the steered response power. Since SC performs better at lower precision it is more natural to go for the lower accuracy real-time variant.
   (b) SSL algorithms based on the steered response power require fast Fourier transform (FFT) before computing the source location. Even if researchers found a method to compute an FFT in SC [43], the transformation alone is more complex than some SSL algorithms based on TDEs.
   (c) In this thesis, we compute CC with stochastic circuits to calculate the time delay between signals (through a simple maximum search of K-centered CC values). The TDE can then be used for a wide range of advanced SSL algorithms for different numbers and spatial distribution of microphones. After showing that SC works for two microphones, the concept can quickly be expanded to microphone arrays with multiple microphones and more advanced algorithms.

3. We implement the TDE using SC instead of other blocks in the SSL algorithm, like interpolation or pre-processing of audio signals.
   (a) TDE is a part of every two-step SSL algorithm (also called *algorithms based on TDE*). TDEs are also used in other applications such as radar ranging, wireless location, sonar direction finding, beamforming, sensor calibration and many more. In other research fields, similar terminologies for TDE include time difference of arrival estimation, time of arrival estimation, and time of flight estimation. The successful implementation of the CC function could be mapped to other applications.
   (b) TDE in the time domain is a maximum search over the CC result of two signals. The CC function can be broken down into a set of multiply-accumulate (MAC) operations. Since SC shines at computing multiplication, saving resources compared to conventional computations is likely.
   (c) Other computation blocks in the SSL processing chain are optional (most pre-processing blocks in Section 3.2.1) and solely enhance the accuracy. It seems natural to start working on the mandatory CC function before targeting optional processing blocks.

4. We use deterministic periodic unary bit-streams for SC computation.
   (a) We chose deterministic bit-streams instead of random (stochastic) bit-streams as randomness is not required for the most advantageous properties of SC (see Section 2.4). As Devon Jenson and Marc Riedel say in [16] "We conclude that there is no clear reason to compute with stochastic bit-streams. Even when randomness is free,...". We don't have access to

a *free* random number generator and implementing one would unnecessarily increase the resource consumption and complexity of the sound source localizer.

(b) We choose unary bit-streams (1s are stacked) for the following reasons.

   i. The generation of bit-streams often consumes more resources than the stochastic circuit that does the actual computation. An SNG for unary bit-streams uses a counter and a comparator and should require less or equal resources than other deterministic SNGs.

   ii. Using two analog-to-time converters (PWM signal generators) instead of first converting voltages to conventional binary values followed by digital bit-stream generation could open new possibilities for saving resources (Near-Sensor Processing).

5. We focus on the unipolar stochastic representation. This decision seems counter-intuitive as audio data has positive and negative half-waves, making a signed representation (bipolar in SC) more natural.

   (a) The reason lies in the limited adding capabilities of the bipolar representation (see the analysis in Section 5.2.1).

   (b) In the unipolar representation, we can use the property of OR-gates to approximate or even exactly calculate the (non-scaled) sum.

   To use the unipolar representation, we can either re-scale the original signal to the [0,1] interval with a mean of 0.5 or only process positive half-waves of the audio signal. We also analyze a digital implementation that uses a counter for summation (so only multiplication is done with stochastic circuits). For that scenario, we can also use bipolar representation as stochastic multiplication works well for unipolar and bipolar encoded bit-streams.

4.4 Contributions

The first and main contribution is SSL as a new application for SC. Second, this work contributes to near-sensor processing in general. We demonstrate that using time-encoded PWM data is possible with stochastic circuits, and therefore, SC does not rely on conventional ADCs. The third contribution of the thesis, on a theoretical level, is to show the potentials and challenges of deterministic multi-stage stochastic circuits. We show that properties of deterministic, stochastic multiplication can be used to design an accurate MAC unit used in the CC function of TDE. The fourth contribution is the practical implementation of an SSL based on SC. We compare its resource consumption with conventional binary processing in both analog and digital domains to demonstrate the advantages of this work. In summary, this thesis contains the following four novelties.

1. The first application of SC to SSL.
2. Practical usage of PWM modules for SC (instead of ADCs).
3. New SC-based MAC, published in [44]. We attached the paper as Appendix C.
4. Implementing a new, efficient sound source localizer with SC.

Chapter 5

Implementation

In Section 4.3.1, we introduced the target SSL architecture, which we discuss in more detail in this chapter. We cover the complete signal flow, starting with microphones capturing sound waves and ending up in an indication of the source location. The sound source localizer in Figure 4.1 uses two microphones and is limited to one-dimensional SSL. Subsequently, we use SSL as a synonym for one-dimensional DOA estimation (see Section 3.1.3). In the following two sections, we discuss the implementations in the analog (Section 5.1) and digital domain (Section 5.2). This chapter contains all implementation variants for real-time demonstrations, analysis (Chapter 6) and discussion (Chapter 7). We use discrete components such as capacitors, resistors and operational amplifiers (OPAMPs) for analog processing and a Digilent Zedboard with an FPGA for digital computations.

5.1 Analog Implementation

In what follows, we describe the analog circuitry built, mainly to handle inputs and outputs of the sound source localizer. Section 5.1.1 covers the analog signal processing that happens before SSL. One signal path is for conventional processing, and a second is specially designed for SC. Section 5.1.2 explains how we implement the input and output of the SSL system, the sound capture and visualization of the estimated source location.

5.1.1 Analog Board

One contribution of this thesis is the Analog Board presented in this section. The Analog Board requires a 5 V direct current (DC) power supply. The board takes two audio signals as inputs and has two outputs. The first is a conventional SPI, and the second is a unidirectional interface with two channels that carry the PWM signals for SC (more details are inserted later in this section). In Figure 5.1, the signals flow from left to right with the input audio jack on the left-hand side and the Peripheremodule

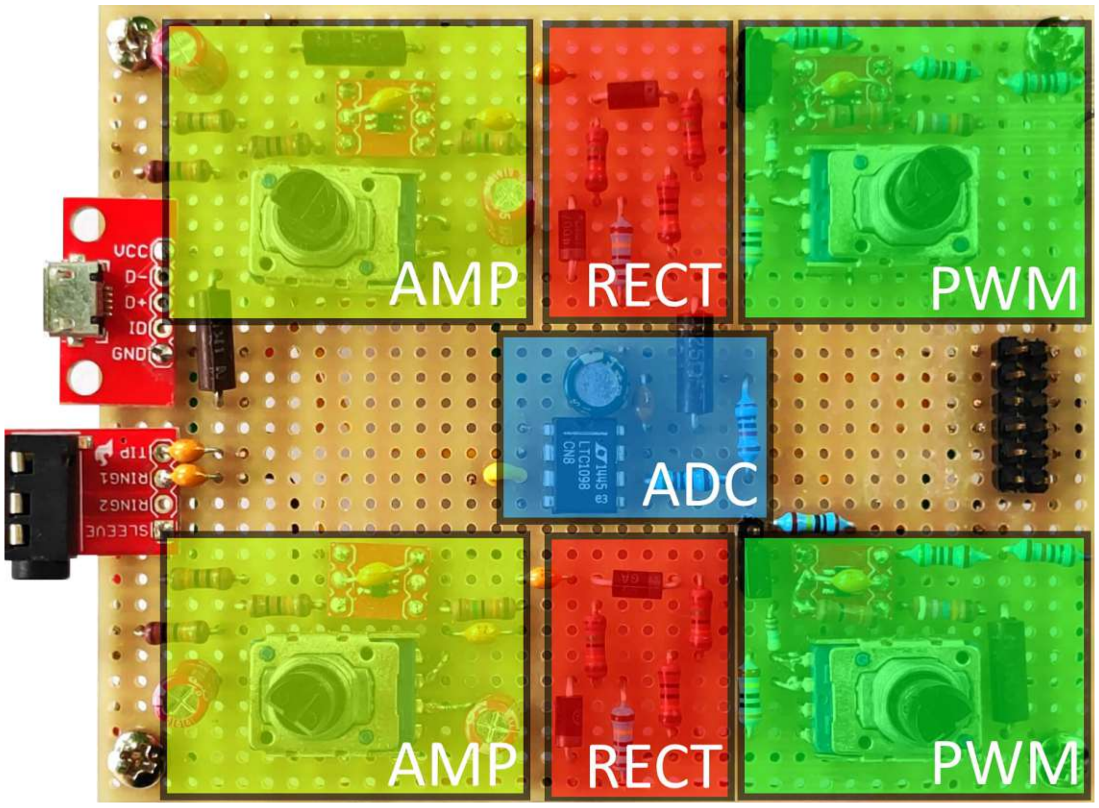

Figure 5.1: Analog Board for two audio signals. The amplifier, half-wave rectifier, PWM generation and ADC circuits are marked yellow, red, green and blue.

(PMOD) interface (Digilent Inc.) to the FPGA on the right-hand side. The PMOD interface combines both outputs to a single cable. The signal processing chain for the left and the right audio channel is on the top and bottom of the Analog Board. Below, we will explain the signal flow and the schematic of the Analog Board.

1) The audio input to the Analog Board connects with a stereo 3.5 mm audio jack on the left-hand side in Figure 5.1. In this thesis, we use two microphone signals (see Section 5.1.2) and a stereo sound card (for repeatable hardware tests with pre-recorded audio data) as audio sources.

2) The stereo signal from the audio jack is fed into two alternating current (AC)-coupled, non-inverting amplifiers, marked with yellow rectangles in Figure 5.1. The circuit uses a Texas Instrument OPA322 OPAMP and is shown in Figure 5.2. The two 100 kΩ resistors at the input of Figure 5.2 bias the input AC signal, and the capacitor decouples the biasing network from the supply. A 10 pF capacitor in the feedback path decouples the AC amplification from the DC operating point at the output stage. The amplifier circuit fulfills three tasks. First, the circuit brings the OPAMPs operating point to 2.5 V (mid-supply) for a maximum voltage output swing of 0 to 5 V. Second, the circuit has an adjustable gain in the $[1.15, 151]$ interval using a potentiometer with 1 kΩ to 1000 kΩ. Third, resistors and capacitors work as low and high-pass filters that attenuate noise and pass-through human speech. The amplified

signal is forwarded to an ADC and two half-wave rectifier circuit.

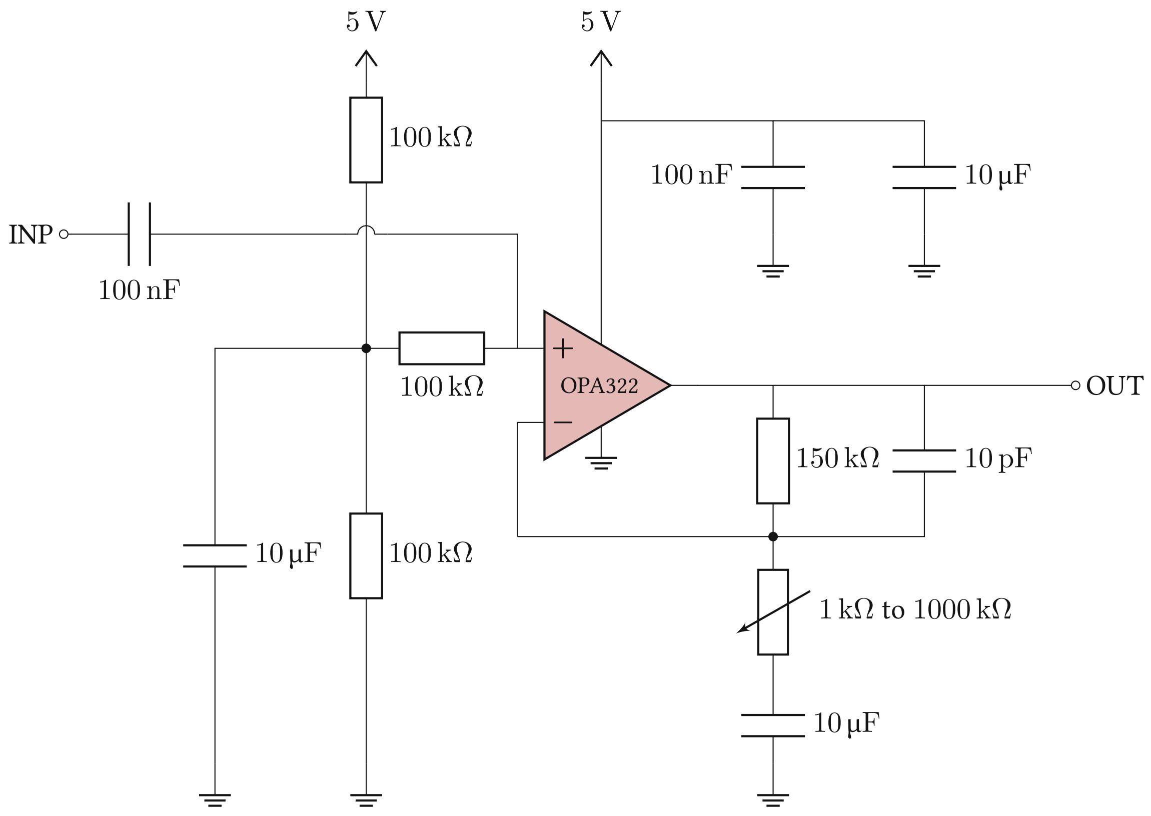

Figure 5.2: Adjustable, non-inverting, analog amplifier circuit. It uses passive components and a Texas Instrument OPA322. The circuit is marked with yellow in Figure 5.1

3) The LTC1098 is an 8-bit ADC with two input channels. The circuit is shown in Figure 5.3 and is marked with blue in Figure 5.1. The LTC1098 offers a channel selection through software, is used for both (amplified) microphone signals and connects to the FPGA over an SPI. The pins driven by the FPGA (master out slave in (MOSI), chip select (CS), serial clock (SCLK)) are in the [0 V,3.3 V] interval and directly connected to the ADC. The master in slave out (MISO) requires a 5 V to 3.3 V voltage divider to protect the input pin of the FPGA.

Besides the ADC, the amplifier outputs connect to two half-wave rectifiers with the schematic shown in Figure 5.4. The 100 nF AC coupling capacitor at the input of Figure 5.4 separates the 2.5 V DC offset from the AC audio signal. The AC component of the input signal does not have to overcome the threshold voltage of Diode D2 because of the bias created by Diode D1 in combination with a 150 kΩ and 1 MΩ resistor. A small positive signal at the input passes through D2, and negative voltages are blocked. The voltage divider at the output limits the positive swing to 1 V for the subsequent PWM module.

4) The PWM circuits, shown in Figure 5.5, receive the inputs from the half-wave rectifiers and are built with Analog Devices LTC6992 microchips. They are located on the right-hand side of the Analog

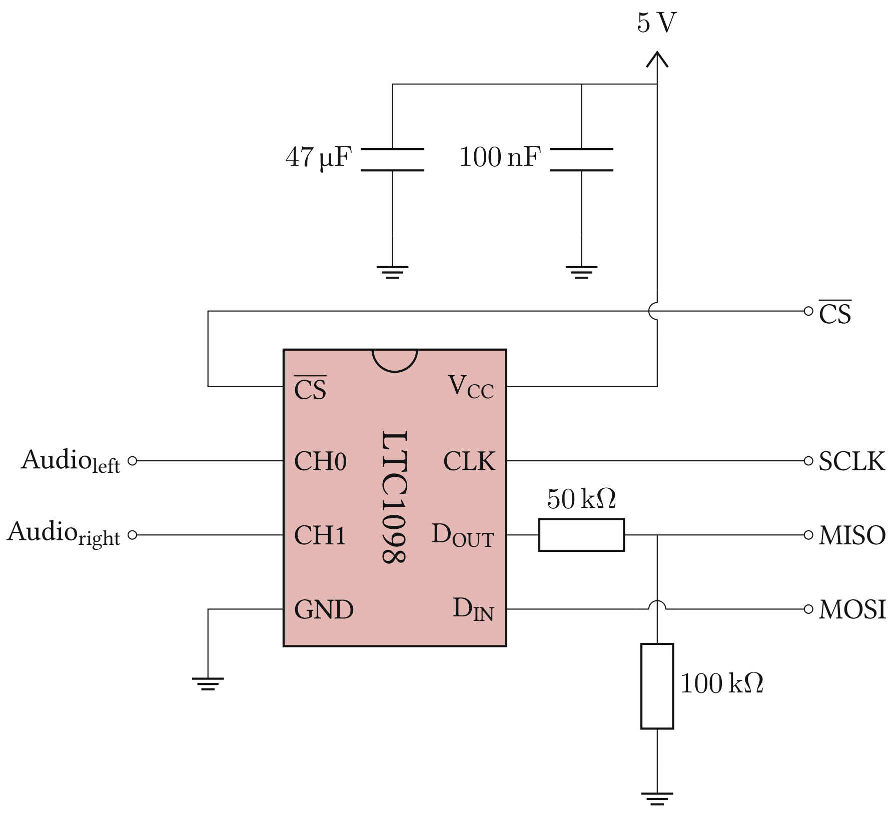

Figure 5.3: Schematic of the ADC (LTC1098) circuitry with two input channels and an SPI. The circuit is marked with blue in the center of Figure 5.1.

Board and marked green in Figure 5.1. The voltage level at the INP pin of the LTC6992 controls the duty cycle of the generated PWM signal, with a non-linear transfer. The duty cycle is $D = 0\,\%$ for input voltages in the $0\,\mathrm{V}$ to $0.1\,\mathrm{V}$ interval and $D = 100\,\%$ for input voltages in the $0.9\,\mathrm{V}$ to $1\,\mathrm{V}$ interval. The transfer function is $D_{out} = (V_{Mod} - 0.1) \times 1.25$ for V_{mod} in the $0.1\,\mathrm{V}$ to $0.9\,\mathrm{V}$ interval. We will analyze the effect of the non-linear conversion in Chapter 6. Figure 2.8 shows a PWM signal with $D = 75\,\%$, representing a voltage of $V_{mod} = \frac{0.75}{1.25} + 0.1 = 0.7\,\mathrm{V}$. The passive components at the SET and DIV inputs determine the frequency of the PWM signal. The potentiometer at the SET input allows adjusting the frequency in the $15\,\mathrm{kHz}$ to $250\,\mathrm{kHz}$ interval. If not stated otherwise, one of the two PWM circuits is set to $f_{PWM,1} = 234\,\mathrm{kHz}$, and the second is set to $f_{PWM,2} = 249.6\,\mathrm{kHz}$. The two PWM signals are sampled with a $3.744\,\mathrm{MHz}$ clock at the input register of the FPGA, which results in bit-streams with relatively prime periods of $n = 16$ and $k = 15$

$$n = \frac{f_{clk}}{f_{PWM,1}} = \frac{3.744 \cdot 10^6}{234 \cdot 10^3} = 16 \tag{5.1}$$

$$k = \frac{f_{clk}}{f_{PWM,2}} = \frac{3.744 \cdot 10^6}{249.6 \cdot 10^3} = 15. \tag{5.2}$$

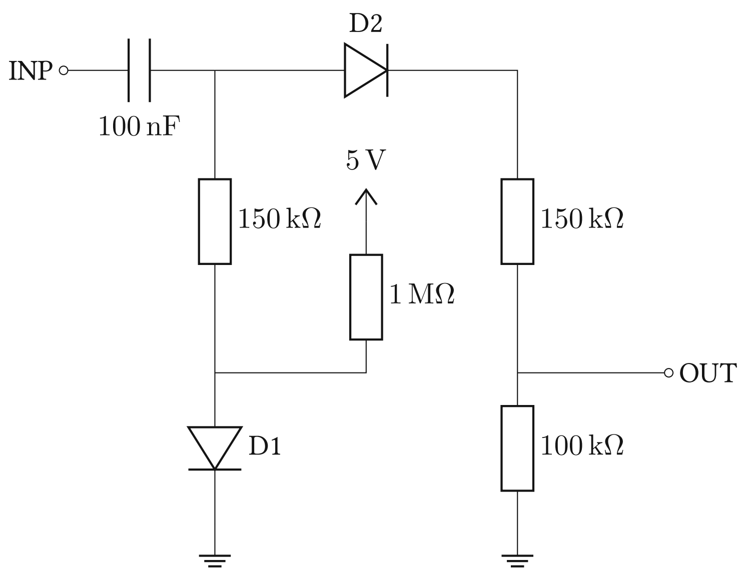

Figure 5.4: Schematic of the half-wave rectifier circuit, marked with red in Figure 5.1.

For the exact stochastic multiplication, we need 15 periods of $f_{PWM,1}$ and 16 periods of the $f_{PWM,2}$ (see Section 2.4), which brings us to the equivalent audio sampling frequency

$$f_s = \frac{f_{PWM,1}}{15} = \frac{f_{PWM,2}}{16} = 15.6\,\mathrm{kHz}, \tag{5.3}$$

also used by the ADC. A voltage divider after the LTC6992 reduces the signal voltage (5 V to 3.3 V) because the FPGA pins are limited to 3.3 V.

5) The outputs of the Analog Board are connected to the FPGA through a PMOD cable. The PMOD connection has a total of 12 pins. Eight of them are reserved for signals, two are connected to the ground and two are for power (3.3 V at pin 6 and 12). The Analog Board uses both ground pins and six signal pins. Four pins are used for the SPI protocol, and two lanes carry the outputs of the PWM circuits.

5.1.2 Input/Output Board

This section presents the Input/Output Board shown in Figure 5.6. The board connects to the Analog Board via a stereo cable and to the Zedboard through a PMOD cable.

The Input/Output Board fulfills two tasks. First, it provides audio signals to the Analog Board via two electret microphones. The microphones are soldered to two ADAFRUIT MAX4466 and are mounted to the Input/Output Board with a 90° angle. The ADAFRUIT MAX4466 is a fully assembled printed circuit board with an electret microphone and an amplifier circuit with the schematic shown in Figure A.1 (Appendix A). Both ADAFRUIT MAX4466 are marked blue in Figure 5.6. The outputs of the

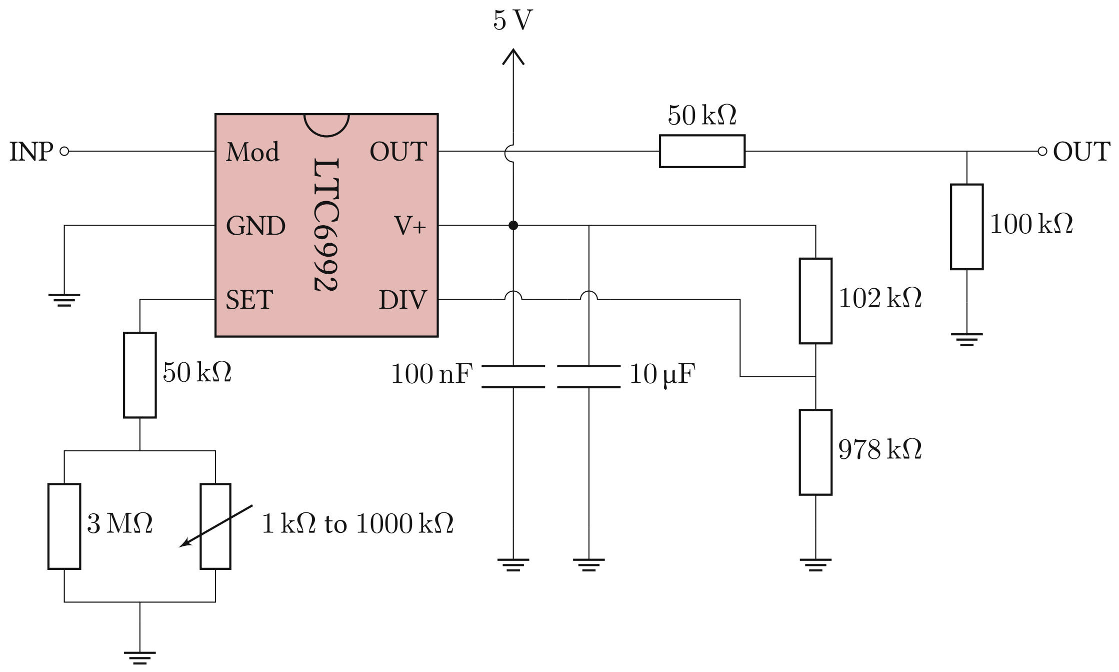

Figure 5.5: The LTC6992 is a voltage-controlled pulse-width-modulator from Analog Devices. The circuit converts voltages to PWM signals for unary SC and is marked green in Figure 5.1.

ADAFRUIT MAX4466 are single-ended microphone signals forwarded to the Analog Board through a 3.5 mm jack located at the top of Figure 5.6.

Second, the Input/Output Board visualizes the SSL output through 15 individually controllable LEDs (SK6812). Figure 5.6 on the bottom shows a LED strip mounted on a half-circle. A speaker in front of the Input/Output Board sees a lighted LED pointing in his direction because the spatial distribution of the LEDs matches the coordinate system of the microphone array. The origin of the coordinate system is between the microphones on the $\pm 90^\circ$ axis. For example, lighting the first LED indicates a sound source at 90° because the LED is located on the right-hand side of Figure 5.6 and lies on the same axis as the two microphones. Lighting up the second LED represents a sound source at $90 - 1 \times \frac{180}{14} \approx 77^\circ$, and the eighth LED indicates $90 - 7 \times \frac{180}{14} = 0^\circ$.

The LED strip expects a 5 V control signal with a threshold greater than the output of the FPGA (HIGH$\geq$3.4 V). For that reason, the Input/Output Board has a level shift circuit shown in Figure 5.7 and marked with red in Figure 5.6. The unidirectional level-shift circuit consists of a single bipolar junction transistor and two resistors at the base and the collector. This circuit only works for high impedance loads at the OUT pin and drivers that can sink enough current at the INP pin.

Before discussing digital signal processing, we want to summarize the most important design parameters from the Analog and Input/Output Board.

1. The implementation has two microphones, and the distance between them is $d = 0.066\,\mathrm{m}$. The

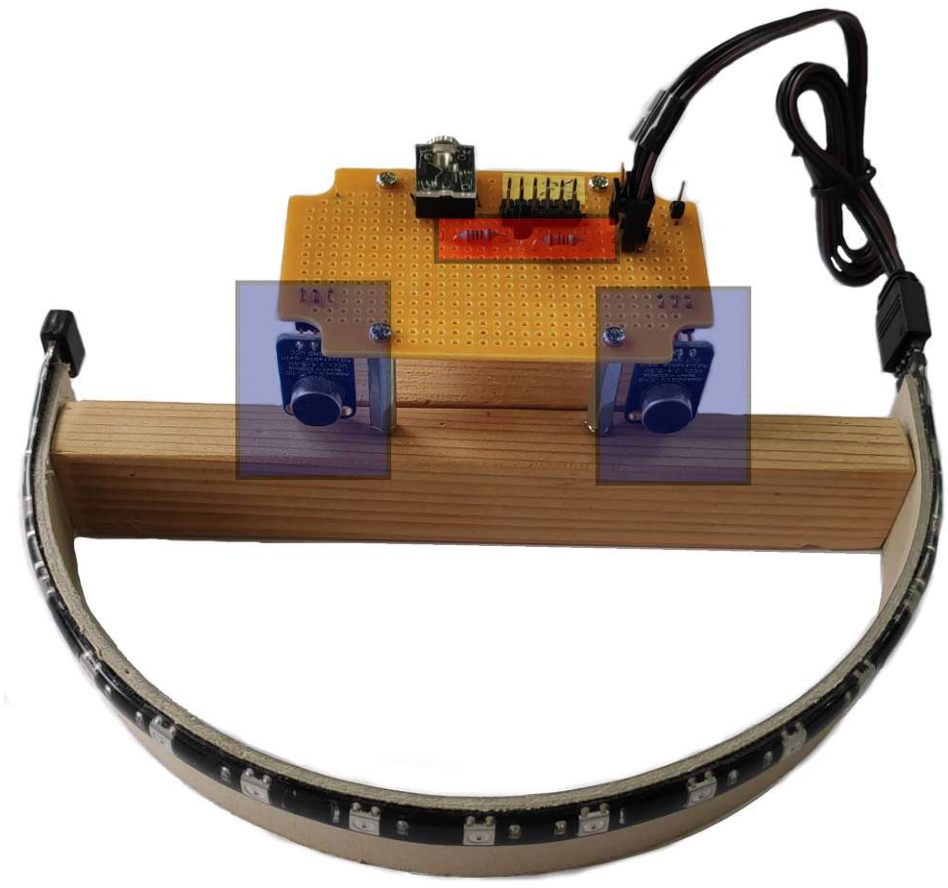

Figure 5.6: Input/Output Board with two microphones as input sensors and LEDs to display outputs. The microphones are soldered to ADAFRUIT MAX4466 and marked blue. The level shift circuit for interfacing with the LED strip is marked with red.

distance sets the maximum time delay of sound waves with $\tau_{d,max} = \frac{d}{v} = 192\,\mu\text{s}$.

2. The audio sampling frequency is $f_s = 15.6\,\text{kHz}$. The main reason for setting $f_s = 15.6\,\text{kHz}$ is that its product with the distance between the microphones is approximately an integer value $N_{MaxLag} = f_s \tau_{d,max} \approx 3$, simplifying the digital implementation and the mapping to the LEDs.
3. The LED strip of the Analog Board can indicate 15 different source locations from $\varphi \in [-90^\circ, 90^\circ]$. The spatial distribution of the LEDs is aligned with the coordinate system of the 2-microphone array. The axis through the microphones is perpendicular to $\varphi = 0^\circ$ and parallel to $\varphi = \pm 90^\circ$.

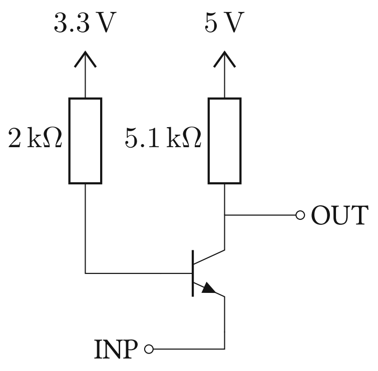

Figure 5.7: Unidirectional 3.3 V to 5 V level-up shift circuit marked red in Figure 5.6.

5.2 Digital Implementation

This section explains the digital processing blocks of the sound source localizer shown in Figure 4.1 on the right-hand side of the vertical line, labeled with FPGA. We will start with the CC function and continue with the remaining digital processing blocks. The focus lies on the SSL-based CC because the other processing blocks are implemented with traditional binary arithmetic.

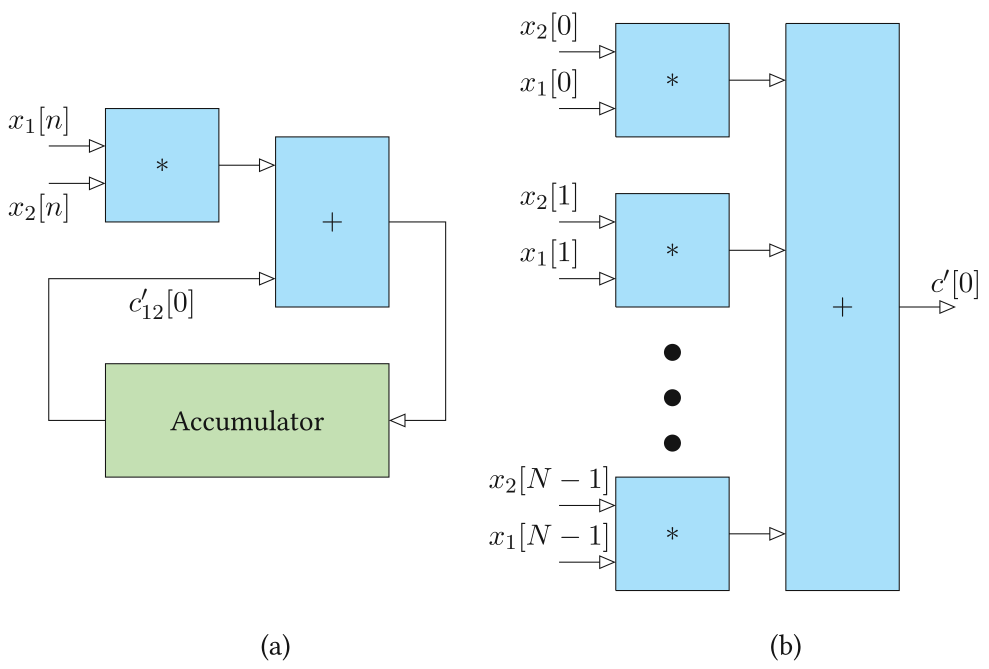

Figure 5.8: Two MAC architectures for calculating one value of the CC function. (a) Sequential MAC architecture. (b) Parallel MAC architecture.

We need to divide the continuous microphone signals into frames before computing the CC function. The window length is a trade-off between responsiveness and accuracy. Setting $N_{frame} = 128$ means that one computation of the CC function requires $\Delta t = \frac{1}{f_s} \times N_{frame} = 8.2\,\text{ms}$. Subsequently, we will assume that the position of a sound source is approximately constant within 8.2 ms, which is essential for accurate estimations. We can write the CC function for signals of length N_{frame} as

$$c'[n] = \sum_{i=0}^{N_{frame}-1} x_1[i]x_2[(i+n)_{\text{mod}\,N_{frame}}]. \tag{5.4}$$

Further, as indicated in Section 3.3.2, only the CC values close to $c'[0]$ are relevant for TDEs. The distance between the microphones of the Input/Output Board in Figure 5.6 is $d = 0.066\,\text{m}$, and the sampling rate is $f_s = 15.6\,\text{kHz}$. Therefore only $N_{MaxLag} = 3$ (see Equation (3.14)) values to the right and the left of $c'[0]$ are of interest for TDEs. We can efficiently calculate the seven ($K = 2\times N_{MaxLag} + 1 = 7$) centered values of the CC function as a set of MAC operations. We will denote $x[n]$ as the current audio sample and $x[n-1]$, $x[n-2]$, $x[n-3]$ as the three former samples. We calculate the K-centered values of the CC function with K MAC units as

$$c'[i] = \begin{cases} c'[-3] & \leftarrow c'[-3] + x_2[n]x_1[n-3] \\ c'[-2] & \leftarrow c'[-2] + x_2[n-1]x_1[n-3] \\ c'[-1] & \leftarrow c'[-1] + x_2[n-2]x_1[n-3] \\ c'[0] & \leftarrow c'[0] + x_2[j]x_1[j] \\ c'[1] & \leftarrow c'[1] + x_2[n-3]x_1[n-2] \\ c'[2] & \leftarrow c'[2] + x_2[n-3]x_1[n-1] \\ c'[3] & \leftarrow c'[3] + x_2[n-3]x_1[n] \\ 0 & \text{if i<-3 or i>3} \end{cases} \tag{5.5}$$

with $j \in \{n-3, n-2, n-1, n\}$[1]. The accumulator is set to zero after N_{frame} samples just before the next frame begins. The two architectures corresponding to Equations (5.4) and (5.5) for sequential and parallel MAC operation are shown as block diagrams in Figure 5.8. The examples in Figure 5.8 compute the centered value $c'[0]$ of the CC because there is no relative shift between microphone signals $\mathbf{x_1}$ and $\mathbf{x_2}$. The MAC architecture in Figure 5.8(a) is more suited for efficient stream-based audio processing than the parallel, block-based method in Figure 5.8(b). Seven MAC units calculate the seven centered values of the CC result while storing only six input samples and seven accumulator values.

[1]We can use any pair of audio samples $(x_1[j]x_2[j])$ to compute $c'[0]$.

5.2.1 Overcoming the Limitations of SC

In this section, we explain how to compute MAC with SC and overcome the limitations of SC in computing multiple arithmetic operations that follow one after another. The MAC operation has a multiplication stage followed by a chain of summation stages. The number of stages depends on the window length N_{fame} and the MAC architecture. If all summands are available at any time, with the architecture in Figure 5.8(b), the number of summation stages is $\log 2(N_{frame}) = 7$. For the stream-based architecture in Figure 5.8(a), only two summands are present at any time, and the MAC unit has $N_{frame} - 1$ summation stages after the multiplication stage.

The difficulty with multi-stage computations in SC is that the correlation of intermediate results becomes unknown, and correlation between inputs affects the arithmetic operation. The result will be no well-defined arithmetic operation for most stochastic circuits when the correlation is not defined properly ($SCC \notin \{-1, 0, 1\}$). Let us take the OR-gate with the unipolar format and undefined correlation as an example, the result is a mixture between a scaled sum ($SCC = -1$), the union ($SCC = 0$) and the maximum of both inputs ($SCC = +1$). Below, we discuss several design options considered during research for efficient MAC operation with SC.

1. We can estimate and accept the error caused by an unknown correlation [45]. Not all stochastic circuits, however, are equally resilient to correlation errors. Non-scaled summation with OR-gates requires *negatively correlated* inputs at all stages. Every overlap between 1s in bit-streams will cause an error that cannot be compensated elsewhere. For that reason, in the case of saturating addition with OR-gates, the error caused by unknown correlation is high.
2. We could use MUX units for scaled addition instead of saturating addition with OR-gates. MUX units calculate scaled addition and do not impose constraints on the correlation of data inputs as long as the select input is uncorrelated. MUX units need an additional random bit-stream for the select input, which is only a minor drawback. The issue of MUX units is the scaling by $\frac{1}{2}$ for each addition. The MAC unit computes $N_{frame} - 1 = 127$ sums, equal to a scaling factor of $1/2^{127}$ for the first addend, $1/2^{126}$ for the second addend, and so forth, when we use the architecture in Figure 5.8(a). Using Figure 5.8(b), the scaling factor is $2^{-\log 2(N_{frame})} = 2^{-7}$. The equivalent precision of stochastic circuits is usually between three and five bits. The MUX-based MAC units compute results too small for accurate representation.
3. After each arithmetic operation, we can convert the intermediate results to weighted binary numbers and generate new bit-streams with the correct correlation. This approach has two major drawbacks. First, it leads to narrow time restrictions as PEs and SNGs require processing time, and the computation needs to finish in time (after $\frac{1}{f_s}$). Second, this method increases resource

consumption due to more circuitry and higher clock speeds.

4. There exist several techniques to manipulate the correlation between bit-streams [30]. However, there is a trade-off between the complexity of additional circuitry and the effectiveness of correlation manipulation. Circuits for manipulating the correlation can become more complex than the actual intended arithmetic operation.
5. With deterministic SC, we calculate the distribution of 1s and 0s in intermediate results as a function of the input values and number of inputs. We published this technique, which focused on the MAC operation, in [44]. We will use it in the structure shown in Figure 5.9. We attached the published paper as Appendix C to this thesis.
6. The final approach avoids multi-stage stochastic circuits by either choosing different algorithms or computing sequential arithmetic operations in weighted binary. Mapping this method to MAC operations gives two options for the architectures in Figure 5.8. First, using the MAC architecture in Figure 5.8(b) and an accumulative parallel counter (APC) to accumulate the products in parallel [46]. Second, choose the sequential algorithm in Figure 5.8(a) and accumulate it with a single counter, as shown in Figure 5.10. The output is a weighted binary number in both cases, which is beneficial for this thesis's SSL architecture as all other processing blocks use conventional arithmetic.

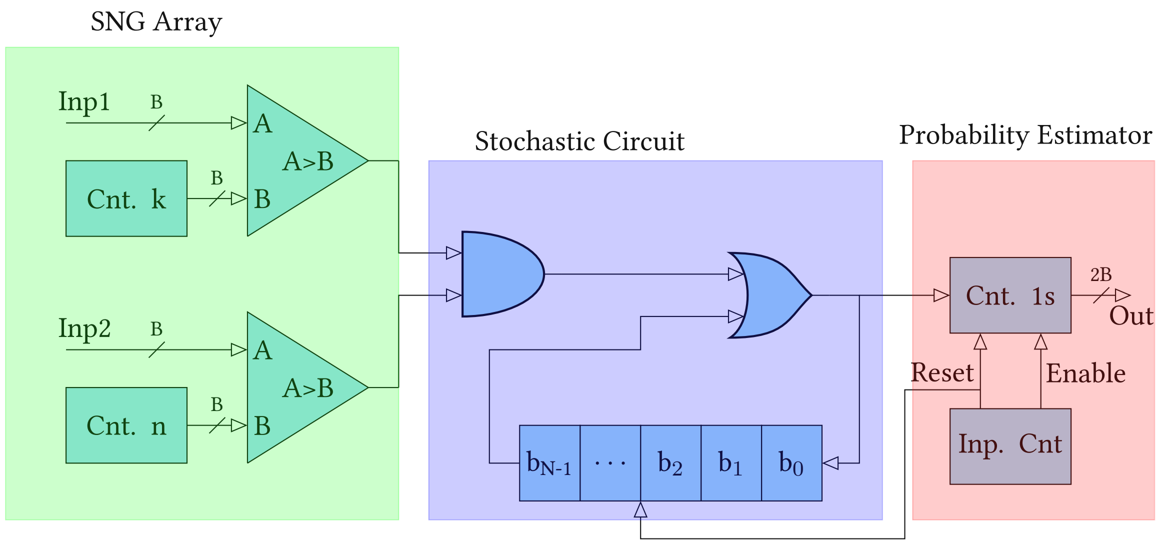

Figure 5.9: OR-based SC processing system for MAC operations.

In what follows, we present two SC implementations for MAC with the two last approaches from the list above. The examples in this section use digital SNGs to generate bit-streams, however, the analog PWM generation with the Analog Board can replace digital SNGs. The first MAC method uses the novel technique that we proposed and published during research [44]. As we showed in our paper too, we use AND-gates for multiplication and OR-gates for addition. AND-gates compute exact multiplication for relatively prime bit-streams with periods of n and k ($k = n - 1$)[2]. In this thesis, we use $n = 16$ and $k = 15$ unless stated otherwise, which is comparable to conventional computations with 4-bit precision ($B = \log_2(n) = \log_2(16) = 4$). The input bit steams, and the results of AND-gate have $N = LCM(nk) = 240$ bits, so the precision of the products is approximately $2B = 8$ [3]. As we use deterministic bit-streams, we can compute the distribution of 1s and 0s within products. It turns out that 1s are grouped at the beginning and the end of the result for relatively prime lengths $n = k - 1$. As mentioned in Section 2.3.2, OR-gates require negatively correlated bit-streams to function as saturating adders. Our novel approach introduces a relative delay between products, that shifts the 1s of the current summand to a section of the accumulator bit-stream with 0s only. Our new technique guarantees a negative correlation between input bit-streams and computes exact or approximate saturate addition. We refer to the published paper in Appendix C for more details.

Combining the stochastic MAC circuit with an SNG array and a probability estimator, we get the architecture shown in Figure 5.9. The two counters (marked with green and labeled with Cnt. n and Cnt. k) repeatedly count to n and $k = n - 1$ to generate relatively prime bit-streams. Then, the bit-streams are multiplied using an AND-gate. The AND-gate is the first component of the stochastic circuit and forwards the product $\mathbf{x_1 x_2}$ to the OR-gate. The OR-gate computes the bitwise disjunction of the most significant bit (MSB) of the shift register (accumulator) and the MSB of the product. The result is then stored back to the LSB of the shift register. The OR-gate result has an error if both inputs are 1s, which we refer to as an error due to overlap. In [44], we show how to calculate the number of bits for the shift register to guarantee no overlap for small input values, but audio signals can have high amplitudes. However, we know that at least half of the input values are zero due to the half-wave rectifier and the unipolar format. In this thesis, we use an approximate variant of the published technique with a shift register of length

$$N_{\text{SR}} = nk + \text{ceil}\left(\frac{nk}{N_{frame} - 1}\right) \cdot (N_{frame} - 1) = 494. \tag{5.6}$$

The first summand, in the beginning, will be in the $[0, 239]$ interval of the shift register. After the second summand, the first one is shifted to $[240, 479]$, and the second summand is in the $[0, 239]$ interval. After that, the OR-gate computes logical disjunction of the first and third summand, with a relative shift between them. The final block in Figure 5.9 is the probability estimator in the form of a simple up-counting counter. The probability estimator is enabled by a separate counter (Input Counter) that sets the enable signal N_{SR} clock cycles before the reset, which occurs every $nk \times N_{frame}$ cycles.

[2]Note that $n = k - 1$ is relatively prime for $n \geq 3$.

[3]A bit-streams with 256 bits would exactly have 8-bit resolution.

After each frame, the reset signal resets the probability estimator and the accumulator bit-stream. In the remainder of this thesis, we refer to the design in Figure 5.9 as the OR-based design.

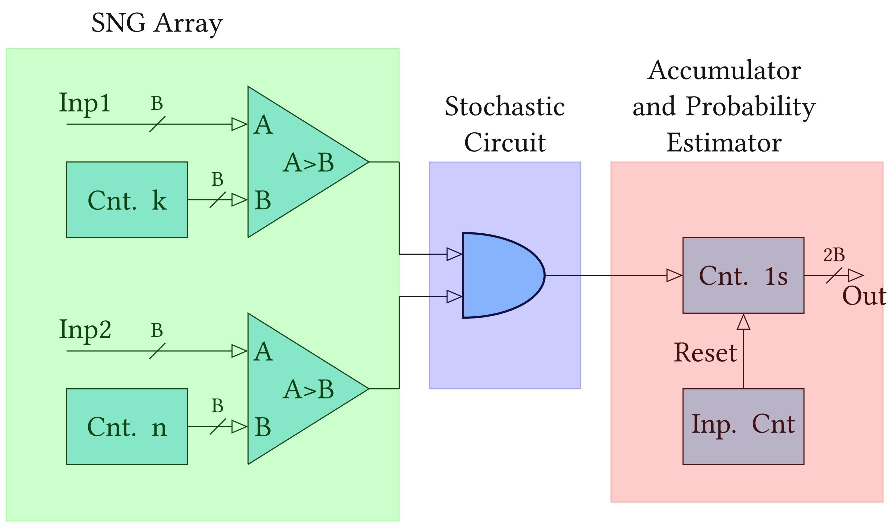

Figure 5.10: Counter-based SC processing system for MAC operations.

The second implementation is shown in Figure 5.10 and simplifies the stochastic circuit. The bit-stream generation and multiplication work as before but in this design, the PE counts all 1s of the AND-gate instead of first accumulating with OR. The results are always exact because the multiplication and the up-counting are exact. Subsequently, we will refer to this implementation as the counter-based design.

5.2.2 Stochastic Circuit for Cross-Correlation

In this section, we discuss how using either the PWM module or the ADC of the Analog Board affects the implementation in the digital domain. The MAC-based CC (Equation (5.5)) requires three (N_{MaxLag}) stored audio samples for each microphone. Designs that use the analog SN generation with the LTC6992 store the audio samples as bit-streams, and designs that use the ADC store the audio samples as weighted binary numbers. Usually, we prefer storing weighted binary numbers because bit-streams require more storage.

Figures 5.11 and 5.12 show the block diagrams of the CC with the counter-based MAC for digital and analog bit-stream generation. Channel One and Two (CH1 and CH2) are PWM signals from the LTC6992 in Figure 5.12 and weighted binary numbers in Figure 5.11. We draw the registers in a simplified manner as three horizontal squares when storing bit-streams and three vertical squares when storing weighted binary numbers. The output ($c[i]$) is the CC result in weighted binary format.

Next, we provide a simple and a more complex method to store the PWM signals from the LTC6992. For the simple variant, we continuously sample with f_{clk} and get $n = 16$ ($k = 15$) bits per period of

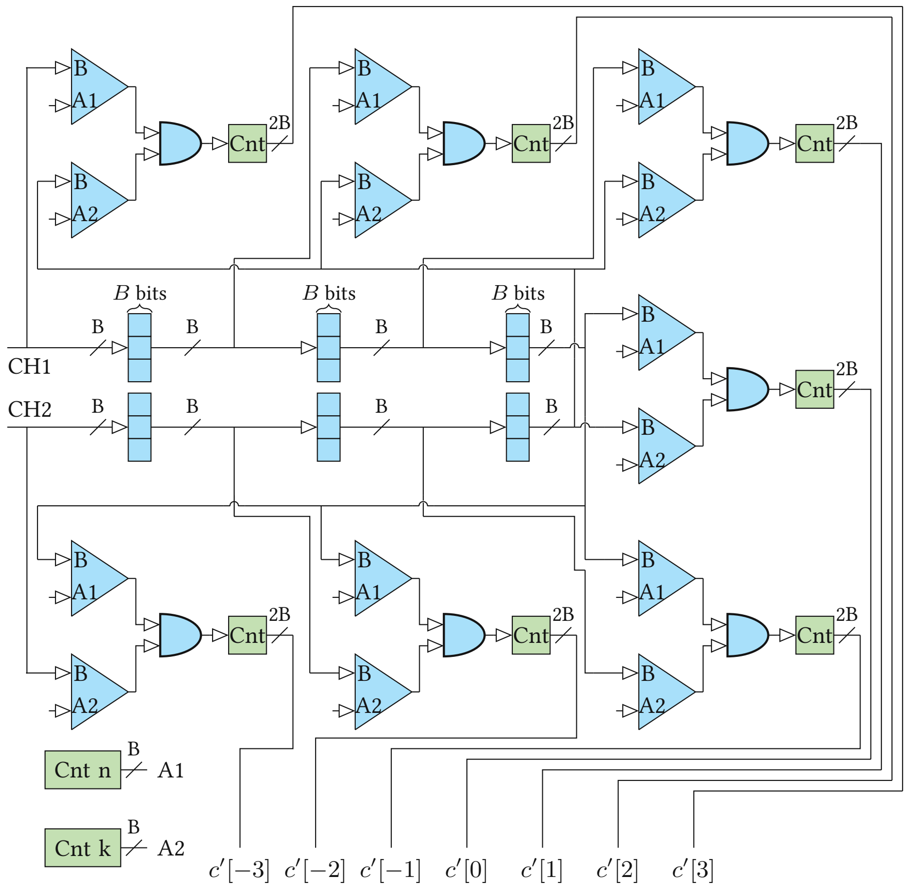

Figure 5.11: Block diagram of the CC with digital bit-stream generation.

the PWM signal. During one audio sample ($\frac{1}{f_s}$), the PWM signal has k (n) periods (see Equation (5.3)), which results in $nk = 240$ bits of data per microphone for each audio sample. The second variant takes advantage of the redundancy within the periodic PWM signals and stores only the first period. Following that, we perform roll operations until the first period of the next audio sample.

In Figure 5.12, we use arrows that point backward from the end to the start of the shift registers to indicate that we either do shift-through operations or roll operations. Both variants provide the same bit-streams to the stochastic circuit if we assume that the duty cycle of PWM signals is constant between two audio samples.

5.2.3 SPI, Maximum Search, Map and Average

The SPI block shown in Figure 4.1 is a minimal SPI-master that polls data from the ADC. The LTC1098 transmits one channel per query with 14 bits. The first six bits configure the ADC, and the remaining eight bits are either the left or right channel. To simplify the implementation, we increase the number of bits per query to $N_{query} = 16$. The ADC operates at its maximum clock frequency of 500 kHz ($f_{sclk} = f_s \times 2 \times N_{query} = 499.2\,\text{kHz}$).

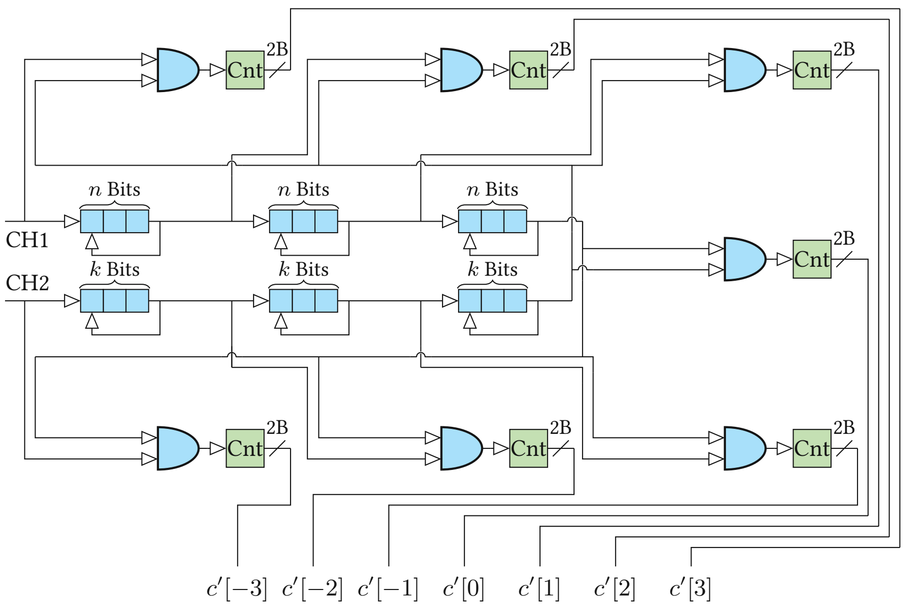

Figure 5.12: Block diagram of the CC with PWM signals as inputs (LTC6992).

The post-processing tasks are the maximum search, mapping and average, that we implement with conventional binary arithmetic. The maximum search finds the maxima of the MAC results $c[i_{max}]$ and forwards its index (i_{max}) to the Map & Average block if the maximum is above a threshold[4]. The Map & Average block maps $i_{max} \in \{-3, -2, \ldots, 2, 3\}$ to $i'max \in \{0, 2.5, 5.0, \ldots, 12.5, 15.0\}$ to create 15 equally sized intervals for the 15 LEDs of the Input/Output Board. Averaging over multiple i'_{max}, with an average filter, results in z_{avg}. We will present the filter design later in this section. z_{avg} is mapped to the LED strip of the Analog Board, as shown in Figure 5.13. The LEDs are marked red, and z_{avg} is labeled with black. The number of the LED that is turned on is the truncated filter output.

$$\text{LED}_{out} = \lfloor z_{avg} \rfloor \in \{0, 1, 2 \ldots, 13, 14\}. \tag{5.7}$$

As z_{avg} is proportional to the TDE, we can use its value to turn on the LEDs of the Input/Output Board. Mathematically, by mapping the filter output to a semicircle, we transform z_{avg} to an angle in the $[-90^\circ, 90^\circ]$ interval. When z_{avg} is close to zero, the first LED turns on. When z_{avg} is close to its maximum, the last one turns on. The two examples indicate sound source locations at $\varphi = \pm 90^\circ$ with the LED positions shown in Figure 5.13. We can generalize and calculate φ as a function of z_{avg} as follows.

[4]As the threshold, we use the median signal energy of all frames that is ≈ 0.33 for the simulation settings outlined in Section 6.1.

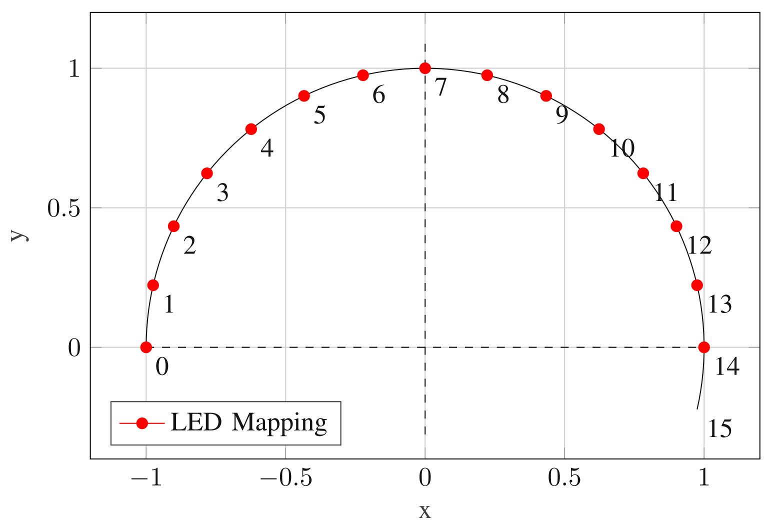

Figure 5.13: The digital output of the FPGA is in the $z_{avg} \in [0, 15]$ interval and is mapped to a LED strip with 15 LEDs mounted to the Input/Output Board (Figure 5.6).

$$\varphi = \sin^{-1}(\frac{v}{d}\tau_d) \cdot \frac{180^\circ}{\pi} \tag{5.8}$$

$$= \sin^{-1}(\frac{z_{avg}}{7.5} - 1) \cdot \frac{180^\circ}{\pi} \tag{5.9}$$

The average filter has an infinite impulse response (IIR) with the linear difference equation

$$z_{avg}[n] = \alpha \cdot i'max[n] \cdot (1 - \alpha) z_{avg}[n-1] \tag{5.10}$$

We choose $\alpha = 2^{-4}$ and approximate a moving average filter of length $N_{average} = 20$, with an average interval of 164 ms ($N_{average} \times 1/f_s \times N_{frame}$). Choosing the α coefficient is a trade-off between responsiveness and the accuracy of location estimations for stationary sound sources. Choosing an even smaller value for α (a larger averaging interval) hardly increases the accuracy of estimations for stationary sound sources in simulations. However, it drastically reduces the accuracy of estimations for moving sound sources.

5.2.4 Weighted Binary Implementation

We implement the CC function with weighted binary arithmetic and B=3,4,5,8-Bit precision signed inputs, for reference. The weighted binary CC design has seven MAC units, and the architecture is shown in Figure 5.8(a). The multiplier is exact with an output bitwidth twice as large as the input bitwidth. Note that the precision of products is comparable to the output bit-stream of the AND-gate with $N = nk$ bits. Choosing the accumulator bitwidth is a trade-off between not adding too many integer bits (saving resources) and avoiding overflow for audio frames with high volume.

5.3 Summary

In this chapter, we discussed the analog and digital components of the sound source localizer. The first and second outcomes are the Analog Board, for analog signal processing, and the Input/Output Board, for audio capture and visualization of location estimations. The third and fourth outcomes are digital implementation variants of the CC function for weighted binary and SC and the digital SSL post-processing based on an IIR average filter.

Chapter 6

Resource and Accuracy Comparison

The previous chapter discussed an SSL architecture that uses SC for the CC function. We can optionally use voltage-controlled PWM modules for SN generation and either OR-based or counter-based MAC units for the CC function. However, The previous chapter did not show any results for the SC variants nor the weighted binary reference. In this section, we compare the accuracy and resource consumption to test the hypotheses from Chapter 4. We describe and analyze the results but leave the interpretation and discussion for Chapter 7. This chapter consists of two main sections. In the first section, we compare the implementation variants concerning their estimation error. Section 6.2 continues with an analysis of their resource consumption. We distinguish between the following sound source localizer variants in the remaining of the thesis.

Design (I) The **conventional** design that uses weighted binary arithmetic, described in Section 5.2.4, with 4-bit precision signed inputs.

Design (II) The **counter**-based CC shown in Figure 5.11 with 4-bit unipolar (unsigned) inputs.

Design (III) The **counter**-based CC architecture shown in Figure 5.12 with **LTC6992** for analog SN generation and 4-bit precision unipolar inputs.

Design (IV) The architecture of Figure 5.12 with OR-based MAC units (Figure 5.9) and LTC6992 for analog SN generation of the 4-bit precision unipolar inputs.

Design (V) The architecture of Figure 5.11 with OR-based MAC units and digital 4-bit precision SNGs.

Design (VI) The stochastic circuits using **signed** instead of unsigned inputs. **B=3,4,5,8-bit** refers to the input precision of the CC function. For example, B=5-bit means one sign bit and four fraction bits. Due to having signed inputs, we use the bipolar format (instead of unipolar), the clock division method (instead of relative prime[1]) and XNOR-gates (instead of AND-gates) for multiplication

[1]Simulations show that the bipolar format with relatively prime method leads to inaccurate CC results caused by the unequal resolution of bit-streams ($k = n - 1$). We can avoid errors by switching to the clock division method discussed in Section 2.4, with no additional resource consumption [16].

Table 6.1: Summary of the relations between the digital results and estimated source locations.

i_{max}	i'_{max}	τ_d	φ
-3	0	$-192\,\mu s$	$-90°$
-2	2.5	$-128\,\mu s$	$-41.8°$
-1	5	$-64\,\mu s$	$-19.5°$
0	7.5	$0\,\mu s$	$0°$
1	10	$64\,\mu s$	$19.5°$
2	12.5	$128\,\mu s$	$41.8°$
3	15	$192\,\mu s$	$90°$

(see Equation (2.18)). The architecture of the CC is the same as in Figure 5.11, after replacing AND-gates with XNOR-gates. The CC results are exact and equivalent to weighted binary results from Design (I).

For all designs, the CC output has double the bitwidth of the input due to the multiplication stage of the MAC unit. The fraction bits of the accumulator stage are truncated to represent the result $2B$ bits without overflow. Unless otherwise stated, we round the CC inputs of all designs to the nearest representable value. We want to emphasize that the resource and accuracy differences are due to different CC implementations and SNGs. The maximum search, mapping and average filter are equal for all designs.

6.1 Accuracy

We use the Grid corpus audio library [47] which consists of 50 files with male and female speakers for accuracy simulations. All 50 files sum up to $90.08\,s$ of audio data with $76.03\,s$ audible speech (CC results are above the VAD threshold) and $14.05\,s$ breaks. The distance between the sound source and the center of the two microphones is $1\,m$, and the speaker radiates with $60\,dB$ SPL. The microphone SNR is $75\,dB$, and reverberations are disabled[2]. We sweep the sound source from $-90°$ to $90°$ in a $5°$ grid.

In [44], we propose and analyze the summation behavior of an OR-based MAC unit and conclude that smaller input ranges lead to lower errors. The audio inputs to Design (IV) and (V) are scaled-down by $\frac{1}{2}\times$ which quarters the maxima of the CC result[3] and reduces the error caused by saturation. We tabulate the simulation results for all designs in Appendix B. The tables contain the distribution of i_{max},

[2]Decreasing the SNR and increasing reverberation leads to higher SSL errors for all designs.

[3]If we assume that both microphone signals are equal, then the maximum of the CC function is the signal energy $E_s = \sum_{n=-\infty}^{+\infty} |x[n]|^2$. The energy quarters if the inputs are halved.

which can be used for comparisons with future CC implementations or to approximately recalculate the results presented below.

The accuracy results show the estimated source location ($\hat{\varphi}$) over the actual source location (φ). We show the results for $\hat{\varphi}$ rather than z_{avg} because $\hat{\varphi}$ is more intuitive and the LEDs of the Input/Output board also indicate $\hat{\varphi}$. We list the relation between i_{max}, i'_{max}, τ_d and $\hat{\varphi}$ in Table 6.1. The index i_{max} is proportional to the TDE with $\tau_d = i_{max}/f_s$ (Equation (3.11)). Both $i'_{max} = (i_{max} + 3) \times 2.5$ and $z_{avg} \approx \text{mean}(i'_{max})$ are related to $\hat{\varphi}$ through the inverse sinus.

6.1.1 Systematic Error

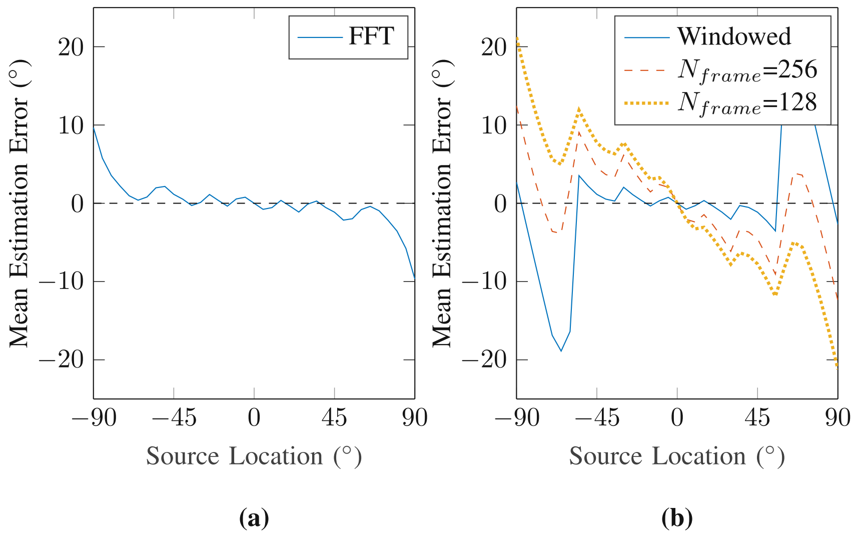

Figure 6.1: The Systematic error of the sound source localizer architecture. (a) Frame length of 256, Hanning window, FFT-based TDE, (b) variable frame length with and without Hanning window.

Before analyzing the effect of low precision computations and approximate computing in Section 6.1.2, we discuss the SSL error caused by short window lengths, not using a window function, and doing TDEs through CC in the time domain.

Figure 6.1 shows the mean estimation error of four sound source localizers with double-precision arithmetic and TDEs in the time and frequency domain. We can see a location-dependent bias, even with high computational effort[4] and no external disturbances, such as reverberation and noise. Figure 6.1(b) shows the mean estimation error for the sound source localizer with $N_{frame} = 128$ and $N_{frame} = 256$. The dataset labeled with *Windowed* comes from a sound source localizer that has an additional processing block preceding the CC, where frames are multiplied with the Hanning window (Equation (3.8)). Figure 6.1(b) shows that applying an appropriate window function after framing and increasing the window length reduces the systematic error. Figure 6.1(a) also uses a Hanning window

[4]Here, we also apply cubic interpolation after CC for high accuracy TDEs.

and frame length of $N_{frame} = 256$ but computes the time delay through multiplication in the frequency domain instead of CC in the time domain[5]. The sound source localizer with FFT-based TDEs has a lower systematic error for $|\varphi| > 45°$ than the one labeled with *Windowed*. We include Figure 6.1(a) to show the lowest possible error with two microphones and the averaging filter as post-processing.

The sound source localizers in both plots have a systematic bias towards positive mean errors for source locations at negative angles and a tendency towards negative mean errors for source locations with positive angles.

1. Mathematically, by cutting the microphone signals into frames, we multiply the *infinite* microphone signals with the rectangular function of value 1 during the current frame and 0s otherwise. In the frequency domain, the framing is equal to a convolution with the sinc function[6]. The sinc function shows a high peak at $t = 0$, so we are more likely to see the maximum of the CC result at $i_{max} = 0$. As $i_{max} = 0$ occurs more frequently, we see a bias towards $0°$ and a positive or negative mean error depending on the source location. As a countermeasure, we can either increase the window length, for example, doubling N_{frame} in Figure 6.1(b), to widen the peak of the sinc function or use a proper window function such as the Hanning function to avoid the sinc altogether.
2. For source locations close to $\varphi = \pm 90°$, the average IIR filter cannot compensate errors with an inverted sign. For example, for a sound source at $\varphi = -90°$, $84\,\%$ of all CC evaluations have the maximum at $c[-3]$ (Table B.1 for 8-bit precision). However, $16\,\%$ of all CC evaluations are either $c[-2]$ and $c[-1]$ and cause a positive bias.

Figure 6.1 shows ripples from spatial aliasing, discussed in Section 3.1.3. A low-pass filter eliminates the ripples if all frequencies that cause spatial aliasing are attenuated. In Figure B.1 (Appendix B) we show the mean estimation error for a sound source localizer with a first-order low pass filter at 2.5 kHz, which is low enough to avoid spatial aliasing.

6.1.2 Low Precision and Approximate Computing

In the previous section, we analyzed the systematic bias of the sound source localizer architecture with double precision arithmetic. The following sections are dedicated to efficient, low precision implementations. From here on, the sound source localizers always use fixed-point arithmetic, time domain CC, no Hanning window function and a frame length of $N_{frame} = 128$.

Figure 6.2(a) shows the mean SSL error for Design (VI) and the following three trends:

[5]A multiplication in the time domain is a convolution (mirror of the CC function) in the frequency domain and vice versa.
[6]The Fourier transform of the rectangular function is the sinc function.

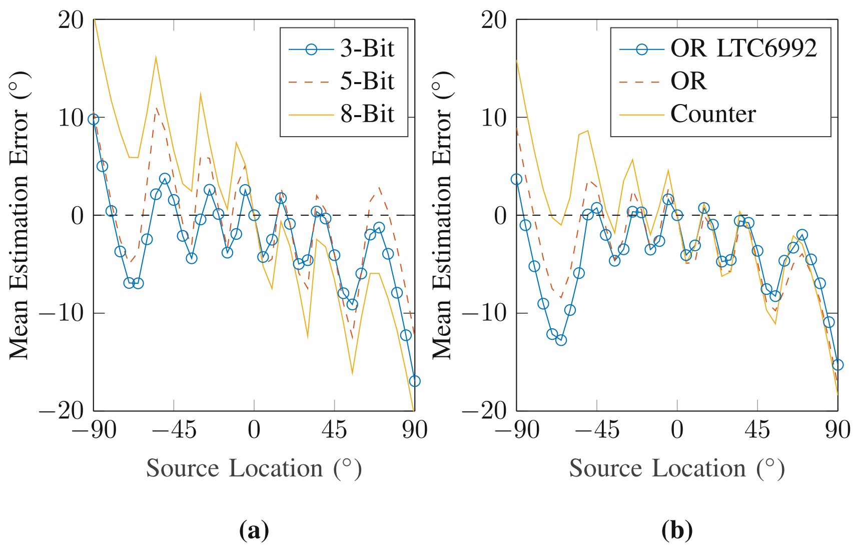

Figure 6.2: Mean estimation error of (a) Design (VI) (exact and bipolar) and (b) Design (II), (IV) and (V) (4-bit unipolar inputs).

1. The error due to not using a window function and a short window length is higher for designs with 8-bit precision than 3 or 5-bit precision. For example, the mean estimation error at $\varphi = 55°$ is $-8.8°$ for 2-bit and $-14.9°$ for 8-bit precision.
2. For 5-bit and 8-bit precision, the results are point symmetrical with $\varphi = 0°$. For example, the mean estimation error for $\varphi = -30°$ is the negation of the mean estimation error for $\varphi = +30°$, for 5 and 8-bit, but not for 3-bit precision. The probability of having multiple equal maxima in the CC result increases with low precision. The Max Search block forwards the index of the first maxima (with the lowest index) to the Map & Average block, which causes a bias towards $\hat{\varphi} = -90°$.
3. The mean estimation error plotted over the source location follows a triangle function. For source locations at $\hat{\varphi} \in \{-41.8°, -19.5°, 0°, 19.5° \text{ and } 41.8°\}$, the mean error is approximately zero because these angles correspond to $i_{max} \in \{-2, -1, 0, 1, 2\}$ (see Table 6.1). Between these angles, the averaging filter keeps the error below 20°, but for $i_{max} = \pm 3$ ($\pm 90°$), the error is higher because of the systematic error discussed in Section 6.1.1.

Figure 6.2(b) shows the mean estimation error for the unipolar Design (II), (IV) and (V). The error difference between the unipolar and bipolar format is low, but the difference between the OR-based and the counter-based summation is high due to more bias towards $-90°$. Using OR-gates for summation limits the output range of the CC and further increases the probability of having multiple equal maxima in the CC result.

For the designs in Figure 6.2, we also list the mean absolute error (MAE) in Table 6.2. The right-most column in Table 6.2 shows the mean of MAEs, which is below $8.6°$ for all designs. The MAE decreases with lower input precision[7] because low precision designs are less susceptible to the systematic error discussed in Section 6.1.1.

Table 6.2: MAE (°) of the angle estimation ($\hat{\varphi}$) in degree.

		φ (°)													
		-90	-75	-60	-45	-30	-15	0	15	30	45	60	75	90	**Mean**
Design (VI)	3-Bit Inputs	8.0	6.7	5.7	2.0	2.7	3.8	0.0	2.2	4.6	4.1	6.4	5.9	15.5	**4.9**
	4-Bit Inputs	6.2	8.5	5.1	3.2	2.9	4.0	0.0	3.0	5.3	3.8	6.4	6.4	11.0	**5.2**
	5-Bit Inputs	7.8	7.6	5.5	3.8	6.0	3.5	0.0	3.0	7.4	3.9	6.6	6.8	10.0	**5.6**
	8-Bit Inputs	18.5	9.0	10.6	6.5	12.3	2.7	0.0	2.7	12.3	6.5	10.6	9.0	18.6	**8.6**
Counter (II)		13.8	6.0	4.7	4.7	3.6	2.3	0.0	1.9	5.8	5.3	6.2	7.5	16.5	**5.8**
OR (III)		6.2	7.9	7.0	2.9	3.0	3.4	0.1	2.0	5.7	4.8	7.6	6.7	15.6	**5.3**
OR LTC6992 (IV)		2.0	11.3	10.3	2.3	3.5	3.5	0.0	1.5	4.5	3.6	4.6	5.0	13.5	**5.1**

During simulation, it turned out that the rounding method during the quantization of inputs affects the estimation error, and truncation rather than rounding gives lower MAEs. For example, the average MAE (last column of Table 6.2) changes to $4.7°$, $4.9°$, $5.4°$ and $8.2°$ for 3, 4, 5, and 8-bit precision, respectively, an improvement of $3\,\%$ to $9\,\%$. The results are similar for the unipolar designs. Truncation leads to sharper peaks in the CC result and better TDEs. An in-depth analysis of the rounding method is out of the scope of this thesis.

6.2 Resource Consumption

Lowering resource consumption is one of the main goals when considering SC in designing a computing system. This section shows the synthesis results produced by the Synopsys Design Compiler v2018.06 with the $45\,\mathrm{nm}$ FreePDK CMOS library [48] for the ASIC design flow and Vivado Design Suite for synthesis on an FPGA. The HDL synthesis tools analyze the digital processing blocks on the right hand-side of the vertical line in Figure 4.1. We further show the power consumption measurements for the processing blocks implemented with discrete components.

Table 6.3: Power consumption of the Analog Board.

Total Idle		Active LTC6992		Active LTC1098		Total Active
$17.65\,\mathrm{mW}$	+	$2 \times 0.95\,\mathrm{mW}$	+	$0.5\,\mathrm{mW}$	=	$20.05\,\mathrm{mW}$

The Analog Board in Figure 5.1 has four black current sense resistors for power consumption measurements. The low current consumption of the Analog Board in conjunction with the low resistance of the current sense resistors ($R = 0.025\,\Omega$) causes a small voltage drop over the current sense resistor

[7]Lowering the input precision of the CC to 2-bit increases the MAE.

that is difficult to measure. For example, the voltage drop over the current sense resistor of the LTC1098 is approximately $0.1\,\mathrm{mA} \times 0.025\,\Omega = 2.5\,\mu\mathrm{V}$.

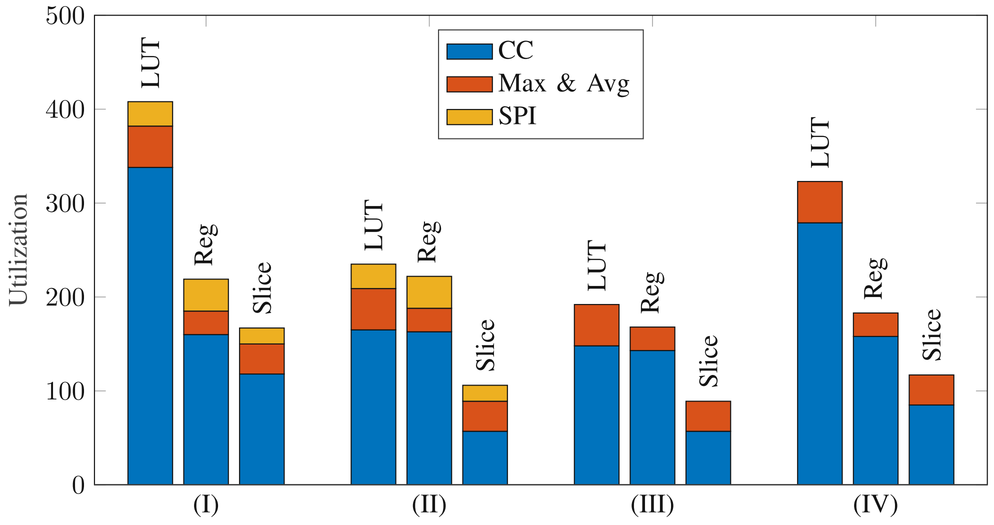

Figure 6.3: LUT, registers and slice utilization of the digital processing blocks. (I) Weighted binary, (II) counter, (III) Counter with LTC6992, (IV) OR with LTC6992.

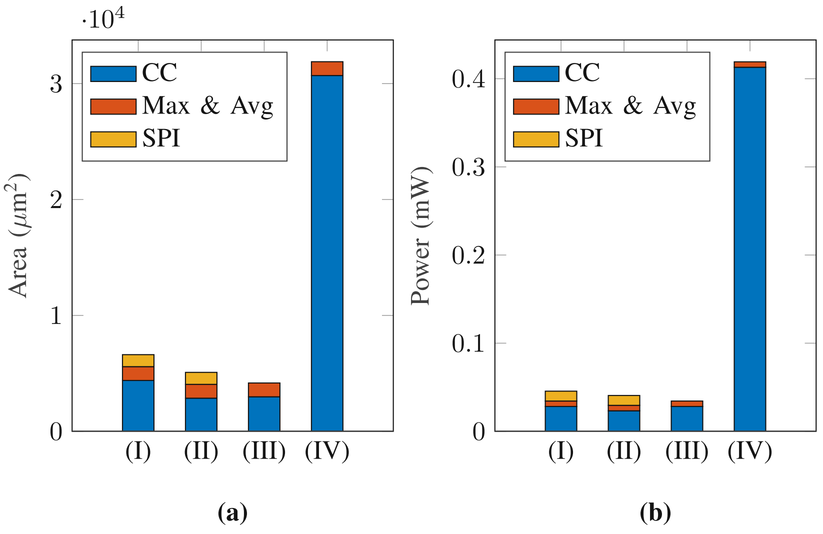

Figure 6.4: Area (a) and power (b) consumption of SC and weighted binary designs for 4-bit precision and $f_s = 15.6\,\mathrm{kHz}$. (I) Weighted binary, (II) counter, (III) Counter with LTC6992, (IV) OR with LTC6992.

Figures 6.3 and 6.4 show the resource consumption for the Max Search and Map & Average (Subsequently referred to as Max & Average), SPI and CC processing blocks for FPGAs and ASICs. The resource consumption of the 4-bit bipolar design (Design (VI)) is equal to the Counter design (Design (II)). The figures visualize the area reports of Vivado and Synopsys. However, we only show the power consumption estimations for ASICs. Power estimations from Vivado are not precise enough as the static power consumption of the FPGA is significantly higher than the power consumption of the individual processing blocks. The power consumption estimation of Synopsys reports that designs with SC components have higher dynamic power consumption but lower static power consumption[8]. Vivado and Synopsys estimate that the CC function consumes more power than the Max & Average and SPI processing blocks. Both HDL synthesis tools agree that the Counter with LTC6992 design (Design (III)) consumes the least resources, followed by the Counter (Design (II)) and weighted binary design (Design (I)). The synthesis tools estimate a different resource consumption of the OR with LTC6992 design (Design (IV)). Synopsys reports about ten times higher power and area consumption, whereas the utilization report of Vivado shows *only* a doubling in LUT utilization compared to other CC designs. The OR with LTC6992 design (Design (IV)) stores a bit-stream accumulator with $nk = 240$-bit (for 4-bit precision arithmetic). FPGAs are optimized for storing data, whereas ASICs synthesize a separate flip-flop for each bit. When comparing Design (I) with Design (II), we save $11\,\%$ power and $23\,\%$ area if we use SC for cross-correlation. The savings of the digital implementation increase to $25\,\%$ power and $37\,\%$ area when we compare the conventional design with Design (III). However, Design (III) uses the PWM modules on the Analog Board, which consume more power than the ADC and outweigh the savings in the digital domain.

Table 6.4 lists the resource consumption for the counter-based CC with digital SN generation (used in Design (II)) and the conventional CC (used in Design (I)). The table shows the utilization, power and area consumption for 3-, 4-, and 5-bit precision inputs. The area consumption increases with the input precision, but the differences are higher for the conventional implementation. For higher precision, the weighted binary design has larger multipliers and adders. In contrast, the area consumption of the counter-based CC implementation hardly increases because only the counter size for SN generation and the accumulation changes. Going from 3-bit to 5-bit inputs increases the area consumption by $53\,\%$ for the conventional and $18\,\%$ for the SC design. The utilization increase on FPGAs is in the same range.

The clock speed to finish one MAC operation is constant for the weighted binary and increases exponentially in the SC-based implementations. The conventional CC always computes with $15.6\,\mathrm{kHz}$. The clock speed of the SC-based CC increases from $873.6\,\mathrm{kHz}$ for 3-bit to $3.744\,\mathrm{MHz}$ for 4-bit and $15.4752\,\mathrm{MHz}$ for 5-bit ($f_{clk} = nk \times f_s$). Table 6.4 shows a power consumption increase of $53\,\%$ for the conventional design and $254\,\%$ for the SC design. SC consumes less power for 3-bit, more power for 4-bit and 5-bit inputs, and fewer area resources in all cases.

A close look at the resource consumption of the CC for Design (I) and (II) in Figure 6.4 indicates that the conventional CC consumes slightly more power than the counter-based CC. However, Table 6.4

[8]Vivado also reports higher dynamic power consumption for designs with SC, if we increase clock speeds and fill up the FPGA with duplicates of the HDL modules.

shows it the other way around for 4-bit inputs. We use a faster clock in full system simulations because all designs need one high-frequency clock to control the SPI and generate the SCLK. When simulating Design (I) and (II), we use the same 3.744 MHz clock and a clock-enable signal to enable the weighted binary CC every $f_s = 15.6\,\text{kHz}$[9]. In Table 6.4, we use a 15.6 kHz clock and connect the clock-enable signal to a constant high (Logic-1). Using a higher base clock and a clock-enable signal increases the power consumption of the conventional CC from 23.4 µW to 28.2 µW.

Table 6.4: FPGA utilization and ASIC area and power consumption of the CC function for $f_s = 15.6\,\text{kHz}$.

Vivado	3-Bit Inputs			4-Bit Inputs			5-Bit Inputs		
	LUT	Reg	Slice	LUT	REG	Slice	LUT	Reg	Slice
Conventional	139	145	80	257	160	122	303	182	130
Counter-based	72	151	61	83	163	69	88	174	72

Synopsys	3-Bit Inputs		4-Bit Inputs		5-Bit Inputs	
	Area	Power	Area	Power	Area	Power
Conventional	3956 μm^2	19.6 µW	4788 μm^2	23.4 µW	6066 μm^2	29.9 µW
Counter-based	3136 μm^2	18.4 µW	3393 μm^2	27.9 µW	3699 μm^2	65.1 µW

[9]We use switching annotations to increase the accuracy of the power estimations.

Chapter 7

Discussion and Interpretation

In the previous three chapters, we first outlined the hypotheses and contributions of this thesis. Then we presented the implementation details of the sound source localizer, focusing on SC and weighted binary variants of the CC function. Chapter 6 showed the simulation and measurement results that we will interpret in Chapter 7. Structurally, this chapter consists of four sections. Sections 7.1 and 7.2 discuss the results and cover the first four hypotheses of Section 4.2. Section 7.3 discusses the complexity and implications of considering SC in a processing system, covering the fifth hypothesis. The last section points out the limitations of this research.

7.1 Resource Consumption

LTC6992 is not as efficient as LTC1098. However, other research [17] shows that generating PWM signals can consume less power than conventional analog-to-digital conversion. We still use LTC6992 in our analog design because the PWM module is adjustable and can output frequencies in the 4 Hz to 1 MHz interval. The power consumption difference of LTC6992 and LTC1098 is too large to be compensated through savings in the digital domain. Since achieving a lower total power with SC is an important goal of this thesis, we also provide the resource consumption of SC designs that use conventional ADCs (Design (II) and (V)).

The resource analysis confirms that the CC function consumes more resources than the average filter, SPI-module and maximum search. Resource savings due to applying SC in the CC function have a relevant impact on the overall resource consumption. The analysis also showed that storing bits consumes more resources than combinational logic, with three main consequences. First, the sequential MAC architecture from Figure 5.8(a) is more efficient than the architecture in Figure 5.8(b) because it stores fewer audio samples. Second, it is crucial to capture only one period of LTC6992 and digitally recreate the PWM signal (see Section 5.2.2). Third, the OR-based approach requires the most resources due to the cost of the accumulator, which is a large shift register. We need to use the counter-based CC with the comparisons in Table 6.4 and figs. 6.3 and 6.4 to be more area and occasionally power-efficient than the conventional implementation. The SC design requires less static power but more dynamic power. The threshold, whether SC is more power-efficient, shifts with technology and is currently at 4-bit precision inputs for the 45 nm CMOS library. Newer technologies have lower dynamic and higher static power consumption and favor the SC-based design.

7.2 Accuracy

It turns out that the precision of inputs has little effect on the accuracy of SSL because the algorithm only uses the position (index) of the CC's maxima. LTC1098 is an 8-bit ADC, and we can either round or truncate to lower precisions, but simulations show that truncation leads to sharper peaks in the CC result and lower MAEs.

This work does not yield any advantages while studying the approximate saturating OR-based adder for the stream-based MAC unit and SSL. In contrast to the accurate counter-based MAC, the OR-based design has several disadvantages. First, the accumulator needs to be redesigned for different window lengths and computational precisions. Second, the OR-based MAC is sensitive to volume changes because its accuracy rapidly decreases when the OR-based adder is saturating. Third, the OR-gate adder is limited to the unipolar format.

The outputs of the SC approach are the same as the outputs from the binary approach when using the bipolar format with the clock-division method and the Counter design.

7.3 Design Complexity

The VHDL code of the stochastic circuits is synthesizable with the same tools used to synthesize the weighted binary circuits.So, the SC approach does not add any additional synthesis effort. Digital chips have a limited number of physical pins. LTC6992s uses two (PWM) instead of four (SPI) connections to the ASIC or FPGA, which can be advantageous when physical pins are limited. However, alternative protocols for ADCs such as I^2C also require two pins.

The SC and weighted binary CC designs change with computational precision. The stochastic circuit stays the same[1], and only the clock speed needs to be adjusted when lowering or increasing the precision. For weighted binary arithmetic, the size of adders and multipliers change, but the clock speed stays constant. SC-based designs could use different clock speeds based on the accuracy demand, affecting the dynamic power consumption. Dynamically adapting the precision is similar to saving

[1]Only the counters for bit-stream generation and probability estimation use more or fewer bits.

power by exploiting progressive precision [10], which was excluded from the analysis in this thesis (Section 4.2).

The counter-based CC shown in Figure 5.11 contains SNGs, the stochastic circuit and PEs, which form a complete SC processing system (Figure 2.1). It is self-contained and can replace a conventional CC design without further changes to other processing systems as it has weighted binary inputs and outputs. The counter-based CC works for unsigned and signed inputs by replacing AND-gates with XNOR-gates. The SC-based and the weighted-binary CC designs are equally scalable as additional MAC units compute more values of the CC function[2].

7.4 Limitations

We reported the power consumption estimations based on the synthesis results from the Synopsys Design Compiler with a 45nm CMOS gate library. The results will be different for other tools and technology. Any changes to the base clock or computational speed benefit either the conventional or the SC designs. Simulating the designs for different clock speeds (with clock-enable signals), different sampling frequencies, and synthesis tools would give a complete view of the power consumption.

Our analysis consider SSL as the only processing task in the digital design. However, the system usually includes other signal processing tasks. The parallel MAC architecture could be more efficient than the stream-based MAC (Figure 5.8) if the audio samples are stored in any case and do not add costs to SSL. The counter-based MAC is not an option for the MAC architecture in Figure 5.8(b). It can be replaced with APCs. Even OR-based summation could be a competitive alternative because it does not need a bit-stream accumulator. We need $N_{frame} = 128$ multipliers and adders in the parallel architecture (instead of only one), changing the resource comparison of SC and weighted binary designs.

The resource consumption savings with the SC approach are maximized if the cost of other subsystems is similarly low. The block diagram in Figure 4.1 could additionally contain a processing block to apply a window function and an adaptive whitening filter which both increase the accuracy of estimations but lower the overall gain through using SC. Further, SC designs cannot benefit from analog SN generation if the first digital processing block uses conventional digital binary arithmetic.

[2]The example in the block diagram computes the seven centered values of the CC.

Chapter 8

Conclusion

In this thesis, we first introduced SC and SSL. Chapter 2 discussed random and deterministic SC and the linear unipolar and bipolar SC formats. The background chapter for SSL focused on efficient TDE-based algorithms. We compute the TDE with the CC function, either block-wise in time or frequency domain, or stream-based with MAC units. Efficient implementation of multiplication operation is a well-known strength of SC. So the research was narrowed down to efficient accumulation in MAC operations. To prove the functionality of the SC-based sound source localizer, we implemented the full signal processing chain. This resulted in two prototype circuit boards and multiple HDL designs.

Implementing a separate analog processing chain for SC did not provide power savings because generating voltage-controlled PWM signals with LTC6992 consumes about two times more power than the conventional analog-to-digital conversion with LTC1098. However, the analog implementation served as the proof of concept that SC does not rely on conventional ADCs. The digital HDL design showed the weaknesses and strengths of SC. On the one hand, the approximate OR-based design has lower flexibility and more design constraints than the counter-based and conventional designs because the OR-gate adder only works for unsigned inputs and is sensitive to data value changes. On the other hand, we also designed a new counter-based CC with SC that is completely exact and easy to use.

Conventional adders and multipliers become more complex for higher computational precision, whereas stochastic circuits hardly change. When synthesized on FPGAs, the SC designs approximately halve the slice and LUT utilization compared to the conventional design. Our synthesis results showed that the ASIC area consumption decreases by 21 % for 3-bit inputs and by 39 % for 5-bit inputs. The power consumption increases with computational precision for weighted binary and SC but exponentially faster for SC. For 3-bit input precision, the SC-design consumes 18.4 µW with an 874 kHz clock, and the conventional design consumes 19.6 µW with a clock equal to 15.6 kHz sampling frequency. When we increase the computational precision to 4-bit or 5-bit, we can keep the clock speed of the conventional design but must increase the clock frequency of the stochastic circuit to finish the computations in the same time. For 4-bit, for example, the power consumption increases to 27.9 µW for the SC-based design but only to 23.4 µW for the conventional design. To keep the power consumption in favor of the SC-based design, we can decrease the sampling rate to loosen the time constraints, or use technologies with lower dynamic power.

The Counter design is not tied to SSL or any specific signal processing task. The SC-based CC design can be reused in other applications, which use low-precision CC functions, and can provide the same resource savings as for SSL.

Bibliography

[1] S. Liu, W. J. Gross, and J. Han, “Introduction to dynamic stochastic computing,” *IEEE circuits and systems magazine (New York, N.Y. 2001)*, vol. 20, no. 3, pp. 19–33, 2020.

[2] S. T. Ribeiro, “Random-pulse machines,” *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 3, pp. 261–276, 1967.

[3] W. J. Poppelbaum, C. Afuso, and J. W. Esch, “Stochastic computing elements and systems,” in *American Federation of Information Processing Societies: Proceedings of the AFIPS ’67 (Fall)*, 1967, p. 635–644.

[4] B. R. Gaines, “Stochastic computing,” in *American Federation of Information Processing Societies: Proceedings of the AFIPS ’67 Spring Joint Computer Conference, April 18-20, 1967, Atlantic City, New Jersey, USA*, 1967, pp. 149–156.

[5] B. Gaines, “Stochastic computer thrives on noise,” *Electronics 40 (14)*, pp. 72–79, 07 1967.

[6] W. Gross and V. Gaudet, *Stochastic Computing: Techniques and Applications.* Springer Nature Switzerland AG, 02 2019.

[7] A. Alaghi and J. P. Hayes, “Survey of stochastic computing,” *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, May 2013.

[8] G. E. Moore, “Cramming more components onto integrated circuits,” *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[9] R. K. Budhwani, R. Ragavan, and O. Sentieys, “Taking advantage of correlation in stochastic computing,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[10] A. Alaghi, W. Qian, and J. P. Hayes, “The promise and challenge of stochastic computing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1515–1531, 2018.

[11] H. Wang, Z. Zhang, X. You, and C. Zhang, “Low-complexity winograd convolution architecture based on stochastic computing,” in *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, 2018, pp. 1–5.

[12] R. Xu, B. Yuan, X. You, and C. Zhang, “Efficient fast convolution architecture based on stochastic computing,” in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2017, pp. 1–6.

[13] B. Brown and H. Card, “Stochastic neural computation. i. computational elements,” *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 891–905, 2001.

[14] P. Li, D. J. Lilja, W. Qian, M. D. Riedel, and K. Bazargan, “Logical computation on stochastic bit streams with linear finite-state machines,” *IEEE Transactions on Computers*, vol. 63, no. 6, pp. 1474–1486, 2014.

[15] I. Tashev, *Sound Capture and Processing: Practical Approaches.* John Wiley & Sons, 07 2009.

[16] D. Jenson and M. Riedel, “A deterministic approach to stochastic computation,” in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[17] M. H. Najafi, S. Jamali-Zavareh, D. J. Lilja, M. D. Riedel, K. Bazargan, and R. Harjani, “Time-encoded values for highly efficient stochastic circuits,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1644–1657, 2017.

[18] M. H. Najafi, D. J. Lilja, and M. Riedel, “Deterministic Methods for Stochastic Computing using Low-Discrepancy Sequences,” in *Proceedings of the 37th International Conference on Computer-Aided Design*, ser. ICCAD ’18, 2018.

[19] B. Yuan and Y. Wang, “High-accuracy fir filter design using stochastic computing,” in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, pp. 128–133.

[20] M. H. Najafi, S. Jamali-Zavareh, D. J. Lilja, M. D. Riedel, K. Bazargan, and R. Harjani, “An overview of time-based computing with stochastic constructs,” *IEEE Micro*, vol. 37, no. 6, pp. 62–71, 2017.

[21] S. Liu and J. Han, “Energy efficient stochastic computing with sobol sequences,” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 650–653.

[22] M. Alawad and M. Lin, “Survey of stochastic-based computation paradigms,” *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 98–114, 2019.

[23] J. H. Anderson, Y. Hara-Azumi, and S. Yamashita, "Effect of lfsr seeding, scrambling and feedback polynomial on stochastic computing accuracy," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 1550–1555.

[24] A. Alaghi and J. P. Hayes, "Fast and accurate computation using stochastic circuits," *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–4, 2014.

[25] S. Buchovecká, R. Lórencz, F. Kodýtek, and J. Bucek, "True random number generator based on ropuf circuit," in *2016 Euromicro Conference on Digital System Design (DSD)*, 2016, pp. 519–523.

[26] H. Hata and S. Ichikawa, "Fpga implementation of metastability-based true random number generator," *IEICE Transactions*, vol. 95-D, pp. 426–436, 02 2012.

[27] P. C. Knag, W. D. Lu, and Z. Zhang, "A native stochastic computing architecture enabled by memristors," *IEEE Transactions on Nanotechnology*, vol. 13, pp. 283–293, 2014.

[28] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Transactions on Computers*, vol. 60, no. 1, pp. 93–105, 2011.

[29] S.-S. Choi, S.-H. Cha, and C. C. Tappert, "A survey of binary similarity and distance measures," *Journal of systemics, cybernetics and informatics*, vol. 8, no. 1, pp. 43–48, 2010.

[30] V. T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 1417–1422.

[31] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, 2013, pp. 39–46.

[32] M. H. Najafi, D. Jenson, D. J. Lilja, and M. D. Riedel, "Performing stochastic computation deterministically," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2925–2938, 2019.

[33] K. K. Parhi, "Analysis of stochastic logic circuits in unipolar, bipolar and hybrid formats," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[34] R. K. Budhwani, R. Ragavan, and O. Sentieys, "Taking advantage of correlation in stochastic computing," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[35] J. H. DiBiase, "A high-accuracy, low-latency technique for talker localization in reverberant environments using microphone arrays," Ph.D. dissertation, B.S., Trinity College, 1991 Sc.M., Brown University, 1993, 2000.

[36] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 4, pp. 320–327, Aug 1976.

[37] M. Brandstein, J. Adcock, and H. Silverman, "A closed-form location estimator for use with room environment microphone arrays," *IEEE Transactions on Speech and Audio Processing*, vol. 5, no. 1, pp. 45–50, 1997.

[38] P. Svaizer, M. Matassoni, and M. Omologo, "Acoustic source location in a three-dimensional space using crosspower spectrum phase," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 1997, pp. 231–234 vol.1.

[39] B. Van Den Broeck, A. Bertrand, P. Karsmakers, B. Vanrumste, H. Van hamme, and M. Moonen, "Time-domain generalized cross correlation phase transform sound source localization for small microphone arrays," in *2012 5th European DSP Education and Research Conference (EDERC)*, Sep. 2012, pp. 76–80.

[40] S. Birchfield and D. Gillmor, "Acoustic source direction by hemisphere sampling," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 5, 2001, pp. 3053–3056 vol.5.

[41] J. Smith and J. Abel, "Closed-form least-squares source location estimation from range-difference measurements," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 12, pp. 1661–1669, Dec 1987.

[42] J. Delosme, M. Morf, and B. Friedlander, "Source location from time differences of arrival: Identifiability and estimation," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80.*, vol. 5, Apr 1980, pp. 818–824.

[43] B. Yuan, Y. Wang, and Z. Wang, "Area-efficient scaling-free dft/fft design using stochastic computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 12, pp. 1131–1135, 2016.

[44] P. Schober, M. H. Najafi, and N. Taherinejad, "High-accuracy multiply-accumulate (mac) technique for unary stochastic computing," *IEEE Transactions on Computers*, pp. 1–14, 2021.

[45] B. Moons and M. Verhelst, "Energy and accuracy in multi-stage stochastic computing," in *2014 IEEE 12th International New Circuits and Systems Conference (NEWCAS)*, 2014, pp. 197–200.

[46] P.-S. Ting and J. P. Hayes, "Stochastic logic realization of matrix operations," in *2014 17th Euromicro Conference on Digital System Design*, 2014, pp. 356–364.

[47] M. Cooke, J. Barker, S. Cunningham, and X. Shao, “An audio-visual corpus for speech perception and automatic speech recognition,” *The Journal of the Acoustical Society of America*, vol. 120, no. 5, pp. 2421–2424, 2006.

[48] “NCSU FreePDK 45nm Library,” https://eda.ncsu.edu/freepdk/freepdk45/.

Appendix A

Mathematical Symbol Reference

Table A.1: A description of the variables we use in this thesis.

a, b	weighted binary scalar value	B	precision, bitwidth
d	distance between two microphones	D	duty cycle
f	frequency	E	expected value
i, m	integer value / index	G	gain
k, n	relative prime k=n-1	K	number of different possible time-delays in the CC during TDE
r	radius	M	number of microphones
t	time	N	length of the stochastic bit-stream
v	the velocity of sound waves	P	probability
x, y	stochastic number	S	bit-stream /stochastic sequence
z	result/output	X, Y	random variable

- $\mathbf{x}$ …data vectors are written bold
- $\vec{p}$ …vector in 3D coordinate system
- ρ_1 …probability of observing 1
- $\hat{\rho}_1$ …a hat indicates measured or estimated values
- $x_i(t)$ …continuous signal
- $x[n]$ …discrete signal
- $\mathbf{x}^*$ …complex conjugate $\mathbf{x}$
- $c[n]$ …CC function
- τ_d …time delay estimation
- i_{max} …index of the maximum of the CC function

Adafruit MAX4466

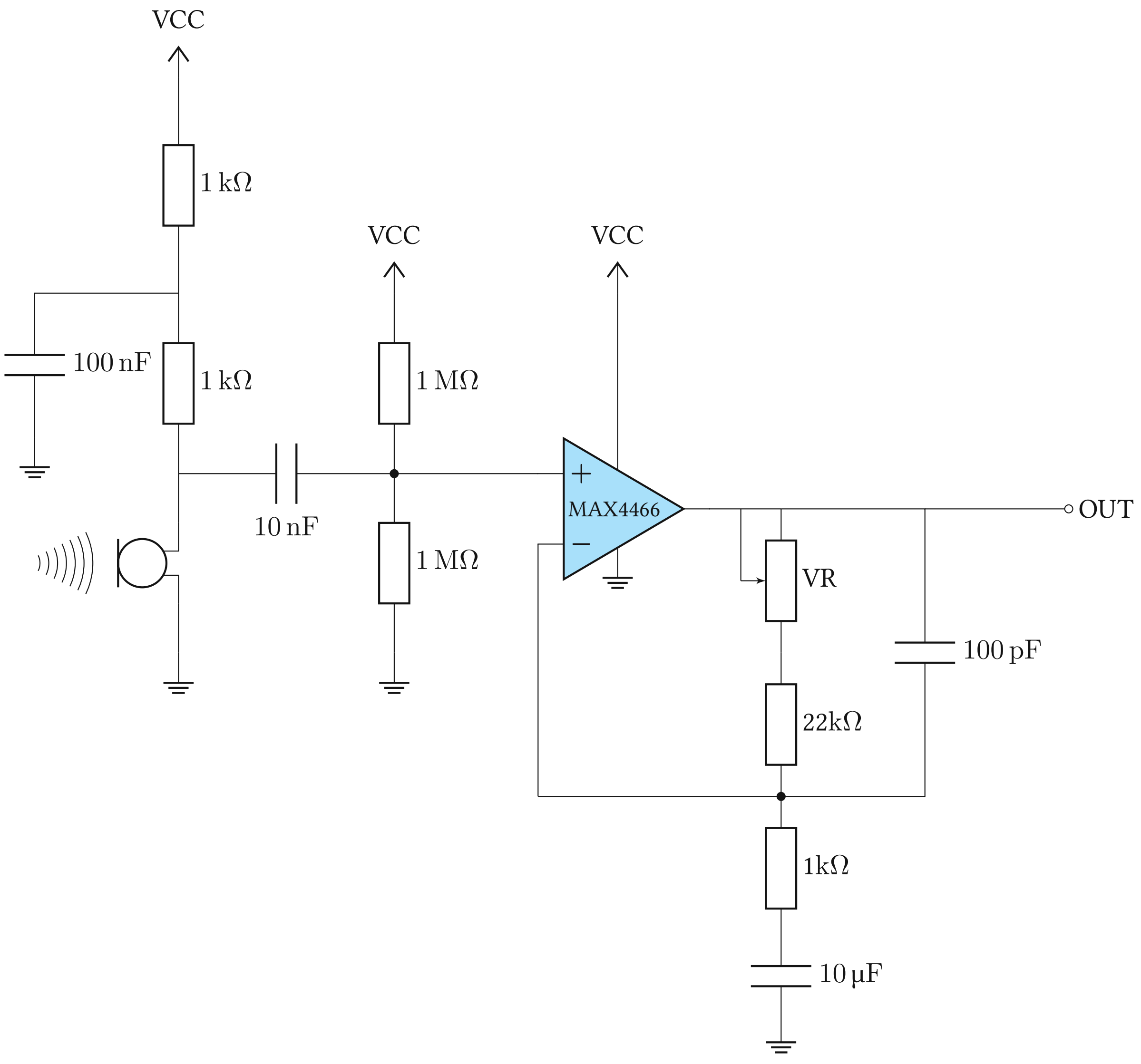

Figure A.1: Schematic of the Adafruit electret microphone amplifier (MAX4466 board). The board requires an input voltage of 2.4 V to 5.5 V. The gain can be changed with a variable resistor in the [25, 125] interval.

Appendix B

Simulation Data

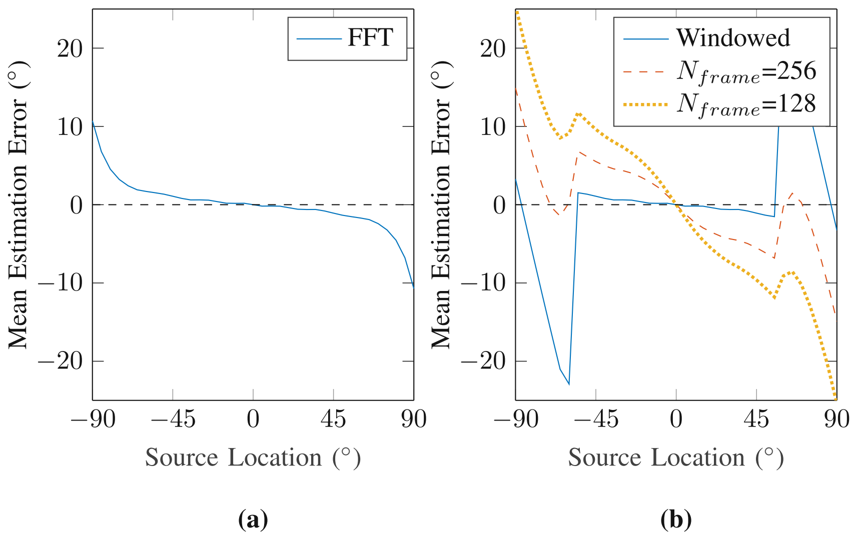

Figure B.1: Systematic error as in Figure 6.1, however, with a 2.5 kHz low-pass filter to suppress spatial aliasing.

Table B.1: Distribution of the CC-maximum ($c[i_{max}]$) in percent (%) for different computation precision and source location in the [-90,-90] interval. For example, 84% of all CC-results have their maximum at $i_{max} = 3$ for 8-bit and double precision inputs.

	i_{max} \ φ	90	80	70	60	50	40	30	20	10	0	10	20	30	40	50	60	70	80	90
3-Bit	-3	96	96	93	70	17	3	0	0	1	0	0	0	0	0	0	0	0	0	0
	-2	3	4	6	28	82	96	54	2	0	0	1	0	0	0	0	0	0	0	0
	-1	0	0	0	0	0	1	44	98	59	0	0	0	1	0	0	0	0	0	0
	0	0	0	0	1	1	0	0	0	39	100	61	2	1	0	1	2	1	0	0
	1	0	0	0	0	0	0	1	0	0	0	38	97	66	7	2	1	0	0	0
	2	0	0	0	0	0	0	0	0	1	0	0	0	32	92	93	49	17	12	10
	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	4	49	82	88	89
4-Bit	-3	97	97	94	65	6	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	-2	2	3	5	33	92	98	43	0	0	0	1	0	0	0	0	0	0	0	0
	-1	0	0	0	0	0	2	55	99	51	0	0	0	2	0	0	0	0	0	0
	0	0	0	0	2	1	0	0	1	47	100	63	1	1	1	1	2	0	0	0
	1	0	0	0	0	0	0	2	0	0	0	34	99	68	4	1	0	0	0	0
	2	0	0	0	0	0	0	0	0	1	0	0	0	29	96	96	47	12	7	6
	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	50	87	92	93
5-Bit	-3	96	95	90	56	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	-2	3	4	9	41	97	96	26	0	0	0	1	0	0	0	0	0	0	0	0
	-1	1	1	1	1	1	3	71	99	37	0	0	0	2	0	0	0	0	0	0
	0	1	1	1	2	1	1	1	1	61	100	69	2	1	1	2	2	1	1	1
	1	0	0	0	0	0	0	2	0	0	0	29	98	77	4	1	1	1	1	1
	2	0	0	0	0	0	0	0	0	1	0	0	0	19	95	97	47	11	6	5
	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	49	88	93	94
8-Bit	-3	84	83	75	40	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	-2	13	14	22	53	91	83	1	0	0	0	1	0	0	0	0	0	0	0	0
	-1	2	2	2	4	6	14	91	87	15	0	0	0	2	0	0	0	0	0	0
	0	1	1	1	3	3	3	6	13	83	100	83	13	6	3	3	3	1	1	1
	1	0	0	0	0	0	0	2	0	0	0	15	87	91	14	6	4	2	2	2
	2	0	0	0	0	0	0	0	0	1	0	0	0	1	83	91	53	22	14	13
	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	40	74	83	84
Double	-3	84	83	74	40	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	-2	13	14	22	53	91	83	0	0	0	0	1	0	0	0	0	0	0	0	0
	-1	2	2	2	4	6	14	92	87	15	0	0	0	2	0	0	0	0	0	0
	0	1	1	1	3	3	3	6	13	83	100	83	13	6	3	3	3	1	1	1
	1	0	0	0	0	0	0	2	0	0	0	15	87	92	14	6	4	2	2	2
	2	0	0	0	0	0	0	0	0	1	0	0	0	0	83	91	53	22	14	13
	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	40	74	83	84

Table B.2: Distribution of the CC-maximum ($c[i_{max}]$) in percent (%) for 4-bit (bipolar), 4-bit unipolar, two approximate OR implementations and source location in the [-90,0] interval.

	φ / i_{max}	90	80	70	60	50	40	30	20	10	0	10	20	30	40	50	60	70	80	90
4-Bit	-3	97	97	94	65	6	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	-2	2	3	5	33	92	98	43	0	0	0	1	0	0	0	0	0	0	0	0
	-1	0	0	0	0	0	2	55	99	51	0	0	0	2	0	0	0	0	0	0
	0	0	0	0	2	1	0	0	1	47	100	63	1	1	1	1	2	0	0	0
	1	0	0	0	0	0	0	2	0	0	0	34	99	68	4	1	0	0	0	0
	2	0	0	0	0	0	0	0	0	1	0	0	0	29	96	96	47	12	7	6
	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	50	87	92	93
4-Bit Unipolar	-3	91	90	85	61	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-2	6	7	12	34	92	92	37	0	0	0	1	0	0	0	0	0	0	0	0
	-1	2	2	2	2	2	6	58	95	47	0	0	0	1	0	0	0	0	0	0
	0	1	1	1	3	2	3	3	5	52	100	63	6	5	4	3	3	2	2	2
	1	0	0	0	0	0	0	1	0	0	0	36	94	66	7	3	2	2	3	2
	2	0	0	0	0	0	0	0	0	1	0	0	0	28	90	92	40	14	8	7
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	54	82	88	89
4-Bit OR	-3	97	96	94	76	20	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	-2	2	3	5	23	78	96	60	1	1	0	0	0	0	0	0	0	0	0	0
	-1	1	1	1	1	1	2	38	98	61	1	1	0	0	0	0	0	0	0	0
	0	0	0	0	1	1	1	1	1	38	99	74	7	5	3	2	2	1	1	1
	1	0	0	0	0	0	0	0	0	0	0	25	93	72	10	4	3	3	2	2
	2	0	0	0	0	0	0	0	0	0	0	0	0	24	87	91	57	22	13	10
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	38	74	84	86
4-Bit OR, LTC6992	-3	99	99	99	85	30	3	0	0	0	0	0	0	0	0	0	0	0	0	0
	-2	1	1	0	15	69	96	66	1	0	0	0	0	0	0	0	0	0	0	0
	-1	0	0	0	0	1	1	33	99	68	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	32	100	66	3	2	1	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	34	97	71	10	2	1	1	1	1
	2	0	0	0	0	0	0	0	0	0	0	0	0	27	89	95	49	20	12	10
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	49	79	86	89

Appendix C

Published Research

High-Accuracy Multiply-Accumulate (MAC) Technique for Unary Stochastic Computing

Peter Schober, M. Hassan Najafi, *Member, IEEE* and Nima TaheriNejad, *Member, IEEE*

***Abstract*—Multiply-accumulate (MAC) operations are common in data processing and machine learning but costly in terms of hardware usage. Stochastic Computing (SC) is a promising approach for low-cost hardware design of complex arithmetic operations such as multiplication. Computing with deterministic unary bit-streams (defined as bit-streams with all 1s grouped at the beginning or end of a bit-stream) has been recently suggested to improve the accuracy of SC. Conventionally, SC designs use multiplexer (MUX) units or OR gates to accumulate data in the stochastic domain. MUX-based addition suffers from scaling of data and OR-based addition from inaccuracy. This work proposes a novel technique for MAC operation on unary bit-streams that allows exact, non-scaled addition of multiplication results. By introducing a relative delay between the products, we control correlation between bit-streams and eliminate OR-based addition error. We evaluate the accuracy of the proposed technique compared to the state-of-the-art MAC designs. After quantization, the proposed technique demonstrates at least 37% and up to 100% decrease of the mean absolute error for uniformly distributed random input values, compared to traditional OR-based MAC designs. Further, we demonstrate that the proposed technique is practical and evaluate area, power and energy of three possible implementations.**

***Index Terms*—Stochastic Computing, Unary Computing, Multiply Accumulate, Unary Bit-streams, deterministic bit-stream processing, pulse-width modulation.**

I. Introduction

STOCHASTIC computing (SC) [1]–[3] is an unconventional computing paradigm providing low-cost and noise-tolerant design for complex arithmetic functions such as multiplication. In contrast to common positional binary representation, in SC, data is represented using non-positional uniform bit-streams. The bit-streams can be random with interleaved bits of 0s and 1s or predictable (deterministic) with uniform *unary* bit-streams having first all 1s and then all 0s (or vice versa) [4]–[6].

Stochastic Computing (SC) can be realized in both digital and analog domain. In the digital domain, the binary to bit-stream conversion is often performed using a stochastic number generator (SNG) unit built from a random number generator (RNG) (or a counter for the unary case) and a comparator [5]. Alternatively, in the analog domain where the input is given in analog voltage or current format, an analog-to-time converter such as a pulse-width modulator can be used to convert the data into a time-encoded stochastic number [7]. The important factor in generating stochastic

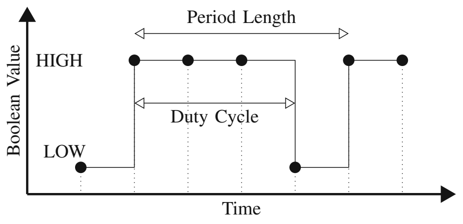

Fig. 1: Encoding the value 0.75 into the duty-cycle of a time-encoded pulse-width-modulated (PWM) signal. SC works with time-continuous PWM signals as well as with their time-discrete represented which we call periodic unary bit-streams.

numbers is the ratio of the number of 1s to the length of bit-stream, or the fraction of the time that the signal is high (i.e., logic-1). For example, if a signal is high 20% of the time, or equivalently, if 20% of the bits in a bit-stream are 1, the signal/bit-stream represents 0.20 in the so-called *unipolar* representation [3]. In the unipolar format, the probability of observing a 1 in the bit-stream is equal to the represented value[1]. Unless otherwise stated, the bit-streams discussed in this paper are in the unipolar format. The outputs of stochastic operations are again two-level signals, which can be used as the input(s) to other stochastic circuits or converted back to positional binary representation for further processing using conventional binary designs or storing in memory.

While (pseudo) random bit-streams have been the common form of representing data in SC [2], [3], unary bit-streams recently attracted attention due to their efficient and low-cost generation, and their potential for deterministic and accurate computation using SC logic [6]–[11]. For example, 1100, 0011, and 1111000 are all examples of unary bit-streams representing 0.5. Unary bit-streams in the digital domain are interpreted as PWM signals in the analog domain [7]. A PWM signal is defined by a duty cycle (D) and a frequency (or period where frequency=1/period). The duty cycle is the fraction of time in which the signal is high. Hence, the duty cycle determines the represented value. Fig. 1 shows a PWM signal with $D = \frac{3}{4}$, which can also be sampled as a discrete unary bit-stream and represented by 0111. Continuous PWM signals can work with significantly higher speed [7], but are more susceptible to environmental conditions and noise compared to discrete bit-streams. While processing of discrete bit-streams is also limited by quantization noise, digital bit-streams are easier to buffer and process compared to continuous signals.

Accumulation (addition) is an essential operation for many computing systems. In unipolar SC, numbers are limited

Peter Schober and Nima TaheriNejad are with the Institute of Computer Technology (ICT), Technische Universität Wien (TU Wien), Vienna, Austria. M. Hassan Najafi is with the School of Computing and Informatics, University of Louisiana, LA, 70504, USA. Email: peter-schober@gmx.at, najafi@louisiana.edu, nima.taherinejad@tuwien.ac.at

[1]A stochastic value is said to be unipolar, if $x = M\rho_x$, where x denotes the represented value, ρ denotes the probability of observing high (logic 1), and M is a positive scaling factor. In this paper, we assume $M = 1$.

to the [0,1] interval [2]. Hence, scaled addition, instead of normal addition, is natural as the maximum output from normal addition will be above the upper bound. A multiplexer (MUX) implements scaled addition in SC, when *correlated* (or *uncorrelated*) bit-streams are connected to the main and an *uncorrelated* bit-stream, representing 0.5, is connected to the select input [5], [7]. In this paper, we use the stochastic cross correlation (SCC) [12] as a measure for correlation. Two bit-streams are called positively correlated ($SCC = +1$) when they have maximum overlap between 1s, and negatively correlated ($SCC = -1$) when they have minimum overlap between 1s. Further, the term uncorrelated ($SCC = 0$) is used interchangeable with *independent*. The correlation of bit-streams can be controlled during generation of them or manipulated [13] when receiving from other stochastic circuits. It is also possible to generate approximately uncorrelated or correlated bit-streams from existing bit-streams using additional circuitry. For example, the uncorrelated select input of the MUX can be generated by using an additional XOR gate and a (toggle) flipflop [14], [15].

The processing time increases exponentially with the bit-stream lengths as the output length equals the product of the periods of the input bit-streams [5]. When the input values are small and the result stays in the representable range, i.e., in the [0,1] interval, non-scaled addition is preferred. As an alternative to the MUX-based scaled addition, OR gate has been suggested for fast and non-scaled addition of data [16]. A requirement for OR-based addition, however, is that the input bit-streams must be negatively correlated to produce accurate output. If a bit in a bit-stream is 1, the same bit position in the other bit-stream(s) must be 0. Any overlap between 1s results in inaccuracy in the OR-based addition.

Besides the traditional summation methods using MUX unit and OR gate, several modifications and alternatives have been proposed in the literature. One approach for non-scaled addition combines an OR-based adder with additional circuitry that includes a shift register [17]. When both inputs of the OR gate are 1, the circuit forwards logic-1 to the output and stores a 1 bit in the shift register. When both inputs are 0, a previously stored 1 bit is added to the inputs. Assuming that the shift register is large enough to store enough 1s and the sum is in the valid interval, this method also allows exact summation. If the sum should be converted back to a binary number, an accumulative parallel counter (APC) can be used to implement the exact non-scaled summation and the conversion to binary in a single circuit. APCs function as multi-input stochastic-to-binary converters that increase an internal counter for each 1 at their inputs [18]. Finally, scaled summation can be mapped to mealy finite state machines [15]. The number of inputs is equal to the number of states and the current state in conjunction with the current input determine the output. Despite the fact that in SC, multiplication can be accurate and efficient, it has been shown in [19] that multiply-accumulate (MAC) circuits can be implemented without conventional stochastic multiplication and addition. In [19], the authors use counters to compute multiplication as well as accumulation and the inputs and outputs are in conventional binary format.

In this work, we propose a novel summation theory that builds on the already existing work on exact multiplication operation (with AND gates) using deterministic unary bit-streams [5], [7], [8]. Combined, this work performs fast, efficient and accurate MAC operation that can be used in many applications. We propose an OR-based MAC unit that accurately accumulates the output of multiplication operations performed on unary bit-streams.

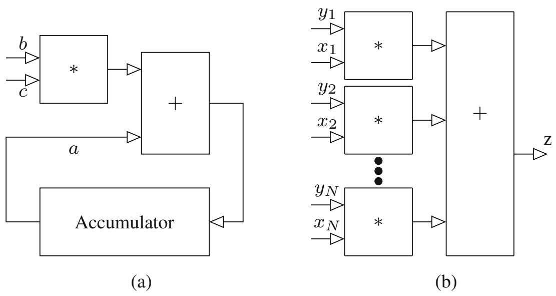

Fig. 2: (a) MAC architecture with accumulator a and factors b and c for sequential data. (b) Parallel MAC architecture with factors x_i, y_i and result z.

The rest of this paper is organized as follows. Section II provides a brief overview of unary SC and MAC operation. In Section III, we present our claims for the proposed theorem as well as the theoretical limits on summation of multiplication results for our technique. In Sections IV and V, mathematical proofs are derived for the proposed theory and its upper error bound, respectively. Section VI presents experimental results of the proposed technique. In this section, we also provide accuracy comparisons with the state-of-the-art MAC designs and evaluate gray-scaling as a practical case study. In Section VII, we provide an analysis of the resource consumption between different MAC implementations. We further discuss constraints and latency of our technique in Section VIII. Finally, we draw conclusions in Section IX.

II. Basics of MAC Operation using Time-Encoded Stochastic Computing

The MAC operation is defined as

$$a \leftarrow a + (b \cdot c). \tag{1}$$

with a as accumulator, and b and c as factors. A block diagram of Equation (1) is illustrated in Fig. 2(a). It is worth pointing out that using multiple multipliers in parallel combined with a multi-input adder produces the same output value z. That is,

$$z = \sum_{i=1}^{N} x_i y_i \tag{2}$$

with inputs x_i, y_i. The corresponding architecture is shown in Fig. 2(b). The advantage of (2) lies in a faster computation speed at the cost of additional hardware resources.

A unary bit-stream is mathematically describable through a length (or period), n, and number of 1s, v. The value represented by a unary bit-stream is $\frac{v}{n}$. For example, if $n = 16$ and $v = 4$, the bit-stream represents $\frac{v}{n} = \frac{4}{16} = 0.25$. We will show that factors with relatively prime lengths of $k = n - 1$ always have a centered interval of uninterrupted 0s in the products. A relative delay (i.e., a unique lag between bit-streams) can position 1s of other summands in an interval where all other products exclusively have 0s. This allows accurate accumulation of stochastic products through logic-OR. The proposed technique has no restrictions concerning the cause of the relative delays between summands. As examples, we provide three possible implementations in Section VI-B.

Throughout the paper, whenever we refer to multiplication, the underlying operation is logic-AND, and addition stands for applying logic-OR to stochastic bit-streams. When we use the terms summands, products or the inputs of the OR gate, we refer to the intermediate results between multiplication and summation. We emphasize that the focus of this paper lies in the summation of unary bit-streams. For more detail on why and how multiplication of unary bit-streams is performed accurately, we refer the reader to [5]. In Section III, we will take a closer look at the limits on summation of products using logic-OR and upper-error-bound, if these limits are exceeded.

III. Theorem

A. Claim

Let v denote the maximum allowed number of 1s in the input bit-streams. If binary numbers, represented by unary bit-streams with relatively prime lengths of n, k (where $k = n-1$ and both inputs are less than or equal to $\frac{v}{k}$) are multiplied using logic-conjunction, then N products (results of multiplication) can be accurately summed using logic-disjunctions, when each summand is processed with a predefined lag. The upper-bound of N and v is given by

$$N \leq \left\lfloor \frac{n-v}{v} \right\rfloor \cdot \left\lfloor \frac{n}{v} \right\rfloor \tag{3}$$

$$v \leq \left\lfloor \frac{n}{\left\lceil \frac{\sqrt{4N+1}-1}{2} \right\rceil + 1} \right\rfloor, \tag{4}$$

where $\{N, n, v \in \mathbb{N}\}$.

B. Expanded Explanation and Examples

Note that N is the number of inputs to the OR gate (not to the MAC unit). A pairwise multiplication of $2N$ inputs with AND gates results in N products, which are then summed by OR gates. For example, if $n = k+1 = 8$ and $v = 4$, at most $N = 2$ products can be summed by an OR gate without producing any error. The four inputs of the MAC unit must be less than or equal to $\frac{v}{k} = 0.5714$. As some additional examples, Table I lists the maximum allowed input thresholds for different $n = k+1$ and N for exact MAC.

Note that, Equation (4) can be used when the application determines the number of inputs. In that case, the input range is $[0, \frac{v}{k}]$. We also provide (3), which is more applicable when the input ranges are known in advance or the application does not imply a specific number of inputs.

The delays mentioned above need to be applied in the computation-chain before performing logic-OR. This is done by either inserting registers, delaying the bit-stream generation, or using the intrinsic delay of sequential arriving data. The important point in performing exact addition is that all summands are delayed differently and therefore, N unique delays are required to sum N products. In Section IV, we will derive two types of delays. Long or *major*, and intermediate or *minor* delays. We will show $N_{major}=\left\lfloor \frac{n-2v}{v} \right\rfloor$ ($N_{major} \in \mathbb{N}$) as the number of different long and $N_{minor}=N_{major}+1=\left\lfloor \frac{n-v}{v} \right\rfloor$ ($N_{minor} \in \mathbb{N}$) as the number of different minor delays. Having defined the required variables, we will now present the algorithm that computes the delays. Algorithm 1 gets two inputs n, v and returns a vector with $N=(N_{major}+1)\cdot(N_{minor}+1)$ unique delays, which guarantees exact MAC for input values less than or equal $\frac{v}{k}$. As an example, we calculate N_{major} = 1 and N_{minor} = 2, when n=16 and v=5. We compute N=6 different delays with Algorithm 1 and get $Delays := \{0, 5, 10, 80, 85, 90\}$. Fig. 3 shows the summation for this example. The first six sequences are the delayed products of two factors ($\frac{5}{16} \cdot \frac{5}{15}$). The last sequence represents the sum, produced using a 6-input OR gate.

TABLE I: Examples for the accuracy threshold $\frac{v}{k}$ for different period lengths $n = k+1$ and number of MAC inputs, see Equation (92)

Calculation with input values below the thresholds is exact.

	Period length $n = k+1$				
	16	32	64	128	256
4 Inputs (N=2)	0.53	0.52	0.51	0.50	0.50
12 Inputs (N=6)	0.33	0.32	0.33	0.33	0.33
24 Inputs (N=12)	0.27	0.26	0.25	0.25	0.25
40 Inputs (N=20)	0.20	0.19	0.19	0.20	0.20
60 Inputs (N=30)	0.13	0.16	0.16	0.17	0.16

Algorithm 1 Computing delays for N summands

```
Input: $n, v$
Output: $Delays$
1: $N_{major} = \left\lfloor \frac{n-2v}{v} \right\rfloor$, $N_{minor} = \left\lfloor \frac{n-v}{v} \right\rfloor$, $i = 1$
2: for $q = 0, 1, 2,$ to $N_{major}$ do
3:     for $p = 0, 1, 2,$ to $N_{minor}$ do
4:         $Delays(i) = q \cdot vn + p \cdot v$
5:         $i = i + 1$
6:     end for
7: end for
8: return $Delays$
```

If input bit-streams have more 1s than allowed by (4), exact MAC operation cannot be guaranteed. In that case, inputs represent values that are greater than $\frac{v}{k}$. Pairwise multiplication of these inputs is exact, but the accumulation of their products causes error. In that circumstance, the upper error bound, E_{upper}, gives the maximum summation error:

$$E_{upper} = \frac{\frac{3}{2}c(c+1) + c(v-1)}{nk} \cdot L \tag{5}$$

where c is the number of extra (and potentially overlapping) 1s per period in the input bit-streams and $v + c$ is the total number of 1s in them. Further, $L(L \leq N | L \in \mathbb{N})$ is the number of products, where inputs represent values in $[0, \frac{v+c}{n}]$. We highlight that L is the number of products, i.e., the number of inputs to the OR gate.

Subsequently, we give two examples for the usage of (5). In the first one, the application guarantees that half of the inputs satisfy (4), but no assumptions are made for the remaining

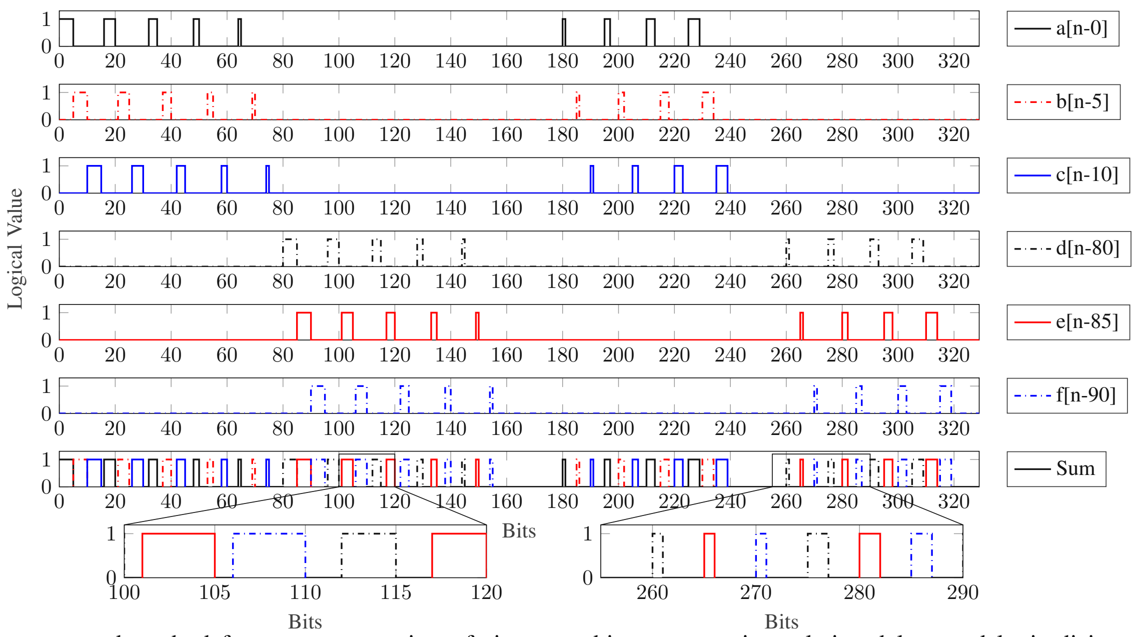

Fig. 3: The proposed method for exact summation of six unary bit-streams using relative delays and logic-disjunction. The summands a, b, c, d, e, f are delayed by $0, 5, 10, 80, 85, 90$ bits, respectively. All summands are equal besides of the different delay ($a = b = c = d = e = \frac{v}{n}\frac{v}{k}$) and sum to $a + b + c + d + e = 6(\frac{v}{n}\frac{v}{k}) = 0.625$, with $n = 16$, $k = 15$ and $v = 5$. The bottom plot shows the logic-OR of upper sequences and represents the exact sum.

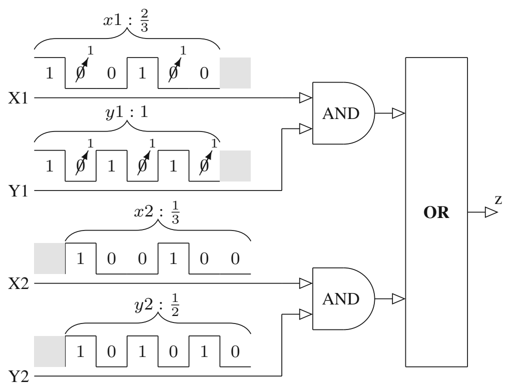

Fig. 4: The proposed MAC technique for four inputs, relative prime lengths $n = k + 1 = 3$ and a relative delay between summands of 1 bit (marked by gray squares). The circuit is exact for inputs with at most 1 bit logic-1 per period (input values $\leq \frac{v}{k} = \frac{1}{2}$). Because $X1$ and $Y1$ exceed this limit by $c = 1$ bit, the result z has a maximum error $E_{upper} = 0.5$ (5).

inputs. This example is meant to point out how to use variable L to consider correct summands during the error estimation. Fig. 4 shows the proposed MAC circuit with two of four input bit-streams ($X1$ and $Y1$) not satisfying (4), having one extra logic-1 ($c = 1$) per period. Because both are connected to the same AND gate, one product ($L = 1$) has too many 1s for exact summation. We use short relative prime periods $n = k + 1 = 3$ for better graphic display. The circuit computes 1101100 which represents $\frac{4}{6}$, the number of 1s divided by the least common multiple (LCM) of the input periods. The MAC error is $\frac{5}{6}$-$\frac{4}{6} = \frac{1}{6}$, which is less than the upper error bound $E_{upper} = \frac{\frac{3}{2}\cdot 1\cdot(1+1)+1\cdot(1-1)}{3\cdot 2}\cdot 1 = \frac{1}{2}$.

In the second example, all input values stay in the allowed $[0, \frac{v}{k}]$ interval. The results are exact, when all input values are less than or equal $\frac{v}{k} = 0.315$, with $n = k+1 = 16$, $v = 5$ and $N = 6$. The maximum exact result is $z = \sum_1^6 \frac{5}{16}\cdot\frac{5}{15} = 0.625$, when all input bit-streams have $v = 5$ 1s per period. In contrast to the first example, all input values exceed the threshold at the same time. Now assume that all 12 MAC inputs ($L = 6$) have one additional bit toggled from 0 to 1 ($c = 1$). The output increases to $z = \sum_{i=1}^{N} \frac{6}{16}\cdot\frac{6}{15} = 0.9$, whereas the proposed method computes 0.8625. The error $0.9 - 0.8625 = 0.0375$ is less than the upper-error estimation in (5):

$$E_{upper} = [\frac{3}{2}\cdot 1(1+1) + 1\cdot(5-1)]/(16\times 15)\times 6 = 0.175 \quad (6)$$

In the next section, we prove that the proposed method is exact when condition (3) or (4) is satisfied.

IV. Proof of the Theorem

Assume S_n and S_k are two unary bit-streams with v 1s, followed by $n - v$ and $k - v$ 0s. Their product is computed by feeding k repetitions of S_n and n repetitions of S_k to an AND gate as elaborated in [7]. The n times repetition of S_k will be denoted as $S_{k,n}$. Similarly, $S_{n,k}$ is the k times repeated sequence of S_n. $S_{n,k}$, $S_{k,n}$ and their logic-conjunction S_{AND} are piecewise defined as

$$S_{n,k}[i] = \begin{cases} 1 & \{nx \leq i < nx + v\} \quad \text{(a)} \\ 0 & \{nx + v \leq i < n(x+1)\} \quad \text{(b)} \end{cases} \quad (7)$$

$$S_{k,n}[i] = \begin{cases} 1 & \{ky \leq i < ky + v\} \quad \text{(a)} \\ 0 & \{ky + v \leq i < k(y+1)\} \quad \text{(b)} \end{cases} \quad (8)$$

$$S_{\text{AND}}[i] = \text{AND}(S_{n,k}, S_{k,n}) \quad (9)$$

$$\text{with } \{0 \leq x < k, 0 \leq y < n | x, y \in \mathbb{N}\}. \quad (10)$$

Note that, in the binary domain, (7) and (8) are factors of the product represented by (9). Fig. 5 shows these sequences for $n = 5$, $k = 4$ and $v = 2$. The subsequent proof of (3) is divided into two parts. We begin by deriving positions of 1s in S_{AND}. We will then use this information to derive the relative delays that lead to no overlap between 1s. Using these delays will guarantee exact summation through logic-disjunction.

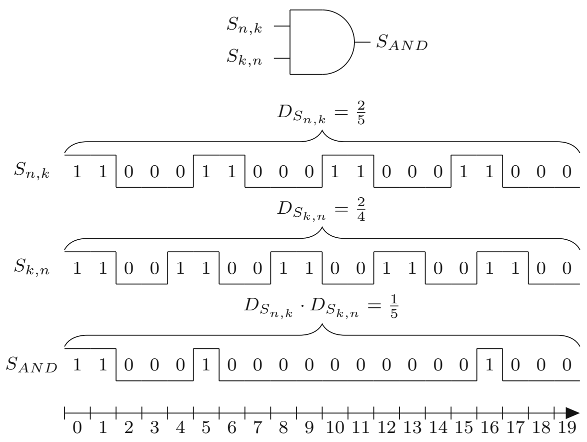
$S_{n,k}$
$S_{k,n}$
S_{AND}
$D_{S_{n,k}} = \frac{2}{5}$
$S_{n,k}$ 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0
$D_{S_{k,n}} = \frac{2}{4}$
$S_{k,n}$ 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
$D_{S_{n,k}} \cdot D_{S_{k,n}} = \frac{1}{5}$
S_{AND} 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Fig. 5: Stochastic multiplication of two unary bit-streams using AND gate. $S_{n,k}$ represents $\frac{v}{n} = \frac{2}{5}$ with $v = 2$ 1s and a period of $n = 5$ bits (Duty Cycle $D_{S_{n,k}=0.4}$). $S_{k,n}$ represents $\frac{v}{k} = \frac{2}{4}$ with a period of $k = 4$ bits (Duty Cycle $D_{S_{k,n}=0.5}$). The output S_{AND} represents $\frac{1}{5}$.

By definition $\texttt{AND}(S_{n,k}[i], S_{k,n}[i])$ produces a 1 when

$$\{S_{n,k}[i] = 1 \quad \texttt{AND} \quad S_{k,n}[i] = 1\}. \tag{11}$$

It follows that both inequations, (7,a) and (8,a) are required to be fulfilled to get a 1 in S_{AND}.

$$\{nx \leq i < nx + v \quad \text{and} \quad ky \leq i < ky + v\} \tag{12}$$

We use natural numbers m, t to rewrite inequations (12) as

$$\{nx \leq i < nx + v\} = \{i = nx + m\} \tag{13}$$
$$\{ky \leq i < ky + v\} = \{i = ky + t\} \tag{14}$$
$$\text{with } \{0 \leq m < v, 0 \leq t < v | m, t \in \mathbb{N}\}.$$

Equations (12) to (14) lead to two possible substitutions:

$$\begin{aligned}
&\{\{nx \leq i < nx + v \quad \text{and} \quad i = ky + t\} \quad \text{or} \\
&\quad \{ky \leq i < ky + v \quad \text{and} \quad i = nx + m\}\} && (15)\\
&= \{\{nx \leq ky + t < nx + v\} \quad \text{or} \\
&\quad \{ky \leq nx + m < ky + v\}\} && (16)\\
&= \{\{-t \leq ky - nx < v - t\} \quad \text{or} \\
&\quad \{-m \leq nx - ky < v - m\}\} && (17)\\
&= \{\{v - t > ky - nx \geq -t\} \quad \text{or} \\
&\quad \{m \geq ky - nx > m - v\}\} && (18)
\end{aligned}$$

The disjunction in (18) has two maximum solutions ($m = t = 0$ and $m = t = v - 1$) that both can be simplified to (19).

$$\{v > ky - nx \geq 0 \quad \text{or} \quad 0 \geq ky - nx > -v\} \tag{19}$$
$$v > |ky - nx| \tag{20}$$

For given k and n there are different solutions to (20). We set $k = n - 1$ for the reasons mentioned in Section I. Hence,

$$v > |n(y - x) - y| \tag{21}$$

A closer look at (21) and constraint $n > v$ reveals that no solution is possible for $y \leq x - 1$ and $y \geq x + 2$ because both equations result in a contradiction:

$$\begin{aligned}
&\text{for } y \leq x - 1 \rightarrow y = x - 1 - m \\
&\quad v > |n(y - x) - y| \overset{y=x-1-m}{=} |-n - mn - y| \not< v && (22)\\
&\text{for } y \geq x + 2 \rightarrow y = x + 2 + m \\
&\quad v > |n(y - x) - y| \overset{y=x+2+m}{=} |2n + mn - y| \not< v && (23)\\
&\quad \text{with } \{m \geq 0 | m \in \mathbb{N}\}
\end{aligned}$$

Next, we find explicit solutions for x, knowing that the only possible values for y in (21) are (I.) $y = x$ and (II.) $y = x + 1$.

$$\begin{aligned}
v > |ky - nx| = |ny - y - nx| && (24)\\
\text{for (I)} \quad y = x \\
v > |nx - x - nx| = x \rightarrow x < v && (25)\\
\text{for (II)} \quad y = x + 1 \\
v > |nx + n - x - 1 - nx| = |n - x - 1| && (26)\\
\overset{x<n}{=} n - 1 - x \overset{k=n-1}{=} k - x \rightarrow x > k - v && (27)
\end{aligned}$$

Therefore, 1s in S_{AND} only appear in the beginning and end ($x < v$ and $x > k - v$). Substituting for y in (12) leads to (30) and (33) that describe 1s of S_{AND}, if both factors have v 1s.

$$\begin{aligned}
&\overset{y=x}{\rightarrow} \{nx \leq i < nx + v \text{ and } kx \leq i < kx + v\} && (28)\\
&= \{\max(nx, kx) \leq i < \min(nx + v, kx + v)\} && (29)\\
&= \{nx \leq i < kx + v\} \text{ for } x < v && (30)
\end{aligned}$$

$$\begin{aligned}
&\overset{y=x+1}{\rightarrow} \{nx \leq i < nx + v \text{ and } kx + k \leq i < kx + k + v\} && (31)\\
&= \{\max(nx, kx + k) \leq i < \min(nx + v, kx + k + v)\} && (32)\\
&= \{k(x + 1) \leq i < nx + v\} \text{ for } x > k - v. && (33)
\end{aligned}$$

$$S_{\text{AND}}[i] = \begin{cases}
1 & \{nx \leq i < kx + v\} \quad \text{(a)} \\
 & \text{for } 0 \leq x < v \\
1 & \{k(x + 1) \leq i < nx + v\} \quad \text{(b)} \\
 & \text{for } k > x > k - v \\
0 & \text{else} \quad \text{(c)}
\end{cases} \tag{34}$$

We will follow the common notation $|I|$ for the length of interval I. The length of an interval is the absolute value of the difference between the two endpoints. The minimal interval with property $\{\{nx \leq i < kx + v\} \text{ for } x < v\} \subset I_{begin}$ that includes all sections of (34,a) is

$$I_{begin} = [0, k(v - 1) + v) \tag{35}$$
$$|I_{begin}| = kv + v - k = nv - k. \tag{36}$$

In the same manner, we define I_{end} for (34,b) and get

$$I_{end} = [k(k - v + 2), n(k - 1) + v) \tag{37}$$
$$|I_{end}| = nv - n - k. \tag{38}$$

Observe that $|I_{begin}|$ is greater than $|I_{end}|$ regardless of v.

$$\max(|I_{begin}|, |I_{end}|) = |I_{begin}| = nv - k \tag{39}$$

Both I_{begin} and I_{end} contain a certain percentage of 1s proportional to the values of factors. On the contrary, $I_{F,major}$ contains 0s only and is between I_{begin} and I_{end}.

$$\begin{aligned} I_{F,major} &= [nv, n(k - v + 1)) \\ &= [nv, n(n - v)) \end{aligned} \tag{40}$$

$$\begin{aligned} |I_{F,major}| &= n(n - v) - (nv) \\ &= n^2 - 2nv. \end{aligned} \tag{41}$$

Knowing $q \cdot vn \geq |I_{begin}|$ $(1 \leq q | q \in \mathbb{N})$, a relative delay of $q \cdot vn$ bits between two summands avoids overlap between their I_{begin}. As a result of $|I_{begin}| > |I_{end}|$ it also avoids overlap between their I_{end}. Delays of form $q \cdot vn$ will be called major delays. We write $S_{\text{AND},q}$ for major-delayed products and $I_{begin,q}$, $I_{end,q}$, $I_{F,major,q}$ for their right-shifted intervals:

$$S_{\text{AND},q}[i] = S_{\text{AND}}[i - q \cdot vn] \tag{42}$$

The task is now to find maximum $q_{max} = N_{major}$ that $I_{begin,N_{major}}$ does not intersect with I_{end} of other products.

$$|I_{F,major}| - (vn)N_{major} \geq 0 \tag{43}$$

$$N_{major} \leq \frac{|I_{F,major}|}{vn} \tag{44}$$

$$N_{major} \leq \frac{n - 2v}{v} \tag{45}$$

It follows that interval $I_{begin,q}$ $(1 \leq q \leq N_{major})$ is in $I_{F,major,q-1}$ and $I_{end,q}$ is right-shifted out of $S_{\text{AND},q-1}$. Therefore, $N_{major} + 1 = \lfloor \frac{n-v}{v} \rfloor$ major-delayed products do not create overlap in logic-disjunction, when each of them is uniquely delayed.

We can now consider the derivation of N_{minor} and factor $\lfloor \frac{v}{n} \rfloor$ in (3). By definition of logic-AND, 0s in $S_{n,k}$ (and $S_{k,n}$) also appear in S_{AND}. Substituting $i \to i - q \cdot vn$ into (7,b) shows that applying major delays extend and do not disturb the periodicity of 0s. Therefore, $S_{\text{AND},q}$ and the logic-disjunction of them share a regular occurring pattern of 0s, denoted $I_{F,minor}$.

$$S_{\text{AND},q}[i] = 0 \text{ for } \{n(x + qv) + v \leq i < n(x + 1 + qv)\} \tag{46}$$

$$I_{F,minor} = [n(x + qv) + v, n(x + 1 + qv)) \tag{47}$$

$$|I_{F,minor}| = n(x + qv + 1) - (n(x + qv) + v) \tag{48}$$

$$= n - v \tag{49}$$

Let $I_{two-level}$ be a recurring interval that does not intersect with $I_{F,minor}$. $I_{two-level}$ has 1s proportional to the input values and contains all 1s of $S_{\text{AND},q}$. The exact positions of 1s are not required for this proof.

$$\begin{aligned} I_{two-level} &= [n(x + qv), n(x + qv) + v) \\ &\quad \text{for } 0 \leq x < v \text{ and } k > x > k - v \\ &\quad \text{with } I_{two-level} \cap I_{F,minor} = \{\} \end{aligned} \tag{50}$$

$$|I_{two-level}| = v \tag{51}$$

According to (49) and (51), $S_{\sum \text{AND},q}$ consists of an alternating pattern of v logical-undetermined bits, followed by $n - v$ bits being guaranteed 0s. Assuming $|I_{F,minor}| \geq |I_{two-level}|$, a relative minor-delay of $|I_{two-level}| = v$ between $S_{\text{AND},q}$ avoids overlap at logic-disjunction, because $I_{F,minor}$ exclusively has 0s. We now proceed similar to (45) and compute the number of unique minor-delays N_{minor}.

$$|I_{F,minor}| - |I_{two-level}| \cdot N_{minor} \geq 0 \tag{52}$$

$$N_{minor} \leq \frac{|I_{F,minor}|}{|I_{two-level}|} = \frac{n - v}{v}. \tag{53}$$

The delay for each summand is computed with two variables $0 \leq q \leq N_{major}$ and $0 \leq p < N_{minor}$ with $\text{Delay}(q, v) = q \cdot vn + p \cdot v$. In its final form $S_{\text{AND},q,p}[i] = S_{\text{AND},q}[i - p \cdot v] = S_{\text{AND}}[i - q \cdot vn - p \cdot v]$ is defined as

$$S_{\text{AND, q, p}}[i] = \begin{cases} 1 & \{nx + q \cdot vn + p \cdot v \leq i < kx + q \cdot vn + (p + 1) \cdot v\} \\ & \text{for } x < v \\ 1 & \{k(x + 1) + q \cdot vn + p \cdot v \leq i < nx + q \cdot vn + (p + 1) \cdot v\} \\ & \text{for } x > k - v \\ 0 & \text{else} \end{cases} \tag{54}$$

The result of MAC operation is

$$S_{SUM}[i] = \sum_{q=0,p=0}^{N_{major},N_{minor}} S_{\text{AND}}[i - q \cdot vn - p \cdot v] \tag{55}$$

Counting the number of different available delays $(N \in \mathbb{N})$, that is the possibilities for q and p leads to the claim in (3):

$$N \leq (N_{major} + 1)(N_{minor} + 1) \tag{56}$$

$$N \leq \left\lfloor \frac{n - 2v}{v} + 1 \right\rfloor \left\lfloor \frac{n - v}{v} + 1 \right\rfloor = \left\lfloor \frac{n - v}{v} \right\rfloor \left\lfloor \frac{n}{v} \right\rfloor \tag{57}$$

The maximum integer function in (57) is required because N_{minor} and N_{major} are naturals. Solving (57) for v is possible because of the property $N_{minor} = N_{major} + 1$. Since N is the product of two consecutive integers we call N a pronic number ($N = x(x+1)$ with $x = N_{major}+1$). By analogy with the square root of N, the pronic-root of N is $x = \frac{\sqrt{4N+1}-1}{2}$. Knowing N we can calculate its N_{major} with

$$N_{major} + 1 = \left\lceil \frac{\sqrt{4N + 1} - 1}{2} \right\rceil \tag{58}$$

Next, we solve (45) for natural v and substitute for $N_{major} + 1$. The result is equal to claim (4).

$$v \leq \left\lfloor \frac{n}{N_{major} + 2} \right\rfloor \tag{59}$$

$$v \leq \left\lfloor \frac{n}{\lceil \frac{\sqrt{4N+1}-1}{2} \rceil + 1} \right\rfloor \tag{60}$$

In the following section, we discuss the behavior of the proposed technique when the maximum input constraint (4) is not satisfied. We derive a maximum error-bound and prove that the error is less than E_{upper} from (5).

V. Proof of the Upper Error Bound

This section proves that inputs representing greater values than (4) cause a maximum error given in (5). So far, we have been working under the assumption that the system was designed for N summands and input values are in the $[0, \frac{v}{k}]$ interval. Now suppose that the allowed input interval expands to $[0, \frac{v+c}{n}]$ and N is not reduced. To account for greater input values, we substitute $v \to v+c$ in (34,a) and (34,b), and write S'_{AND} for products (summands) with increased input values.

$$S'_{\text{AND}} = \begin{cases} 1 \quad \{nx \le i < kx + (v+c)\} & \text{(a)} \\ \quad \text{for } 0 \le x < (v+c) \leftrightarrow x \in \{x_0, x_1\} & \\ 1 \quad \{k(x+1) \le i < nx + (v+c)\} & \text{(b)} \\ \quad \text{for } k > x > k-(v+c) \leftrightarrow x \in \{x_3, x_4\} & \\ 0 \quad \text{else} & \text{(c)} \end{cases} \tag{61}$$

Partitioning index x ($0 \le x < k$) into five shorter indexes ($x = x_0 \cup x_1 \cup x_2 \cup x_3 \cup x_4$) simplifies the subsequent error analysis.

$$0 \le x_0 < v \tag{62}$$
$$v \le x_1 < (v+c) \tag{63}$$
$$(v+c) \le x_2 < k-(v+c)+1 \tag{64}$$
$$k-(v+c)+1 \le x_3 < k-v+1 \tag{65}$$
$$k-v+1 \le x_4 < k \tag{66}$$

Whenever we analyze one specific section of S_{AND}, we abbreviate intervals of S_{AND} with side condition $x = x_i \in \{x_0, x_1 \ldots x_6\}$ to S'_{AND,x_i}:

$$S'_{\text{AND},x_i} = S_{\text{AND}} \text{ and } x = x_i \tag{67}$$

Logic-1s within previously assumed logic-0 intervals produce error at logic-OR, because they can overlap with 1s of other delayed summands. The proposed upper-error bound assumes all extra 1s to cause error and sums the individual components. In what follows, the error is calculated by intersecting both intervals ((47) and (40)) with 1s in S'_{AND} (61,a) and (61,b). The intersection between intervals is defined as $I_i \cap I_j$. We use variable $M_{i,j}$ for the number of bits in the intersections.

$$I_1 \cap I_2 = [a,b) \cap [c,d) \tag{68}$$
$$= [\max(a,c), \min(b,d)) \tag{69}$$
$$M_{1,2} = |I_1 \cap I_2| \tag{70}$$
$$= \min(b,d) - \max(a,c) \tag{71}$$

There is no intersection between $I_{F,major}$ and $S_{\text{AND},x0}$, because $max(S_{\text{AND},x_0}) < min(I_{F,major})$. Same technique for x_1 shows that in the worst case, $S'_{\text{AND},x1}$ is in $I_{F,major}$, so all 1s of $S'_{\text{AND},x1}$ could cause error. We sum all bits of $|S'_{\text{AND},x1}|$ using Gauss-Sum for natural numbers:

$$M_{x_1,F_{major}} = |S'_{\text{AND},x_1} \cap I_{F,major}| \tag{72}$$
$$= \sum_{x=x_1} \min(kx+(v+c), n(n-v))$$
$$- \max(nx, nv) \le \sum |S'_{\text{AND},x_1}| \tag{73}$$
$$= \sum_{x=x_1} |[nx, kx+v+c)| \tag{74}$$
$$= \sum_{x=v}^{v+c-1} v+c-x = \frac{c(c+1)}{2} \tag{75}$$

By definition $S'_{\text{AND},x_2}[i]$=0 regardless of inputs and $M_{x_2,F_{major}}$ is zero as a result. In worst case, $S'_{\text{AND},x3}$ is completely in $I_{F,major}$, which can be seen when comparing their maximum and minimum. Hence, we again sum bits in S'_{AND,x_3}.

$$M_{x_3,F_{major}} = |S'_{\text{AND},x_3} \cap I_{F,major}| \le \sum |S'_{\text{AND},x_3}| \tag{76}$$
$$= \sum_{x=x_3} |[k(x+1), (k+1)x + (v+c))| \tag{77}$$
$$= \sum_{x=k-(v+c)+1}^{k-(v)} (v+c) - k + x = \frac{c(c+1)}{2} \tag{78}$$

Similar to x_0 there is no error for x_4, because $\max(I_{F,major}) < min(S'_{\text{AND},x_4})$. In (47) we showed that $I_{F,minor}$ does not intersect with 1s in S_{AND} and contains 0s only. For x_1, x_2, x_3, $I_{F,minor}$ is in $I_{F,major}$, so error within this interval are already covered. Substituting $v \to (v+c)$ in (50) leads additional error for x_0 and x_4:

$$M_{x_0,F_{minor}} = |I_{F,minor} \cap |S'_{\text{AND},x_0}| \tag{79}$$
$$= \sum_{x=x_0} [nx+v, n(x+1)) \cap [nx, kx+(v+c)) \tag{80}$$
$$= \sum_{x=x_0} \min(n(x+1), kx+(v+c))$$
$$- \max(nx+v, n(x+1)) \tag{81}$$
$$= \sum_{x=x_0} kx + (v+c) - nx - v \tag{82}$$
$$= \sum_{x=0}^{v-1} c - x \le \frac{c(c+1)}{2} \tag{83}$$

Repeating the same concept for $I_{F,minor} \cap S'_{\text{AND},x_4}$ with $\max(nx+v, k(x+1)) = nx+v$ gives $M_{x_4,F_{minor}}$.

$$I_{F,minor} \cap S'_{\text{AND},x_4} = [nx+v, n(x+1))$$
$$\cap \{[k(x+1), nx+(v+c)) \text{ for } k > x > k-v\} \tag{84}$$
$$= \min(n(x+1), nx+(v+c))$$
$$-\max(nx+v, k(x+1)). \tag{85}$$

$$M_{x_4,F_{minor}} = |I_{F,minor} \cap S'_{\text{AND},x_4}| \tag{86}$$
$$= \sum_{x=x_4} (nx+(v+c)) - (nx+v) \tag{87}$$
$$= \sum_{x=k-v+1}^{k-1} c = (v-1)c \tag{88}$$

M_{total} is the sum of individual error contributing intersections.

$$M_{total} = \sum M_{x_i} = \frac{3}{2}c(c+1) + c(v-1) \tag{89}$$

Up to this point we considered only one summand having extra 1s that cause error. This is the result of two MAC inputs having $v+c$ instead v 1s. When L ($L \in \mathbb{N}$) summands exceed the allowed input range, the error increases proportional to L. The number of potential error bits is converted to the positional binary representation E_{upper} by scaling with $\frac{1}{nk}$.

$$E_{upper} = \frac{\frac{3}{2}c(c+1) + c(v-1))}{nk} \cdot L \tag{90}$$

The upper error limit E_{upper} increases linear w.r.t. v and L, and quadratic with c. Note that $E_{upper} = 0$ (the computation is accurate), when $c = 0$ (Equation (4) is satisfied).

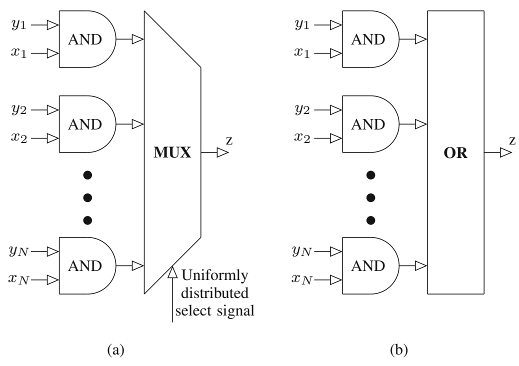

Fig. 6: Two common SC MAC: (a) `AND` gate followed by a multi-input MUX, (b) `AND` gate followed by a multi-input `OR`.

VI. Experimental Result

In this section, we verify our theory by using circuit implementations of the proposed technique. We compare the proposed unary bit-stream-based MAC with three state-of-the-art designs which represent data using Sobol-based low-discrepancy (LD) bit-streams [20] while perform the summation using either MUX or `OR` gates. Since all four designs use `AND` gates for multiplication, we distinguish them by their data representation and summation method, and call them $\text{Sobol}_{\texttt{MUX}}$, $\text{Sobol}_{\texttt{MUX/TFF}}$, $\text{Sobol}_{\texttt{OR}}$ and $\text{Unary}_{\texttt{OR}}$. Below, we discuss the MUX-based methods ($\text{Sobol}_{\texttt{MUX}}$ and $\text{Sobol}_{\texttt{MUX/TFF}}$) and `OR`-based method ($\text{Sobol}_{\texttt{OR}}$) and their corresponding equations and compare them to our proposed technique.

A. Traditional summation

Fig. 6(a) shows the most common SC design for MAC units. Multiplications are performed using `AND` gates and summation is implemented with MUX unit [3] [21] [22]. A MUX-based adder divides the sum by the number of data inputs N. The MAC function is described as follows:

$$z_{\text{MUX}} = \frac{\sum_{1\le i\le N} x_i y_i}{N} \tag{91}$$

where x_i and y_i are inputs, and z_{MUX} is the output. `OR` gates replace the MUX unit in an alternative MAC circuit [23] [16] (see Fig. 6(b)). `OR` gates find the union of input bit-streams. For example, the result of a three-input `OR` is $z_{OR} = a_1 \cup a_2 \cup a_3 = a_1 + a_2 + a_3 - a_1a_2 - a_1a_3 - a_2a_3 + a_1a_2a_3$. In a two-stage stochastic MAC, a_is are the outputs of `AND` gates ($a_i = x_iy_i$ and $i = 1, 2, \dots N$).

Recent works on deterministic methods of SC [8] [7] [5] showed that completely accurate computations can be done using SC designs. Different deterministic approaches were proposed based on LD [24], pseudo-random [5], and unary bit-streams [8]. All these methods produce completely accurate output when processing bit-streams with a specific length (i.e., $2^{N\times M}$ bits where N is the number of inputs and M is the precision of data) and decrease in accuracy when shorter bit-streams are processed. Among these, Sobol-based LD methods [20] [25] have shown minimum random fluctuations and fastest convergence to the expected output [5]. The authors in [5] showed exact and fast converging multiplication with

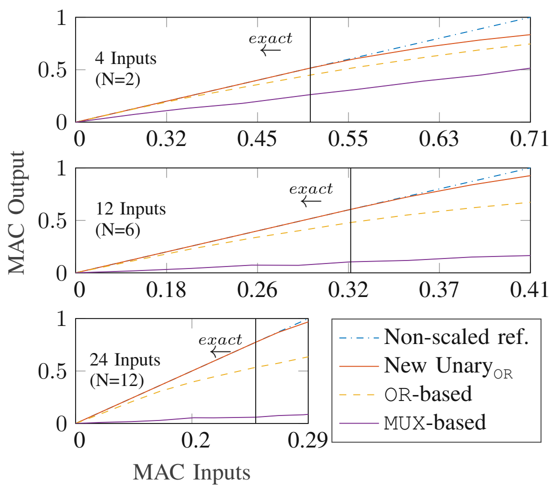

Fig. 7: MAC outputs for `OR`-based and MUX-based as well as the proposed $\text{Unary}_{\texttt{OR}}$ as a function of input value and number of inputs. Up to the vertical line (thresholds from Table I), the proposed $\text{Unary}_{\texttt{OR}}$ is exact and provides the same output as a non-scaled MAC, shown as diagonal line.

`AND` gates and scaled addition with MUX when processing Sobol-based bit-streams. The required independence between Sobol bit-streams is provided by generating each bit-stream based on a different Sobol sequence. When used with `OR` gates, Sobol bit-streams produce the union of the inputs with improved latency compared to pseudo-random and unary bit-streams. We compare our proposed MAC design with three baseline designs processing bit-streams of this form.

Fig. 7 presents the outputs for approximate `OR`-based MAC, scaled MUX-based MAC and the proposed $\text{Unary}_{\texttt{OR}}$ to show the input-ouput behavior of each method. Note that the number of MAC inputs is twice that of N due to the pairwise multiplication before summation. The 5-bit precision ($n = 32, k = 31$) input values are marked on the X-axis and are equally increased until the reference MAC output, shown on the y-axis, reaches 1 ($z = \sum_{i=1}^{N} x_iy_i \overset{!}{=} 1$). The non-scaled reference MAC output is a diagonal line because MAC is a linear operation. While each of the implemented methods exhibits some degree of deviation compared to the reference, MUX-based methods diverge the most from the non-scaled reference, because a MUX unit performs scaled rather than normal addition. Traditional `OR`-based methods perform comparably to the proposed $\text{Unary}_{\texttt{OR}}$ design for small input values but falls behind at larger ones. $\text{Unary}_{\texttt{OR}}$ is closer to the non-scaled reference MAC than MUX-based and `OR`-based designs regardless of the number of inputs (e.g., 4, 12, and 24) and is the only method that can provide completely accurate results for small input values. The thresholds in which the proposed method stops producing exact result is marked by vertical lines in Fig. 7. These match the thresholds from Table I after quantization. For example, the computed threshold for $n = 32$ and $N = 2$ is 0.52 that, after rounding to the next representable value, becomes 0.5. A 32-bit unary bit-stream represents 0.5 with 16 1s followed by 16 0s. For input values greater than the threshold, the proposed method guarantees the upper-error bound (5) discussed in Section V.

B. Practical Implementation

So far we distinguished the individual methods by their summation technique (OR-based or MUX-based) and the SNGs used. In this section, we further refine our MAC techniques and introduce Sobol$_{\texttt{MUX/TFF}}$, Unary$_{\texttt{OR/REG}}$ and Unary$_{\texttt{OR/SEQ}}$. The latter two are alternative designs for the proposed MAC technique, which achieve the same accuracy results as Unary$_{\texttt{OR}}$, but require different resources. Hence, we will not discuss them in accuracy comparisons. However, their differences will be discussed in the resource comparison section (Section VII). Sobol$_{\texttt{MUX/TFF}}$ is a specific implementation of MUX-based addition that does not require an additional select input. The method is extensively discussed in [14]. Both, the accuracy and the resource requirements differ from Sobol$_{\texttt{MUX}}$, so it will be separately discussed in our accuracy and resource comparisons. The three Unary$_{\texttt{OR}}$ methods reflect three examples of making relative delays between summands:

1) Unary$_{\texttt{OR}}$ uses a separate SNG (consists of a counter and a comparator) for each MAC input. The SNGs are enabled pairwise at the clock cycles calculated with Algorithm 1 for nk clock cycles. Two SNGs that are connected to the same AND gate form one pair. The counters have relative prime periods n and $k = n - 1$. The bit-streams are 0, when the SNGs are disabled. The computation architecture is equal to Fig. 2(b). Fig. 3 visualizes the concept of summing *delayed* bit-streams. Each SNG is enabled for 240 clock cycles in unique intervals and disabled (logic 0) for the remaining time.

2) Unary$_{\texttt{OR/REG}}$ differs in both bit-stream generation and cause of relative delays. It uses a pair of two counters instead of one per MAC input (i.e., a total of $2N$ counters). The values of the two counters are compared with each pair of factors. The relative delays are caused by bit-shift registers of different lengths between the outputs of the AND gates and the inputs of the OR gates. The length of the shift register for summand i is equal to $Delays(i)$ computed with Algorithm 1. In Fig. 2(b), the length of the shift register after AND gate with inputs x_1, y_1 is $Delays(1)$ (always 0), after AND gate with inputs x_2, y_2 is $Delays(2)$, and so on. It is important that the output bit stream of a SNG is 0, when the bit-stream generation is finished.

3) Unary$_{\texttt{OR/SEQ}}$ uses one SNG pair (one for each relatively prime length bit-stream) and the architecture in Fig. 2(a). The MAC works sequentially and the summands are added to the accumulator one after another. To achieve the same accuracy and output sequence the following two properties are required. First, the inputs to the SNG pair must change N times during MAC operation (once for each summand). Second, the SNG pair must be disabled for the correct number of cycles between generating bit-streams for different inputs, so that the new summand is added to the accumulator at the right time. The accumulator is a shift register of length $nk{+}N_{major}{\cdot}vn{+}\ N_{minor}{\cdot}v$ (the LCM of inputs plus the maximum delay from Algorithm 1). The SNGs must be stalled for $N_{major} \cdot vn + N_{minor} \cdot v$ cycles between generating the bit-streams of two subsequent input pairs. Simulations show that it is possible to use a different buffer size and avoid stalling

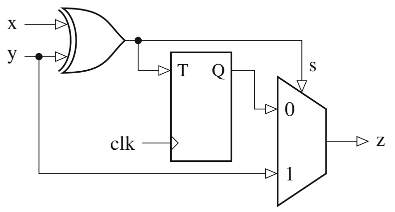

Fig. 8: Toggle flipflop based scaled adder implementation (Sobol$_{\texttt{MUX/TFF}}$) that does not need an additional RNG for the select input of the MUX [14].

the bit-stream generation (i.e., immediately generate the next bit-stream when the previous is done). However, finding a formula for optimal buffer sizes or proofing 100% accuracy for sequential computation without separately controlling the bit-stream generation requires further investigation and is part of our future work.

All Sobol-based methods use the architecture in Fig. 2(b), but could also be adapted to use the sequential architecture of Fig. 2(a).

1) Sobol$_{\texttt{OR}}$ approximates non-scaled summation with a multi-input OR gate and requires N SNGs, because the AND gates as well as the OR gate require uncorrelated inputs.

2) Sobol$_{\texttt{MUX}}$ uses a multi-input MUX unit for scaled summation. The select input comes from a separate RNG that produces uniformly distributed random integer variables in the $[0,N-1]$ interval. The inputs of each multiplication operation need to be uncorrelated, but the MUX inputs can be correlated [3]. Hence, two RNGs can be shared to generate all inputs of multiplication operations [5]. This method therefore requires three uncorrelated streams of random numbers. Two, to make the factor bit-streams (connected to AND gates) uncorrelated and one for the select input of the MUX.

3) Sobol$_{\texttt{MUX/TFF}}$ uses multiple 2-to-1 multiplexer stages shown in Fig. 8 for scaled summation without requiring an additional input (and RNG) for the select bit-stream. The circuit consists of a toggle flipflop and an XOR gate. Sobol$_{\texttt{MUX/TFF}}$ requires two streams of random numbers, which is the least of all methods in the parallel architecture.

C. Accuracy Comparison

a) Sobol$_{MUX}$ and Sobol$_{MUX/TFF}$: The accuracy of a stochastic MUX-adder decreases when the length of bit-stream is fixed but the number of inputs increases [26]. For input values close to 1, the $\frac{1}{N}$ factor in (91) helps keeping the result in the $[0, 1]$ interval. For input values close to zero, however, the result tends to be too small for accurate representation. Sobol$_{\texttt{MUX}}$ produces additional occurs if the select input of the MUX is correlated to its inputs. As argued in [24], different Sobol sequences should be used to generate independent bit-streams and avoid the correlation error.

b) Sobol$_{OR}$: The accuracy of the OR-based adder is low when input values are close to one ($x_i \approx x_i^2$) and is high when input values are close to zero ($x_i >> x_i^2$). Additional error occurs if input bit-streams are correlated; any overlap between the location of 1s in the input bit-streams decreases the accuracy.

TABLE II: Mean Absolute Error (MAE) (%) comparison of the proposed Unary$_{\mathrm{OR}}$ with state-of-the-art design approaches for different number of summands N and different ranges of (5 bit precision) uniformly distributed random input values. The result of MUX-based methods is rescaled by constant N in a second processing step to compensate the MUX scaling

Mean Reference MAC Output		0.05	0.15	0.25	0.35	0.45	0.55	0.65	0.75	0.85	0.95
		Range of Input Values									
4 Inputs (N=2)		[0,0.31]	[0,0.54]	[0,0.71]	[0,0.84]	[0,0.95]	[0,1.05]	[0,1.14]	[0,1.22]	[0,1.31]	[0,1.38]
	Unary$_{\mathrm{OR}}$	**0.0**	**0.0**	0.3	1.4	3.2	5.5	8.1	10.0	12.4	14.4
	Sobol$_{\mathrm{OR}}$	0.1	0.5	1.5	3.1	5.1	7.5	10.1	12.1	14.4	16.4
	Sobol$_{\mathtt{MUX/TFF}}$-rescaled	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	Sobol$_{\mathtt{MUX}}$-rescaled	0.4	0.7	0.9	1.0	1.2	1.3	1.4	1.4	1.5	1.5
		Range of Input Values									
12 Inputs (N=6)		[0,0.18]	[0,0.31]	[0,0.41]	[0,0.48]	[0,0.55]	[0,0.60]	[0,0.66]	[0,0.71]	[0,0.75]	[0,0.80]
	Unary$_{\mathrm{OR}}$	**0.0**	**0.0**	0.2	1.0	2.6	4.6	7.2	10.5	14.8	19.6
	Sobol$_{\mathrm{OR}}$	0.4	1.1	2.5	4.7	7.7	10.9	15.1	19.6	25.1	30.9
	Sobol$_{\mathtt{MUX/TFF}}$-rescaled	0.5	1.5	2.6	3.6	4.6	5.5	6.6	7.6	8.7	9.8
	Sobol$_{\mathtt{MUX}}$-rescaled	0.9	1.6	2.1	2.4	2.8	3.1	3.3	3.5	3.8	4.0
		Range of Input Values									
24 Inputs (N=12)		[0,0.13]	[0,0.22]	[0,0.29]	[0,0.34]	[0,0.39]	[0,0.43]	[0,0.46]	[0,0.50]	[0,0.53]	[0,0.56]
	Unary$_{\mathrm{OR}}$	**0.0**	**0.0**	**0.0**	0.6	1.8	3.6	6.1	9.0	12.5	16.5
	Sobol$_{\mathrm{OR}}$	0.8	1.6	3.3	5.6	8.7	12.1	16.6	21.3	26.7	32.5
	Sobol$_{\mathtt{MUX/TFF}}$-rescaled	0.8	1.4	2.0	2.7	3.4	4.1	4.8	5.5	6.3	6.8
	Sobol$_{\mathtt{MUX}}$-rescaled	1.3	2.4	3.1	3.7	4.1	4.6	4.9	5.3	5.7	5.9

c) Unary$_{OR}$: The proposed method is accurate (4) for input values less than

$$\frac{v}{k} \leq \left\lfloor \frac{n}{\lceil \frac{\sqrt{4N+1}-1}{2} \rceil + 1} \right\rfloor \frac{1}{k}. \tag{92}$$

The accuracy decreases for input values greater than this threshold. A requirement for exact computation is applying the relative delays from Algorithm 1 before summation. The proposed method has a maximum error of (5) for input values greater than (92). As shown in Table I, this threshold highly depends on N, the number of summands. Note that Algorithm 1 must be modified for input values closer to 1. We restrict our modifications to decreasing v until Algorithm 1 returns enough delays. It is likely that a better adaption exists, but we leave the study of this aspect for future works.

D. Evaluation Results

In this section, we use 5-bit precision (n=32, k=31) unary bit-streams of length nk=992 and 10-bit precision Sobol-based bit-streams of length $2^{10} = 1024$, unless stated otherwise. We stop processing of bit-streams in the MUX-based and Sobol$_{\mathrm{OR}}$ designs after n^2 cycles (here, 1024 cycles). Although this introduces some truncation inaccuracy [5] in the computation, the required number of processing cycles to produce high accuracy results with these MAC designs exponentially increase with increasing the number of inputs, which is not feasible in practice. We discuss the differences in the latency of different SC MAC designs in more detail in Section VIII.

Table II compares the mean absolute error (MAE) (in percent) of the implemented MAC designs for different ranges of input values. We stop increasing the range of input values when the reference output reaches 1. The reference computes on double precision and is listed in the first row. Sobol$_{\mathtt{MUX}}$-rescaled is equivalent to Sobol$_{\mathtt{MUX}}$ but with compensation of the scaling inherent to the MUX-based adder. In a second processing step, after converting back to positional binary representation, the result of the scaled Sobol$_{\mathtt{MUX}}$ is multiplied by N to get the correct order of magnitude (Sobol$_{\mathtt{MUX}}$-rescaled = Sobol$_{\mathtt{MUX}}$ $\cdot N$). The same is true for Sobol$_{\mathtt{MUX/TFF}}$-rescaled. All listed designs perform the multiplication part of the MAC operation accurately. Their summation part, however, can cause error. Sobol$_{\mathrm{OR}}$ calculates the union of inputs and suffers from a systematic error when compared to reference sums. The MUX-based designs implement scaled MAC operations and hence have a systematic deviation to non-scaled MAC results. Random fluctuations in the select input [15] of Sobol$_{\mathtt{MUX}}$ and the fact that the MUX unit discards $N - 1$ bits (through multiplexing) each clock cycle leads to further accuracy loss. In general, Sobol$_{\mathtt{MUX}}$-rescaled has the potential to produce accurate results for all input values and number of inputs as the systematic deviation of the Sobol$_{\mathtt{MUX}}$ gets compensated when multiplying with N. However, it would take more than 1024 processing cycles to converge to the correct result.

As can be seen in Table II, the proposed Unary$_{\mathrm{OR}}$ design is the only MAC design that can compute completely accurate results. It is significantly more accurate than the traditional $\mathtt{OR}$-based approaches. When compared to Sobol$_{\mathrm{OR}}$, the error decreases between 10% to 100% depending on the range and number of input values. The proposed design achieves a minimum error decrease of 37% for the case of processing four inputs in the $[0, 0.95]$ interval ($\frac{5.1-3.2}{5.1} \times 100 \approx 37\%$) and 100% decrease when the proposed method is exact.

When comparing the Unary$_{\mathrm{OR}}$ technique to rescaled MUX-based methods the table shows two trends. The proposed Unary$_{\mathrm{OR}}$ design performs better for small input ranges, and medium input ranges with large number of inputs. For 24-input MAC the proposed method has lower error for input values in the $[0, 0.43]$ interval. The MAC error decreases by 100% for input values in the $[0, 0.29]$ interval and decreases by a minimum of $\frac{4.1-3.6}{4.1} \times 100 = 12\%$ for input values in range $[0.0.43]$. In contrast, higher error is observed for input values greater than 0.45. For 12 inputs, Unary$_{\mathrm{OR}}$ is more accurate than both MUX-based MAC designs for input values less than 0.55. For four inputs, Sobol$_{\mathtt{MUX/TFF}}$-rescaled, in most cases, computes more accurate results than Unary$_{\texttt{OR}}$ and Sobol$_{\texttt{MUX}}$-rescaled. The proposed Unary$_{\texttt{OR}}$ is more accurate than Sobol$_{\texttt{MUX}}$-rescaled for inputs values less than 0.71. Nevertheless, considering the first row of Table II, if the mean reference result is below 0.25, our proposed technique achieves either exact results or a mean absolute error close to zero.

To give an example for one table entry, a mean reference result of 0.125 with two summands ($N = 2$) requires uniformly distributed random variables in range of 0 to 0.5 as input. In that case, the inputs have an expected value of $E(x) = 0.25$. Since the maximum input for this case is 0.5, the proposed method will be exact as shown in Table II and Fig 7.

$$z = E(x_1)E(y_1) + E(x_2)E(y_2) \overset{!}{=} 0.125 \tag{93}$$
$$\rightarrow x_1, y_1, x_2, y_2 \sim \mathcal{U}(0, 0.5) \tag{94}$$
$$z = 2 \cdot 0.25^2 = 0.125 \tag{95}$$

In the last four columns of Table II, the range of inputs for $N = 2$ exceeds the $[0, 1]$ interval. The upper bound (i.e., 1) gives a maximum input mean of $E(x) = 0.5$, when the input values are uniformly distributed. The maximum mean reference of the MAC output is therefore $z = \sum_{i=1}^{N} x_i y_i = 0.5^2 + 0.5^2 = 0.5$. However, we need outputs of greater than 0.5 to evaluate the accuracy for full possible range of the results. To provide inputs with mean value greater than 0.5, we generate uniformly distributed random values in the $[0, 1.05]$, $[0, 1.14]$, $[0, 1.22]$,$[0, 1.31]$ and $[0, 1.38]$ intervals, and round the values down when the generated input is greater than 1. This way, the mean of inputs increases to values greater than 0.5 as input values close or equal to 1 occur more often. For example, when we generate 13 uniformly distributed random values within the $[0, 1.2]$ interval we get a mean of 0.6, with on average two values greater than 1.0. If we clip the input values to 1.0, the mean value becomes 0.57. The disadvantage of clipping is a deviation from equal spreading of input values. However, it allows using the same RNG in all evaluations.

Fig. 9 shows the MAE of the proposed technique for different period lengths (n) and number of summands (N) over the expected reference result (marked on the X-axis). The inputs are scaled, uniformly distributed random values equal to the numbers in Table II. As it can be seen, the period length and the number of summands have a negligible impact on the accuracy. The MAC error primarily depends on the value of the reference result. Note that, the input-output relation of MAC is linear, due to the linearity of both multiplication and addition. We recall the threshold (92) for input values and accurate computation derived in Section IV. Fig. 9 shows that the accuracy threshold for the MAC result is approximately 0.25. This can also be seen in the third column of Table II, with the mean reference MAC output listed in the first row. Beyond this exact threshold, the error increases quadratically. This matches the quadratic increase of E_{upper} derived in Section V. When the number of inputs increases, the input ranges need to be decreased to keep the mean reference result in the same interval and achieve a similar accuracy with the case of fewer inputs. We show the impact of increasing the number of inputs on accuracy in Table III, where we compare Unary$_{\texttt{OR}}$, Sobol$_{\texttt{OR}}$, Sobol$_{\texttt{MUX/TFF}}$-rescaled and Sobol$_{\texttt{MUX}}$-rescaled for different bit-stream lengths and number of inputs. For the purpose of not exceeding an output of 1 we limit the input range for $2N-2$ MAC inputs to the $[0, 0.25]$ interval (the total number of MAC inputs is $2N$). The remaining two inputs have values in the $[0, 1]$ interval. As a result, the MAC circuit needs to sum one summand with average of 0.5^2=0.25 with N-1 summands with average of 0.125^2. As can be seen in Table III, all implementations except Sobol$_{\texttt{MUX/TFF}}$-rescaled achieve a comparable MAE as in Table II. Sobol$_{\texttt{MUX/TFF}}$-rescaled produces error if the input values are significantly different ($x >> y$, or $y << x$). For example, in Fig. 8, assume that x is close to 0 and y close to 1. Since the select input is 0 most of the time, the result is $z \approx y$ rather than $z = \frac{x+y}{2}$. Unary$_{\texttt{OR}}$ halves the MAE of conventional $\texttt{OR}$-based approaches and in most cases has a higher error than Sobol$_{\texttt{MUX}}$-rescaled. However, the MUX-based approaches require adjustment of the scaling factor, whereas Unary$_{\texttt{OR}}$ already provides a non-scaled result. Our evaluations confirm that the proposed method is accurate, even with large number of inputs, as long as the input values are small enough.

TABLE III: MAE (%) comparison for different number of summands (N) and input bit-stream lengths (n). One summand is in [0,1] interval and N-1 summands are below 0.25^2.

Reference MAC Output	0.26	0.33	0.33	0.42	0.43	0.55	0.55	0.70
n	16		32		64		128	
# Inputs ($2N$)	4	12	12	24	24	40	40	60
Unary$_{\texttt{OR}}$	0.5	1.5	1.3	2.4	2.4	4.1	4.3	7.6
Sobol$_{\texttt{OR}}$	0.5	2.9	2.3	5.7	5.3	10.3	10.3	17.7
Sobol$_{\texttt{MUX/TFF}}$-rescaled	0.5	7.9	8.0	8.0	8.0	14.9	15.0	3.4
Sobol$_{\texttt{MUX}}$-rescaled	2.2	5.3	2.7	4.5	2.2	3.3	1.6	2.3

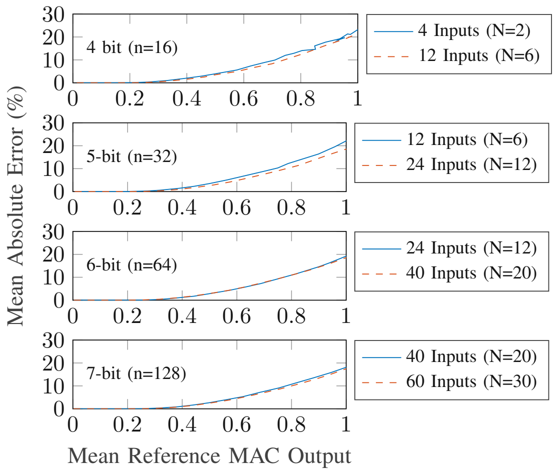

Fig. 9: The MAE (%) of Unary$_{\texttt{OR}}$ versus the mean reference output for different relative prime period lengths (n and $k = n-1$) and number of MAC inputs. The inputs of the MAC are scaled, uniformly distributed random variables.

E. A Case Study: Gray-scale

To evaluate the proposed technique in an end-application, we perform gray scaling on true color images. Gray scaling is an image processing task that requires a MAC operation for each pixel. RGB values are converted to gray-scale values by

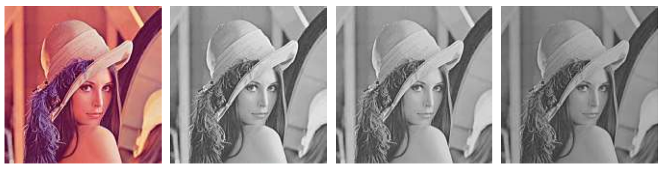

Fig. 10: (a) Original RGB image. Gray-scale using (b) reference design (c) Unary$_{\texttt{OR}}$ and (d) Sobol$_{\texttt{OR}}$

forming a weighted sum of the R, G and B components as in

$$Gray = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B. \qquad (96)$$

Gray-scale is well-suited for unipolar SC. All inputs and results are positive values in the [0,255] interval that be scaled to the [0,1] interval (as Q8 number format with eight fractional bits). The MAC circuit has six inputs ($N = 3$ summands), with three of them being constant and the other three changing with each pixel. The images are shown in Fig. 10 with sizes 512×512×3 for RGB and 512×512×1 for gray-scale. Fig. 10(a) shows the original image and Fig. 10(b) is the reference gray-scale result. The reference uses 5-bit precision inputs and exact arithmetic. Fig. 10(c) and (d) show the gray-scale results for Unary$_{\texttt{OR}}$, and Sobol$_{\texttt{OR}}$ that also use 5-bit precision inputs and therefore ignore the three least significant bits of the 8-bit color values. Fig. 10(c) has a MAE of 1.6% and Fig. 10(d) has a MAE of 10.8% when compared to the output of the reference. The image in Fig. 10(d) appears darker than Fig. 10(c) as `OR`-based MAC result values are smaller than the results of the new Unary$_{\texttt{OR}}$ (as shown in Fig. 7 too) and smaller values mean less brightness. The average value of the pixels in the reference gray-scale image is 125 (=0.49 in Q8 number format). Table II lists a MAE of 2.6% for the proposed method in the case of six summands at a mean reference MAC output of 0.45. Since gray-scale only uses three summands instead of six, we can halve this table entry 2.6/2=1.3%. A MAE of 1.3% is close to the 1.6% error of the proposed Unary$_{\texttt{OR}}$ in this case study.

Unary$_{\texttt{OR}}$ and Sobol$_{\texttt{OR}}$ require almost the same number of clock cycles per pixel. Unary$_{\texttt{OR}}$ finishes after 1012 cycles while Sobol$_{\texttt{OR}}$ stops after $n^2 = 1024$ cycles. The results in Fig. 10(c) can be computed efficiently with Unary$_{\texttt{OR}}$ and Unary$_{\texttt{OR/REG}}$. We refer the readers to Section VI-B and Section VII for an analysis of both implementations and a resource consumption comparison.

VII. Resource Comparison

In this section, we compare area and power consumption of the state-of-the art methods with different implementations of the proposed MAC technique. A SC system often consists of some SNGs, a stochastic circuit that does the actual computation and a probability estimator (for bit-stream to binary conversion). We exclude the overhead of SNGs in our evaluations as they are already discussed in the literature extensively [2], [3], [5]. We also exclude the overhead cost of compensating the scaling for the MUX-based methods (multiplying by N). Therefore, we don't differentiate between Sobol$_{\texttt{MUX}}$ and Sobol$_{\texttt{MUX}}$-rescaled (same for Sobol$_{\texttt{MUX/TFF}}$ and Sobol$_{\texttt{MUX/TFF}}$-rescaled). The resource comparison includes the stochastic circuit, a minimal control logic (receives start signals and transmits a done signal after completion) and the probability estimator unit. All flipflops are synchronous with synchronous resets. We synthesized the designs using the Synopsys Design Compiler v2018.06 with a 45nm gate library. The designs were synthesized for 100MHz frequency.

Table IV lists the area and power consumption of all six implementations, with three of them being based on the proposed MAC, for different number of inputs and input precisions. The comparison shows that the implementations without delay registers are significantly more efficient in both area and power than implementations with registers (Unary$_{\texttt{OR/REG}}$ and Unary$_{\texttt{OR/SEQ}}$). The exception is Unary$_{\texttt{OR/REG}}$ with four inputs, as it requires comparable area and power but can be implemented with fewer RNGs (see Section VI-B). Note that, Unary$_{\texttt{OR/REG}}$ is also viable for six inputs, because for four and six inputs the proposed method only requires *minor* relative delays. We refer to Section III-B for the definition of minor delays and to Fig. 3 (first tree subplots) for an example of minor delayed summands. All implementations except Unary$_{\texttt{OR/SEQ}}$ require approximately the same computation time. So, the ratio of the required energy is similar to the ratio of the required power. However, this is not true for Unary$_{\texttt{OR/SEQ}}$. HDL simulations show that the sequential implementation requires $2\times$, $7.8\times$, and $17.9\times$ more computation time than the Sobol-based methods for 4, 12 and 24 inputs, respectively. This is primary due to the sequential processing of inputs and secondary due to the stalled bit-stream generation, which is required to achieve the same accuracy as Unary$_{\texttt{OR/REG}}$ and Unary$_{\texttt{OR}}$. As a result of the longer processing time, the energy consumption is significantly higher for the Unary$_{\texttt{OR/SEQ}}$ design than for the other implementations. The resource consumption of Sobol$_{\texttt{OR}}$, Sobol$_{\texttt{MUX}}$, and Unary$_{\texttt{OR}}$ hardly changes for different accumulator sizes and bit-stream lengths. The reason is that the reported numbers does not include the cost of the SNGs, and the changes only show different counter bitwidths, additional parallel `AND` gates for multiplication, and more inputs to the `OR` gate (or the MUX unit) for summation. Sobol$_{\texttt{MUX/TFF}}$ shows more resource consumption with increasing MAC size. This is because it needs additional toggle flipflops and `XOR` gates to compute the MUX's select bit-stream (see Fig. 8). The required delay registers of Unary$_{\texttt{OR/REG}}$ increases heavily with the input count as each additional summand requires a larger delay register. In contrast, Unary$_{\texttt{OR/SEQ}}$ has a high base resource consumption which barely changes with the number of inputs as the overall feedback buffer size is similar for all input counts. Since the number of delay elements heavily depends on the bit-stream lengths, both Unary$_{\texttt{OR/SEQ}}$ and Unary$_{\texttt{OR/REG}}$, require significantly less resources for $n = 16$ than for $n = 32$.

Between the three implementations of the proposed technique (i.e., Unary$_{\texttt{OR}}$, Unary$_{\texttt{OR/SEQ}}$, Unary$_{\texttt{OR/REG}}$), Unary$_{\texttt{OR}}$ is selected if the number of MAC inputs is greater than six and counters for bit-stream generation can be shared between multiple parallel MAC units. If counters can not be shared, or the number of MAC inputs is less than or equal six, Unary$_{\texttt{OR/REG}}$ could be more efficient, when taking bit-stream generation into account. Unary$_{\texttt{OR/SEQ}}$ uses the MAC architec-

TABLE IV: Area (μm^2) and Power consumption (μW) for three state-of-the-art SC MAC designs and for three implementations of the proposed MAC technique.

		4 Inputs		12 Inputs		24 Inputs		40 Inputs		60 Inputs	
		Area	Power	Area	Power	Area	Power	Area	Power	Area	Power
n=32	Sobol$_{\text{OR}}$	413	1.24	429	1.67	450	2.16	487	2.74	521	3.52
	Sobol$_{\text{MUX/TFF}}$	434	1.52	525	3.15	660	5.55	840	8.77	1069	12.7
	Sobol$_{\text{MUX}}$	428	1.46	461	1.85	528	2.82	574	3.43	643	4.65
	Unary$_{\text{OR}}$	451	1.44	520	1.75	542	1.71	664	2.07	743	2.01
	Unary$_{\text{OR/REG}}$	596	1.48	11026	15.5	33755	47.9	62586	89.5	103730	0.147
	Unary$_{\text{OR/SEQ}}$	10891	14.7	14355	19.6	16435	22.6	17154	23.4	17878	24.6
n=16	Sobol$_{\text{OR}}$	385	1.21	389	1.50	403	2.32	426	2.75	481	3.92
	Sobol$_{\text{MUX/TFF}}$	405	1.62	495	4.11	585	4.93	765	8.16	993	12.4
	Sobol$_{\text{MUX}}$	389	1.20	416	1.64	439	2.25	506	3.26	581	4.59
	Unary$_{\text{OR}}$	444	1.40	502	1.76	502	1.72	557	1.94	640	2.49
	Unary$_{\text{OR/REG}}$	490	1.39	3246	5.26	9114	14.1	16606	27.9	16470	28.9
	Unary$_{\text{OR/SEQ}}$	3025	4.09	4012	6.35	4465	5.31	4717	6.87	4572	6.70

ture in Fig. 2(a) and is not efficient enough to compete with the parallel architecture in Fig. 2(b), as the cost for large shift registers outweighs potential savings. The proposed Unary$_{\text{OR}}$ design requires, on average, 21% more area than Sobol$_{\text{OR}}$ due to the overhead of stalling the SNGs to make relative delays between bit-streams. Further, Unary$_{\text{OR}}$ and Sobol$_{\text{OR}}$ require similar power and energy due to similar processing time. The main difference is their distinct bit-stream generation. Both Unary$_{\text{OR}}$ and Sobol$_{\text{OR}}$ require N SNGs, but Unary$_{\text{OR}}$ requires counters as RNG while Sobol$_{\text{OR}}$ requires costly Sobol sequence generators [20]. Without the costs for SNGs, Sobol$_{\text{MUX}}$ consumes slightly less and Sobol$_{\text{MUX/TFF}}$ more resources than Unary$_{\text{OR}}$. However, we emphasize that MUX-based circuits compute scaled results. Rescaling by N is not possible in unipolar SC and the resource consumption of the weighted binary multiplier is not included in this analysis.

VIII. Further Discussions

In this section, we discuss three properties of the proposed technique in more detail. First, we examine the implications of N (the number of possible summands) being pronic. Second, we discuss the consequences of requiring bit-streams with relative prime period lengths as inputs. Finally, we take a closer look at the convergence behavior of the proposed technique compared to the state-of-the-art methods.

1) In Section IV, we derived that the maximum allowed number of summands is a pronic number ($N = 2, 6, 12, 20, 30 \ldots$). The maximum allowed number of inputs for the MAC unit is twice of that ($2N = 4, 12, 24, 40, 60 \ldots$). The number of inputs is often given by the application. The proposed method also works for the number of inputs that are not pronic. In that case, it is possible to use parts of the delays from Algorithm 1 and leave the other unused. Then, the accuracy threshold (92) and error (Table II) is similar as if all possible inputs are used. Compared to prior methods, the proposed design performs best when the number of inputs equals the maximum allowed from (3).

2) The proposed MAC technique of this work guarantees uncorrelated inputs by using relatively prime bit-stream lengths. The input bit-streams to the AND gates use relative prime period lengths of n and k ($n = k + 1$). Thus, one input value needs to be converted to a unary bit-stream with a slightly higher resolution than the other one. This difference in the representation introduces a systematic quantization inaccuracy during bit-stream generation particularly for small values of n and k. For example, the bit-stream representation for 0.5 with period lengths $n = 8$ and $k = 7$ is 11110000 and 1111000, respectively. When converting back to positional binary the values are $\frac{v}{n} = \frac{4}{8} = 0.5$ and $\frac{v}{k} = \frac{4}{7} = 0.5714$. We claim that the proposed MAC design is deterministic and accurate because its inaccuracy is predictable, systematic and occurs at the bit-stream generation and not in the computation circuit.

The maximum number of 1s in a stochastic bit-stream is limited by the length of bit-stream. A relative delay between input bit-streams increases the length of the output bit-stream. In the proposed MAC design, the output bit-stream has up to $n \cdot k$ (the LCM of input lengths) plus $N_{major} \cdot vn + N_{minor} \cdot v$ (the maximum delay from Algorithm 1) bits. Assuming that the maximum value (i.e., 1.0) is represented by $n \cdot k$ bits of 1s, the expanded bit-stream result may represent a value greater than 1, which needs to be considered when converting the output back to the conventional weighted binary representation.

3) The number of processing cycles (i.e., latency) is different for different stochastic MAC designs. The first stage of MAC operation is multiplication which is similarly implemented in all MAC designs using standard AND gates. Different MAC designs, however, are different in the second stage of MAC operation which accumulates the multiplication results. The inputs of each AND gate are two uncorrelated bit-streams. For highest accuracy the inputs are 2^{2B} bit long when multiplying two B-bit precision input values [5]. In Section VI, we choose $B = 5$, so the multiplication latency is $2^{10} = 1024$ cycles (assuming each bit of the input bit-streams are processed in one cycle) for the Sobol-based designs and $n \cdot k = 992$ cycles for the proposed design.

In Sobol$_{\text{OR}}$, the OR-based addition converges to the union of its inputs after $2^{N \cdot 2B} = 2^{N \cdot 10}$ cycles where N is the number of summands. MUX-based MAC converges to the scaled addition of the summands after $2^{2 \times 2B} = 2^{2 \times 10}$ cycles because the select input of the MUX must be uncorrelated to the outputs of multiplications. The summation stage of the proposed Unary$_{\text{OR}}$ MAC technique does not introduce any additional latency besides the extra cycles from the relative delays. Unary$_{\text{OR}}$ requires exactly nk, the LCM of inputs, plus $N_{major} \cdot vn + N_{minor} \cdot v$ cycles, the maximum delay from Algorithm 1 to reach its maximum accuracy (the maximum latency is less than $2nk$ cycles). The MAE results of the MUX-based methods in Section VI improve with longer processing times. In contrast, running the proposed MAC technique for the exact number of cycles is important, because the accuracy decreases if the computation is stopped too early or too late.

IX. Conclusions

In this work, we proposed a novel SC MAC technique based on deterministic unary bit-streams. We showed that the proposed design can compute completely accurate non-scaled MAC and calculates overall more accurate results than the OR-based MAC design fed with Sobol-based LD bit-stremas. It also achieves lower error compared to the rescaled MUX-based MAC designs, except for the case of processing a small number of large inputs where the inherent scaling of MUX units is beneficial. We provided practical implementations that show the proposed technique is suitable for applications with low input counts that require exact computation as well as large accumulator sizes that can tolerate small errors. Modification of the proposed algorithm and using additional logic gates are potential solutions to further increase the threshold for exact MAC operation and to enhance the accuracy for large input values. Such solutions however require further investigation and are part of our future work.

Acknowledgements

This work was supported in part by the Louisiana Board of Regents Support Fund no. LEQSF(2020-23)-RD-A-26 and National Science Foundation grant no. 2019511.

References

[1] B. Gaines. Stochastic computing systems. In *Advances in Information Systems Science*, pp. 37–172. Springer US, 1969.
[2] A. Alaghi *et al.* The Promise and Challenge of Stochastic Computing. *IEEE Trans. on Computer-Aided Design of Integ. Circuits and Systems*, 37(8):1515–1531, Aug 2018.
[3] W. Qian *et al.* An architecture for fault-tolerant computation with stochastic logic. *IEEE Transactions on Computers*, 60(1):93–105, Jan 2011.
[4] W. Poppelbaum *et al.* Unary processing. In *Advances in Computers*, volume 26, pp. 47 – 92. Elsevier, 1987.
[5] M. H. Najafi *et al.* Performing Stochastic Computation Deterministically. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 27(12):2925–2938, 2019.
[6] M. H. Najafi *et al.* Low-Cost Sorting Network Circuits Using Unary Processing. *IEEE Trans. on VLSI Systems*, 26(8):1471–1480, Aug 2018.
[7] M. H. Najafi *et al.* Time-Encoded Values for Highly Efficient Stochastic Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(5):1644–1657, May 2017.
[8] D. Jenson and M. Riedel. A deterministic approach to stochastic computation. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2016.
[9] S. R. Faraji and K. Bazargan. Hybrid Binary-Unary Hardware Accelerator. *IEEE Trans. on Computers*, 69(9):1308–1319, 2020.
[10] S. A. Faraji *et al.* HBUNN - Hybrid Binary-Unary Neural Network: Realizing a Complete CNN on an FPGA. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*, pp. 156–163, 2019.
[11] S. Mohajer *et al.* Parallel Unary Computing Based on Function Derivatives. *ACM Trans. Reconfigurable Technol. Syst.*, 14(1), October 2020.
[12] A. Alaghi and J. Hayes. Exploiting correlation in stochastic circuit design. In *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, pp. 39–46, Oct 2013.
[13] V. T. Lee *et al.* Correlation manipulating circuits for stochastic computing. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1417–1422, 2018.
[14] V. T. Lee *et al.* Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pp. 13–18, March 2017.
[15] P. Ting and J. P. Hayes. Eliminating a hidden error source in stochastic circuits. In *2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6, 2017.
[16] B. Li *et al.* Using stochastic computing to reduce the hardware requirements for a restricted boltzmann machine classifier. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, pp. 36–41, New York, NY, USA, 2016. ACM.
[17] B. Yuan and Y. Wang. High-accuracy fir filter design using stochastic computing. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 128–133, 2016.
[18] P. Ting and J. P. Hayes. Stochastic logic realization of matrix operations. In *2014 17th Euromicro Conference on Digital System Design*, pp. 356–364, 2014.
[19] H. Sim and J. Lee. Cost-effective stochastic mac circuits for deep neural networks. *Neural Networks*, 117:152–162, 2019.
[20] S. Liu and J. Han. Energy efficient stochastic computing with sobol sequences. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pp. 650–653, 2017.
[21] Wong, Ming Ming *et al.* A new stochastic inner product core design for digital fir filters. *MATEC Web Conf.*, 125:05006, 2017.
[22] Y. Chang and K. K. Parhi. Architectures for digital filters using stochastic computing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2697–2701, 2013.
[23] J. A. Dickson *et al.* Stochastic arithmetic implementations of neural networks with in situ learning. In *IEEE International Conference on Neural Networks*, pp. 711–716 vol.2, 1993.
[24] M. H. Najafi *et al.* Deterministic Methods for Stochastic Computing using Low-Discrepancy Sequences. In *Proceedings of the 37th International Conference on Computer-Aided Design*, ICCAD '18, 2018.
[25] S. Liu and J. Han. Toward Energy-Efficient Stochastic Circuits Using Parallel Sobol Sequences. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(7):1326–1339, July 2018.
[26] B. Moons and M. Verhelst. Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 4(4):475–486, Dec 2014.

Peter Schober received his B.Sc. degree in Electronics and Information Technology from Johannes Keppler University, Linz, Austria, in 2017. He is currently a Master's student in Embedded Systems at the TU Wien (formerly known as Vienna University of Technology as well), Vienna, Austria. From September to December 2019, he was a visiting research scholar at the School of Computing and Informatics, University of Louisiana, LA, USA.

M. Hassan Najafi received the B.Sc. degree in Computer Engineering from the University of Isfahan, Iran, the M.Sc. degree in Computer Architecture from the University of Tehran, Iran, and the Ph.D. degree in Electrical Engineering from the University of Minnesota, Twin Cities, USA, in 2011, 2014, and 2018, respectively. He is currently an Assistant Professor with the School of Computing and Informatics, University of Louisiana, LA, USA. His research interests include stochastic and approximate computing, unary processing, in-memory computing, and machine-learning. In recognition of his research, he received the 2018 EDAA Outstanding Dissertation Award, the Doctoral Dissertation Fellowship from the University of Minnesota, and the Best Paper Award at the ICCD'17.

Nima TaheriNejad (S'08-M'15) received his Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, Canada, in 2015. He is currently an assistant professor at the TU Wien, Vienna, Austria, where his areas of work include cyber-physical embedded systems, computer architecture, computational self-awareness, in-memory computing, systems on chip, and health-care. He has published two books and more than 60 peer-reviewed articles. He has also served as a reviewer, an editor, an organizer, and the chair for various journals, conferences, and workshops. Dr. Taherinejad has received several awards and scholarships from universities and conferences he has attended.

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Copyright Statement

I, Peter Schober, hereby declare that this thesis is my own original work and, to the best of my knowledge and belief, it does not:

- Breach copyright or other intellectual property rights of a third party.
- Contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
- Contain material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.
- Contain substantial portions of third party copyright material, including but not limited to charts, diagrams, graphs, photographs or maps, or in instances where it does, I have obtained permission to use such material and allow it to be made accessible worldwide via the Internet.

Signature: [illegible]

Vienna, Austria, January 2022 Peter Schober