Sound Source Localization using Stochastic Computing

Peter Schober[+*], Seyedeh Newsha Estiri[*], Sercan Aygun[*], Nima TaheriNejad[+], and M. Hassan Najafi[*]
[+]Technische Universität Wien, Vienna, Austria, [*]University of Louisiana at Lafayette, Louisiana, USA
Corresponding Author: najafi@louisiana.edu

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 smaller computer chips. SC has been primarily used in scientific research, not in practical applications. Digital sound source localization (SSL) is a useful signal processing technique that locates speakers using multiple microphones in cell phones, laptops, and other voice-controlled devices. SC has not been integrated into SSL in practice or theory. In this work, for the first time to the best of our knowledge, we implement an SSL algorithm in the stochastic domain and develop a functional SC-based sound source localizer. The developed design can replace the conventional design of the algorithm. The practical part of this work shows that the proposed stochastic circuit does 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 baseline 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 such as radar ranging, wireless location, sonar direction finding, beamforming, and sensor calibration.

1 INTRODUCTION

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 mobile phones 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 [1].

Sound source localization (SSL) is a real-time application, starting with capturing audio data and ending in location estimations. Reducing sound source localizer's area and power consumption is in high demand. 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. Stochastic computing (SC) [2, 3] is a promising computing paradigm that can provide a lower power consumption, a lower hardware area footprint, and a higher tolerance to soft errors. SC promises high power efficiency that can contribute to longer battery lives. Developing a sound source localizer using SC could save considerable hardware costs.

In SC, numbers are encoded into stochastic bit-streams of 0s and 1s. SC is possible in both time-continuous and time-discrete domains [4]. However, some operations, such as delaying and storing stochastic sequences, are simpler in the time-discrete domain. The concept of coding a value into a stochastic number (SN) (aka bit-stream) is fundamentally different from the conventional binary-radix data encoding. Stochastic sequences have voltage level 1 (high) with probability ρ and level 0 (low) with probability $1-\rho$. In other words, a bit within a stochastic bit-stream is 1 with probability $P(S_i = 1) = \rho$ and is 0 with probability $1-\rho$. For example, a stochastic sequence S with 20% 1s and 80% 0s is an SN with a probability value $\rho = 0.2$. Unsigned and signed numbers are supported in SC with unipolar and bipolar SN formats, respectively [2]. In the unipolar format, each bit of the bit-stream is 1 with probability ρ, and in the bipolar format is 1 with probability $(\rho+1)/2$. We use the notation S_i for the ith bit in bit-stream S with length N ($0 \le i \le N-1$). SC can perform arithmetic operations with simple logic gates. For example, the multiplication operation can be performed using a single `AND` gate, which provides a significantly lower hardware cost than the conventional binary multiplication.

This work targets the computationally intensive parts of SSL 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. This work focuses on time-discrete SC, although a time-continuous variant of the sound source localizer is also feasible. To the best of our knowledge, this is the first work that applies SC to SSL. Our design takes advantage of the recently introduced deterministic approaches of SC [4–7] which provide lower latency and higher accuracy compared to the conventional random SC. In summary, the main contributions of this work are as follows:

- Developing the first SC-based design for SSL, a promising alternative to its costly conventional implementation.
- Near-sensor processing of sound signals using time-encoded pulse-width-modulated (PWM) data; Our signal processing system avoids using conventional analog-to-digital converters (ADCs).
- Designing an accurate SC-based multiply-and-accumulate (MAC) unit used in the SSL design based on time-delay estimation (TDE).
- Practical hands-on implementation of a new, efficient sound source localizer with SC in both digital and analog domains.

2 BACKGROUND

2.1 Sound Source Localization (SSL)

SSL algorithms exist in both the frequency and time domains. Search-based SSL algorithms in the frequency domain are superior in locating multiple sources simultaneously but are more computationally intensive [8, 9]. This work uses a time-domain algorithm without Fourier transforms and assumes only one active source. An SSL system consists of three subsystems. The pre-processing subsystem usually segments, filters, and weights the audio stream. It splits the stream into short data blocks as 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 accuracy of estimations [1]. Figure 1 summarizes the overall flow of a sound source localizer.

The first subsystem of a sound source localizer prepares raw microphone signals for the SSL algorithm. The pre-processing usually

ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA © 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9217-4/22/10…
https://doi.org/10.1145/3508352.3549373

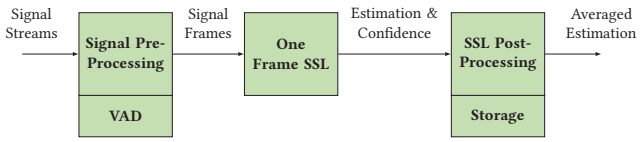

Figure 1: Block Diagram of a Sound Source Localizer.

includes low-pass or band-pass filtering, framing, and VAD. The theory of SSL can be divided into algorithms that locate sources in either one or two processing steps. The one-step methods are based on the steered response power. Algorithms based on the steered response power require a Fourier transform in the pre-processing subsystem because they compute the source location in the frequency domain. This work focuses on the two-step methods that rely on TDEs for each microphone pair [1, 9]. The post-processor is the final part 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.

2.2 SSL Based on Time Delay Estimation (TDE)

Algorithms based on TDE use the propagation speed of sound waves to localize sources [10]. 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 $N_{pairs} = (M(M-1))/2$ 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 direction-of-arrival (DOA):

$$\tau_d = \frac{d}{v} \sin \varphi, \tag{1}$$

where d is the distance between the microphones and v is the velocity of sound waves.

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

$$c[n] = (\mathbf{x_1} \star \mathbf{x_2})[n] \sum_{i=-\infty}^{\infty} x_1^*[i] x_2[i+n] \tag{2}$$

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

$$\tau_d = \frac{i_{max}}{f_s} \tag{3}$$

$$\varphi = \arcsin \frac{\tau_d \cdot v}{d} = \arcsin \frac{i_{max} \cdot v}{f_s \cdot d} \tag{4}$$

Equation (4) results from 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 [11] or search-based. For search-based algorithms, the final location estimation is the position with the minimum root-mean-square error of the calculated and measured TDE. For a detailed analysis, the readers are referred to [1, 8, 11]. The resolution of CC is determined by the distance between the microphones (d) and the sampling frequency (f_s). For example, a distance $d = 0.066$ m between two microphones and a sample rate of $f_s = 15.6$ kHz results in $2 \times N_{MaxLag} + 1 = 7$ different possible time delays:

$$\tau_{d,max} = \frac{d}{v} = 192\ \mu sec \tag{5}$$

$$N_{MaxLag} = \tau_{d,max} * f_s = 3 \tag{6}$$

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 $\tau_d = i_{max}/f_s = 192\ \mu$sec, whereas a maxima at $c[i_{max} = N_{MaxLag} - 1]$ means a time delay of 128 μsec. The resolution is limited to 64 μsec. 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 $c[i_{max}]$.

This section underscored 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 then forwarded to the SSL post-processing block.

2.3 Deterministic Stochastic Computing

True random and pseudo-random numbers have been used for generating stochastic bit-streams since the introduction of SC [2, 12]. The target probability is compared with N random numbers in N clock cycles to generate a bit-stream of N bits. A 1 is produced in each comparison if the random number is less than the target value. Bit-streams generated with these methods suffer from random fluctuations error. Deterministic methods of SC [6, 7] have been recently introduced to remove this source of error and generate bit-streams *accurately*. SC using these methods can produce completely accurate results. Besides quantization errors, processing bit-streams with *specific lengths* is the only constraint to produce completely accurate results with these deterministic methods. These methods operate on the same SC constructs but use special stochastic number generators (SNGs) to generate deterministic bit-streams with the desired distribution. Low-discrepancy (LD) and unary bit-streams are two common variants of deterministic bit-streams [4–6]. The so-called *unary bit-streams* have the 1s grouped at the beginning or end of the bit-stream. Unary bit-streams are generated by replacing the random number generator (RNG) with an up- (or down-) counter [13]. 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 *PWM signal*. As shown in Figure 2, a PWM signal is defined by a duty cycle (D) and a period or frequency ($f = 1/period$). The duty cycle is the fraction of time the signal is high and is equivalent to the probability of observing 1s in the time-discrete domain [14].

Correlation between inputs can change the functionality of logic circuits in SC. An `AND` gate works as a multiplier if connected with *uncorrelated* or *independent* inputs [15, 16]. However, when connected with *correlated* inputs, it performs a minimum value operation [14]. When *independent* inputs are needed, unary bit-streams

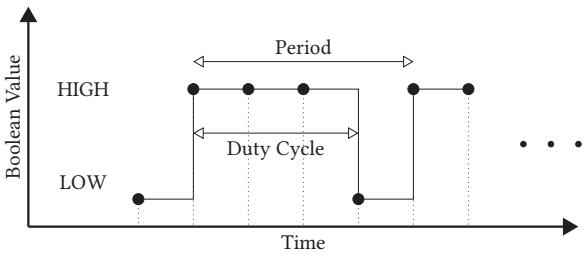

Figure 2: Encoding in time-domain with PWM signals. Repeated unary bit-streams are equal to sampled PWM signals.

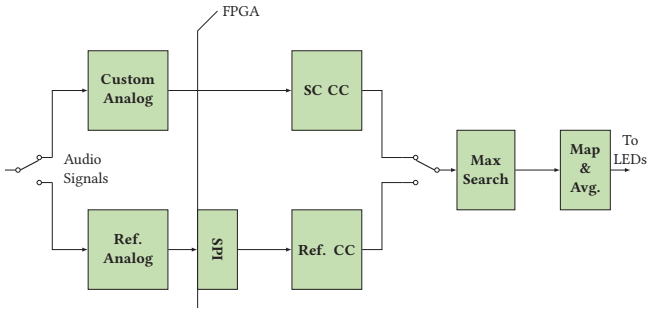

Figure 3: The implementation overview. Block diagram of the signal processing chain.

(or PWM signals) with relatively prime lengths (periods) are generated [4]. In the example below, bit-stream A is built from four times repeating unary bit-stream 100 (period=3) and bit-stream B from three times repeating unary bit-stream 1110 (period=4). The periods of the unary bit-streams are relatively prime ($GCD(3,4) = 1$). The unary bit-streams are repeated to generate bit-streams A and B with a length equal to the *least common multiple* (LCM) of both periods ($LCM(4,3) = 12$). Bit-streams are bit-wise ANDed to perform multiplication operation. Selecting relatively prime periods satisfies the independence requirement of the multiplication operation, and producing bit-streams with the LCM of the periods guarantees the accuracy of the operation. We can see accurate output (here, 3/12) is produced from bit-wise ANDing the two bit-streams.

$$A = \frac{1}{3} = 100\,100\,100\,100$$
$$B = \frac{3}{4} = 1110\,1110\,1110$$
$$A \times B = \frac{1}{3} \times \frac{3}{4} = \frac{3}{12} = 1000\,0010\,0100 \quad (7)$$

3 PROPOSED SC-BASED SSL DESIGN

We design a simple representative sound source localizer based on the SSL architecture flow in Figure 1. The design uses two microphones, analog signal processing, digital TDE with CC, an average filter for SSL post-processing, and light-emitting diodes (LEDs) for outputs. Figure 3 shows the block diagram of the implemented sound source localizer. The signal processing starts with a custom analog processing that generates PWM signals from the two input microphones. Those signals are sampled by a field-programmable gate array (FPGA) and converted to periodic unary bit-streams. We use a serial peripheral interface (SPI)-controlled ADC for reference as an alternative path. The remaining signal processing is performed on 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. After CC, the subsequent computation block searches for the maximum $c[i_{max}]$ in the centered K values of the CC, as $[i_{max}]$ is proportional to the time delay. 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 that point towards the direction of the source location.

TDE is a part of any two-step SSL algorithm. In the literature, TDE is also called the time difference of arrival estimation, time of arrival estimation, or time of flight estimation. 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 MAC operations. Using SC for implementing these operations can lead to significant savings in the hardware resources, particularly in implementing the multiplication operations [17]. In our SC design, we use *periodic unary bit-streams* for the following reasons: (1) Generating random bit-streams is costly [3], and a unary bit-stream generator reduces the bit-stream generation cost. (2) Using analog-to-time converters (PWM signal generators) instead of analog voltage to digital binary converters followed by digital bit-stream generators could save further hardware resources. This is, in particular, appealing for near-sensor processing [18].

4 ANALOG IMPLEMENTATION

In what follows, we describe the analog circuitry we built to handle the inputs and outputs of the sound source localizer. 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.

4.1 Analog Board

Figure 4 shows the *analog board*. The board requires a 5V direct current (DC) power supply. It 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 the SC circuit. In the board shown in Figure 4, the signals flow from left to right with the input audio jack on the left side and the Peripheremodule (PMOD) interface (Digilent Inc.) to the FPGA on the right side. The PMOD interface combines both outputs into a single cable. The signal processing chain for the left and right audio channels is on the top and bottom of the *analog board*. In what follows, we describe different components of the board:

(1) The audio input to the *analog board* connects with a 3.5 mm stereo audio jack on the left side. We use two microphone signals and a stereo sound card.
(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 4.) The circuit uses a Texas Instrument OPA322 OPAMP. 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 $k\Omega$ to 1000 $k\Omega$. 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 circuits.
(3) LTC1098 is an 8-bit ADC with two input channels (marked with blue in Figure 4.) LTC1098 offers a channel selection through software, is used for both (amplified) microphone signals, and connects to the FPGA over an SPI. with 14 bits. 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.

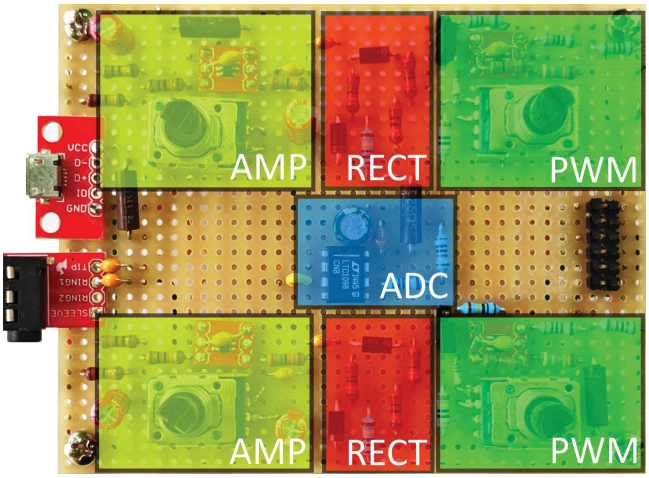

Figure 4: *Analog board* designed for two audio signals. The amplifier (AMP), half-wave rectifier (RECT), PWM generation, and ADC circuits are marked yellow, red, green and blue, respectively.

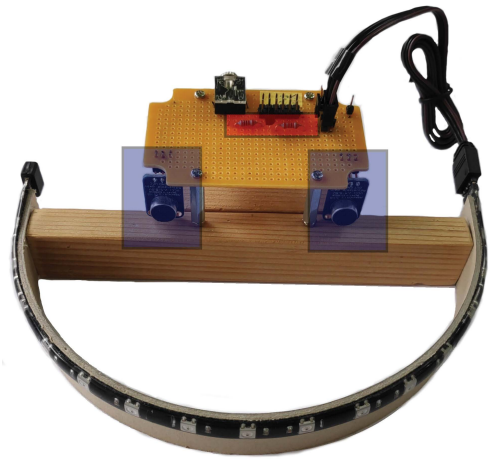

Figure 5: Input/output board with two microphones as input sensors and 15 LEDs as outputs. The microphones are soldered to ADAFRUIT MAX4466 (marked blue.) The level shift circuit for interfacing with the LED strip is marked with red.

(4) The PWM circuits receive the inputs from the half-wave rectifiers and are built with Analog Devices LTC6992 microchips (marked green in Figure 4.) The voltage level at the input (INP) pin of LTC6992 controls the duty cycle of the generated PWM signal with a non-linear transfer function. The duty cycle is $D = 0\%$ for input voltages in the 0 V to 0.1 V interval and is $D = 100\%$ for input voltages in the 0.9V to 1V interval. The two PWM signals are sampled with a clock frequency of 3.744 MHz at the input register of the FPGA, which results in bit-streams with relatively prime periods of $n = 16$ and $k = 15$:

$$n = \frac{f_{clk}}{f_{PWM,1}} = \frac{3.744 \cdot 10^6}{234 \cdot 10^3} = 16 \tag{8}$$

$$k = \frac{f_{clk}}{f_{PWM,2}} = \frac{3.744 \cdot 10^6}{249.6 \cdot 10^3} = 15. \tag{9}$$

For accurate stochastic multiplication, we need 15 periods of $f_{PWM,1}$ and 16 periods of $f_{PWM,2}$ which brings us to the equivalent audio sampling frequency:

$$f_s = \frac{f_{PWM,1}}{15} = \frac{f_{PWM,2}}{16} = 15.6\ kHz. \tag{10}$$

(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 are reserved for signals, two connected to the ground, and two for power. The 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.

4.2 Input/Output Board

Figure 5 shows the *input/output* board. The board connects to the *analog board* via a stereo cable and 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 at a 90° angle. The ADAFRUIT MAX4466 is a fully assembled printed circuit board with an electret microphone and an amplifier circuit. Both ADAFRUIT MAX4466 are marked blue in Figure 5. The outputs of the ADAFRUIT MAX4466 are single-ended microphone signals and forwarded to the *analog board* through a 3.5 mm jack located at the top of the board shown in Figure 5. Second, the *input/output* board visualizes the SSL output through 15 individually controllable LEDs (SK6812). Figure 5 shows a LED strip mounted on a semicircle at the bottom. 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 ±90° axis. The LED strip expects a 5 V control signal with a threshold greater than the output of the FPGA (HIGH≥3.4V). Thus, the *input/output* board has a level shift circuit (marked with red in Figure 5.) The unidirectional level-shift circuit consists of a single bipolar junction transistor and two resistors at the base and collector. This circuit only works for high impedance loads at the output pin and drivers that can sink enough current at the input pin.

5 DIGITAL IMPLEMENTATION

This section explains the digital processing blocks of the sound source localizer (right side of the vertical line, labeled "FPGA", in Figure 3). We start with the CC function and continue with the remaining digital processing blocks. The focus is on the SSL-based CC because the other processing blocks are implemented with traditional binary arithmetic. We must 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 $\Delta t = \frac{1}{f_s} \times N_{frame} = 8.2$ ms. Subsequently, we 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:

$$c'[n] = \sum_{i=0}^{N_{frame}-1} x_1[i]x_2[(i+n)_{\bmod N_{frame}}]. \tag{11}$$

Only the CC values close to $c'[0]$ are relevant for TDE. The distance between the microphones of the *input/output* board is $d = 0.066$ m, and the sampling rate is $f_s = 15.6$ kHz. Therefore, only $N_{MaxLag} = 3$ (see Equation (6)) values to the right and left of $c'[0]$ are of interest for TDE. We can efficiently calculate the seven

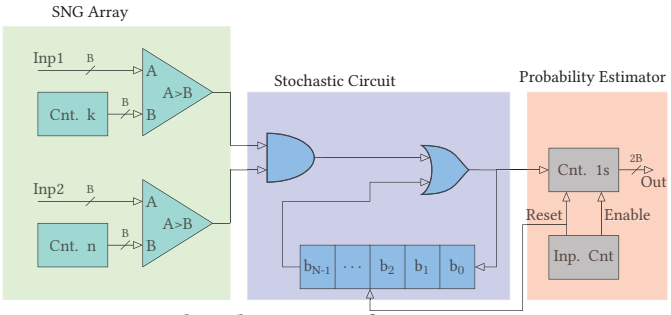

Figure 6: OR-based SC Design for MAC operations.

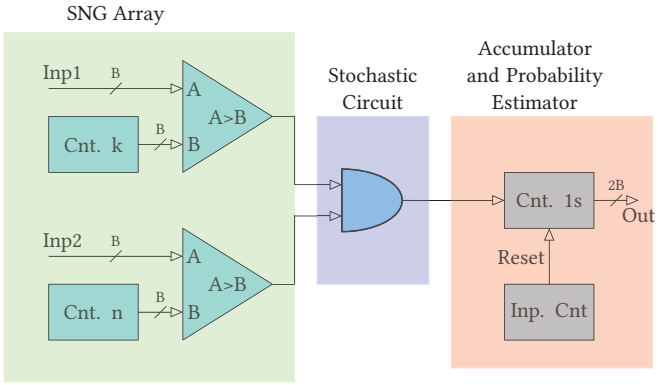

Figure 7: Counter-based SC Design for MAC operations.

($K = 2 \times N_{MaxLag} + 1 = 7$) centered values of the CC function as a set of MAC operations. We 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:

$$c'[i] = \begin{cases} c'[-3] & \leftarrow c'[-3] + x_2[n]x_1[n-3] \\ c'[-2] & \leftarrow c'[-2] + x_2[n-1]x_1[n-3] \\ c'[-1] & \leftarrow c'[-1] + x_2[n-2]x_1[n-3] \\ c'[0] & \leftarrow c'[0] + x_2[j]x_1[j] \\ c'[1] & \leftarrow c'[1] + x_2[n-3]x_1[n-2] \\ c'[2] & \leftarrow c'[2] + x_2[n-3]x_1[n-1] \\ c'[3] & \leftarrow c'[3] + x_2[n-3]x_1[n] \\ 0 & \text{if i<-3 or i>3} \end{cases} \quad (12)$$

with $j \in \{n-3, n-2, n-1, n\}$. We can use any pair of audio samples ($x_1[j]x_2[j]$) to compute $c'[0]$. The accumulator is set to zero after N_{frame} samples just before the next frame begins. The two architectures corresponding to Equation (12) and Equation (11) for sequential and parallel MAC operation compute the centered value $c'[0]$ of the CC because there is no relative shift between microphone signals $\mathbf{x_1}$ and $\mathbf{x_2}$.

In what follows, we present two SC designs for MAC operation. The first design is based on the MAC technique proposed by Schober et al. [17]. This design uses AND gates for multiplication and OR gates for addition. The AND gates compute accurate multiplication by processing bit-streams with relatively prime lengths of n and k ($k = n-1$)[1]. 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-streams and the results of AND gates have $N = LCM(nk) = 240$ bits, so the precision of the products is approximately $2B = 8$. [2] The OR gates require negatively correlated bit-streams to function as saturating adders. By introducing a relative delay between the products, we shift the 1s of the summand to a section of the accumulator bit-stream with 0s only. This guarantees a negative correlation between input bit-streams and performs saturate addition.

Combining the stochastic MAC circuit with an SNG array and a probability estimator gives the architecture shown in Figure 6. The two counters (marked with green and labeled with Cnt. n and Cnt. k) repeatedly count to n and $k = n-1$ to generate bit-streams with relatively prime periods. The bit-streams are multiplied using an AND gate. The AND gate is the first component of the stochastic circuit and forwards the product $\mathbf{x_1x_2}$ to the OR gate. The OR gate computes the bitwise disjunction of the most significant bit (MSB) of the shift register (accumulator) and the current bit of the product. The result is then stored back to the shift register's least significant bit (LSB). The OR gate's result has an error if both inputs are 1s, which we refer to as an *error due to overlap*. Schober et al. [17] 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. We know that at least half of the input values are zero due to the half-wave rectifier. Hence, we use an approximate variant of the technique with a shift register of length:

$$N_{\text{SR}} = nk + \text{ceil}(\frac{nk}{N_{frame} - 1}) \cdot (N_{frame} - 1) = 494. \quad (13)$$

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 the logical disjunction of the first and the third summand, with a relative shift between them. The final block in Figure 6 is the *probability estimator* in the form of a simple up-counting counter. The *probability estimator* is enabled by a separate counter (*Inp. Cnt*) that sets the *Enable* signal N_{SR} clock cycles before the reset and occurs every $nk \times N_{frame}$ cycles. After each frame, the *Reset* signal resets the *probability estimator* and the accumulator bit-stream. For the rest of the paper, we refer to the MAC design in Figure 6 as the *OR-based design*.

The second MAC design is shown in Figure 7. It performs the multiplication operations in the stochastic domain but the accumulation part in the conventional binary domain. The probability estimator directly counts all 1s from the AND gate instead of first accumulating with OR. The results are accurate because both the multiplication and the accumulation are accurate. We will refer to this MAC design as the *counter-based design*.

5.1 Stochastic Circuit for Cross-Correlation

Figure 8 and Figure 9 show the block diagram of the CC with the counter-based MAC for digital bit-streams and analog PWM signals, respectively. Channel One and Two (CH1 and CH2) in Figure 8 are weighted binary numbers and in Figure 9 are PWM signals from LTC6992. 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 LTC6992. For the simple approach, we continuously sample with f_{clk} and get $n = 16 (k = 15)$ bits per period of the PWM signal. During one audio sample ($\frac{1}{f_s}$), the PWM signal has $k(n)$ periods (see Equation (10)), which results in $nk = 240$ bits of data per microphone for each audio sample. The second approach 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

[1] Note that $k = n - 1$ is relatively prime for $n \geq 3$.
[2] Bit-streams with 256 bits would exactly have 8-bit precision.

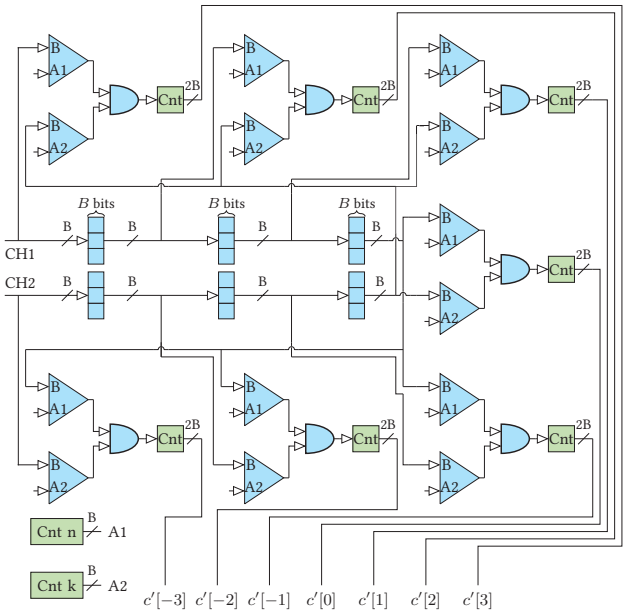

Figure 8: Block diagram of the CC with digital bit-stream generation.

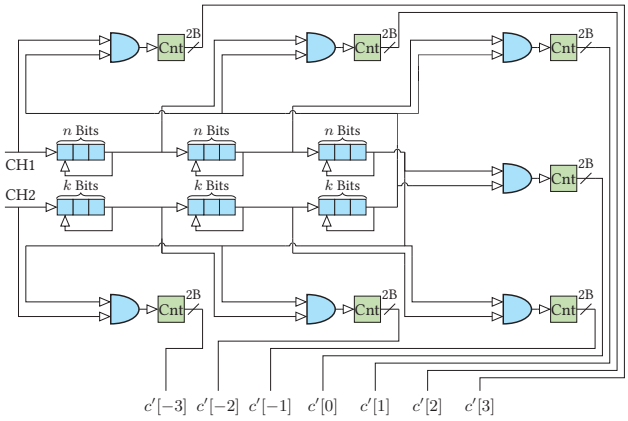

Figure 9: Block diagram of the CC with PWM signals as inputs (LTC6992).

Figure 9, we use arrows that point backward from the end to the start of the shift registers to indicate that we either do shift-through or roll operations. Both approaches provide the same bit-streams to the stochastic circuit if we assume the duty cycle of the PWM signals is constant within one audio sample.

5.2 Weighted Binary Implementation

We also 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. The multiplier is exact, with an output bit-width twice as large as the input bit-width. We note that the precision of the products is comparable to that of the output bit-stream from the AND gate with $N = nk$ bits. Choosing the accumulator bit-width is a trade-off between not adding too many integer bits (saving resources) and avoiding overflow for audio frames with high volume.

6 EVALUATION

In this section, we compare and analyze the accuracy and resource consumption of the following sound source localizer designs:

(1) The conventional design that uses weighted binary arithmetic with 4-bit precision signed inputs (Design (1)).
(2) The counter-based CC shown in Figure 8 with 4-bit unipolar (unsigned) inputs (Design (2)).
(3) The counter-based CC architecture shown in Figure 9 with LTC6992 for analog SN generation and 4-bit precision unipolar inputs (Design (3)).
(4) The architecture of Figure 9 with OR-based MAC units (Figure 6) and LTC6992 for analog SN generation of the 4-bit precision unipolar inputs (Design (4)).
(5) The architecture of Figure 8 with OR-based MAC units and digital 4-bit precision SNGs (Design (5)).
(6) The stochastic circuits using signed instead of unsigned inputs (Design (6)).

For all designs, the CC output has a bit-width twice that of the input due to the multiplication stage of the MAC unit. The fraction bits of the accumulation stage are truncated to represent the result with 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 post-processing tasks (maximum search, mapping, and average filter) are the same for all designs and implemented with conventional binary arithmetic.

6.1 Accuracy

We use the Grid corpus audio library [19], which consists of 50 files with male and female speakers for accuracy evaluations. All 50 files sum up to 90.08 sec. of audio data with 76.03 sec. audible speech (CC results are above the VAD threshold) and 14.05 sec. breaks. The distance between the sound source and the center of the two microphones is 1 m, and the speaker radiates with a 60 dB sound pressure level (SPL). The microphone SNR is 75 dB, and reverberations are disabled[3]. We sweep the sound source from -90° to 90° in a 5° grid.

Figure 10 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 and no external disturbances, such as reverberation and noise. Figure 10 (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. Figure 10 (b) shows that applying an appropriate window function after framing and increasing the window length reduces the systematic error. Figure 10 (a) also uses a Hanning window and a frame length of N_{frame} = 256 but computes the time delay through multiplication in the frequency domain instead of CC in the time domain. The sound source localizer with FFT-based TDEs has a lower systematic error for $|\phi| > 45°$ than the one labeled with Windowed. Figure 10 (a) shows 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.

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

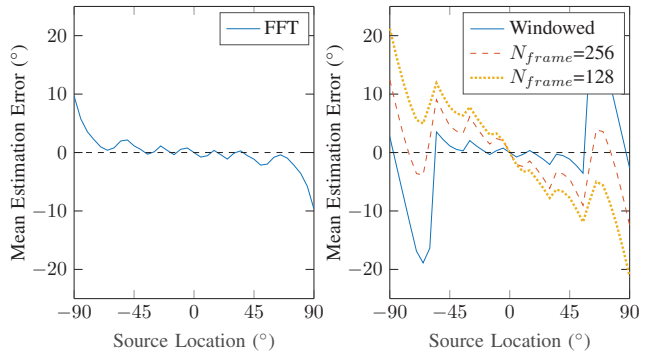

Figure 10: The Systematic error of the sound source localizer architecture. (a) Frame length of 256, Hanning window, FFT-based TDE, (b) variable frame length with and without Hanning window.

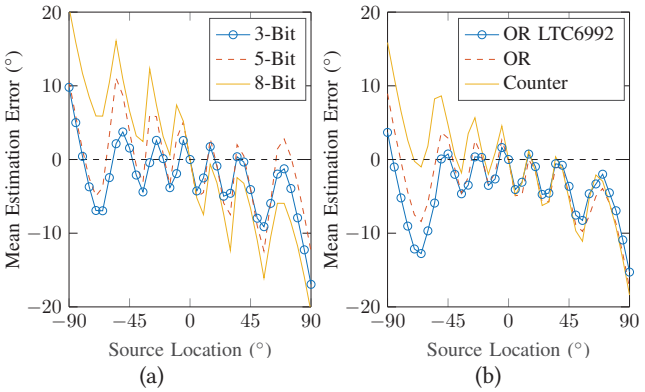

Figure 11: Mean estimation error of (a) Design (6) (exact and bipolar) and (b) Design (2), (4) and (5) (4-bit unipolar inputs).

Figure 11 shows the systematic bias of the sound source localizers that use fixed-point arithmetic, time domain CC, no Hanning window function, and a frame length of N_{frame} =128. Figure 11 (a) shows the mean SSL error for Design (6). 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 5-bit and 8-bit precision, the results are point symmetrical with $\phi = 0°$. The probability of having multiple equal maxima in the CC result increases with low precision. Figure 11 (b) shows the mean estimation error for the unipolar designs (Designs (2), (4), and (5)). 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 11, we also list the mean absolute error (MAE) in Table 1. The right-most column in Table 1 shows the mean of MAEs, below 8.6° for all designs. The MAE decreases with lower input precision because low precision designs are less susceptible to the systematic error. In our simulations, we observed that the rounding method during the quantization of the inputs affects the estimation error, and truncation rather than rounding gives a lower MAE. For example, the average MAE (last column of Table 1) 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 TDE.

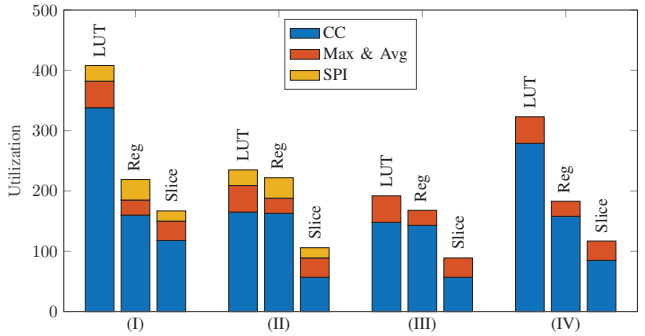

Figure 12: LUT, registers and slice utilization of the digital processing blocks. (I) Weighted binary, (II) counter, (III) Counter with LTC6992, (IV) OR with LTC6992.

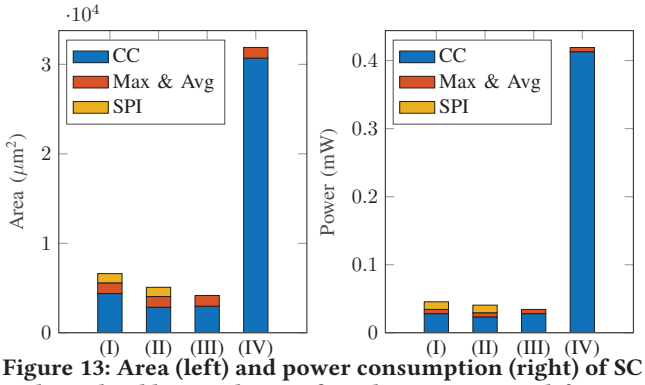

Figure 13: Area (left) and power consumption (right) of SC and weighted binary designs for 4-bit precision and f_s = 15.6 kHz. (I) Weighted binary, (II) counter, (III) Counter with LTC6992, (IV) OR with LTC6992.

6.2 Resource Consumption

Saving on hardware resources is one of the main goals of implementing SC-based systems. This section shows our synthesis results produced by the Synopsys Design Compiler v2018.06 with the 45nm FreePDK CMOS library [20] for the ASIC design flow and Vivado Design Suite for synthesis on the FPGA. The HDL synthesis tools are used to analyze the digital processing blocks.

The *analog board*, shown in Figure 4, has four black current sense resistors for measuring power consumption. The low current consumption of the *analog board*, in conjunction with the low resistance of the current sense resistor (R = 0.025 Ω), causes a small voltage drop over the current sense resistor that is difficult to measure. For example, the voltage drop over the current sense resistor of LTC1098 is approximately 0.1 $mA \times 0.025\ \Omega = 2.5\ \mu V$.

Figure 12 and Figure 13 show the resource consumption for the Max Search and Map & Average (referred to as Max & Average), SPI, and CC processing blocks for FPGA and ASIC implementations. The resource consumption of the 4-bit bipolar design (Design 6) is equal to that of the counter-based design (Design 2). The figures visualize the area reports of Vivado and Synopsys. However, we only show the power consumption for the ASIC design. 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. Synopsys reports that the designs with SC components have higher dynamic power consumption but lower static power [4]. Vivado and Synopsys estimate that the CC function consumes more power than the Max &

[4]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.

Table 1: 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 (6)	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 (2)		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 (3)		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 (4)		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**

Table 2: FPGA utilization of the CC function for f_s = 15.6 kHz.

	3-Bit Inputs			4-Bit Inputs			5-Bit Inputs		
Vivado	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

Table 3: ASIC area and power consumption of the CC function for f_s = 15.60 kHz.

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

Average and SPI processing blocks. Both HDL synthesis tools agree that counter-based Design (3) with LTC6992 consumes the least resources, followed by counter-based Design (2) and conventional Design (1). The synthesis tools estimate resource consumption for OR-based Design (3) with LTC6992. Synopsys reports about ten times higher power and area consumption, whereas the utilization report of Vivado shows *only* a doubling in the LUT utilization compared to other CC designs. The OR-based design with LTC6992 (Design (4)) stores a bit-stream accumulator with $nk = 240$ bits (for 4-bit precision arithmetic). FPGAs are optimized for storing data, but the ASIC design synthesizes a separate flip-flop for each bit. When comparing Design (1) and Design (2), Design (2) saves 11% power and 23% area by using the SC-based CC design. The savings of the digital implementation increase to 25% power and 37% area when comparing the conventional design (Design (1)) with Design (3). However, Design (3) uses the PWM modules on the analog board, which consume more power than the ADC and outweigh the savings from the digital design.

Table 2 and Table 3 list the resource consumption for the counter-based CC with digital SN generation (used in Design 2) and the conventional CC (used in Design 1). The tables show 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 FPGA 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 kHz. The clock speed of the SC-based CC increases from 873.6 kHz for 3-bit, to 3.744 MHz for 4-bit, and 15.4752 MHz for 5-bit ($f_{clk} = nk \times f_s$). Table 3 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.

7 DISCUSSION AND CONCLUSION

In this work, for the first time to the best of our knowledge, we proposed an accurate SC-based SSL system. To prove the functionality of the proposed 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 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-based adder is sensitive to data value changes. On the other hand, the counter-based CC is accurate and easy to use. Our synthesis results showed that the area consumption of the ASIC design decreases by 21% for 3-bit inputs and by 39% for 5-bit inputs. The power consumption increases with computational precision for weighted binary 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 sampling clock frequency of 15.6 kHz. 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 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-based 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. The readers can view a video demonstration of the implemented design on [21] and access the source code on [22].

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation (NSF) grant #2019511, the Louisiana Board of Regents Support Fund #LEQSF(2020-23)-RD-A-26, and a generous gift from Cisco.

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] B. R. Gaines, "Stochastic computing," in *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM, 1967, pp. 149–156.

[13] C. Chuah and T. Nandha Kumar, "Fast and exact multiple-input unary-to-binary multiplier with variable precision for stochastic computing," *Electronics Letters*, vol. 56, no. 10, pp. 482–485, 2020. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/el.2020.0206

[14] 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.

[15] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *ICCD*, Asheville, NC, USA, 2013, pp. 39–46.

[16] V. T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *DATE'18*, 2018, pp. 1417–1422.

[17] 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.

[18] V. T. Lee, A. Alaghi, J. P. Hayes, V. Sathe, and L. Ceze, "Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 13–18.

[19] 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.

[20] "NCSU FreePDK 45nm Library," https://eda.ncsu.edu/freepdk/freepdk45/.

[21] P. Schober, "Sound Source Localization using Stochastic Computing," https://youtu.be/ETfOnW--0bU, July 2022.

[22] "Sound source localization using stochastic computing," https://github.com/serco425/stochastic_computing_sound_source_localization/, 2022.