Performance and Error Tolerance of Stochastic Computing-based Digital Filter Design

Roshwin Sengupta and Ilia Polian
Institute of Computer Architecture and Computer Engineering
University of Stuttgart
Stuttgart, Germany
{roshwin.sengupta, ilia.polian}@iti.uni-stuttgart.de

John P. Hayes
Computer Engineering Laboratory
University of Michigan
Ann Arbor, Michigan, USA
jhayes@umich.edu

2024 27th International Symposium on Design &amp; Diagnostics of Electronic Circuits &amp; Systems (DDECS) | 979-8-3503-5934-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/DDECS60919.2024.10508903

Abstract— **Recent advances in near-sensor computing have prompted the need to design low-cost digital filters for edge devices. Stochastic computing (SC), leveraging its probabilistic bit-streams, has emerged as a compelling alternative to traditional deterministic computing for filter design. This paper examines error tolerance, area and power efficiency, and accuracy loss in SC-based digital filters. Specifically, we investigate the impact of various stochastic number generators and increased filter complexity on both FIR and IIR filters. Our results indicate that in an error-free environment, SC exhibits a 49% area advantage and a 64% power efficiency improvement, albeit with a slight loss of accuracy, compared to traditional binary implementations. Furthermore, when the input bit-streams are subject to a 2% bit-flip error rate, SC FIR and SC IIR filters have a much smaller performance degradation (1.3X and 1.9X, respectively) than comparable binary filters. In summary, this work provides useful insights into the advantages of stochastic computing in digital filter design, showcasing its robust error resilience, significant area and power efficiency gains, and trade-offs in accuracy compared to traditional binary approaches.**

Keywords*— *FIR filter, IIR filter, stochastic computing

I. Introduction

Digital filters, including both finite impulse response (FIR) and infinite impulse response (IIR) filters, play a vital role in various applications, like signal processing, communication, and biomedical signal analysis [1]. Notably, these systems can often tolerate approximate results but have stringent implementation constraints that demand low power consumption and compact size to cater to evolving computational needs across application domains. By utilizing random bitstreams, stochastic computing (SC) offers a promising way to address these design characteristics.

SC finds applications in diverse fields like image processing [3], neural networks [4][5], low-density parity-check (LDPC) decoding [6], and digital filters [7]-[11] due to its ability to perform fundamental arithmetic operations using simple logic circuits. For example, multiplexer (mux) based weighted adders are commonly employed in SC circuits for addition operations, although they pose accuracy challenges due to the inherent random fluctuation errors associated with SC's use of random bitstreams. While longer bitstreams can alleviate many errors, they come at the cost of increased latency and energy consumption. Addressing this concern, the CeMux addition method [12] takes advantage of correlation to reduce randomness in mux operations and significantly improve accuracy.

The drawbacks of accuracy loss and increased latency of SC are related to the stochastic numbers (SNs) generated by the stochastic number generators (SNGs), which typically consist of a random number source (RNS) and a comparator. The SC operations become faster and more energy efficient when the SNs rapidly converge to the target input numbers [13], while for accurate results, the generated SNs should closely match the input numbers. Furthermore, since the arithmetic operations are implemented by a small number of logic gates, the relatively large SNGs play a pivotal role in determining the overall circuit area. Most SC-related works utilize linear feedback shift registers (LFSRs) as the RNS in their SNGs [5]–[8]. Recent innovations [14] include low-discrepancy (LD) sequences, such as Sobol-based SNGs [15], that show improved accuracy over LFSR-based ones but may require additional hardware. Consequently, carefully selecting efficient SNGs is paramount for optimizing SC performance.

Hardware implementations of digital filters on resource-constrained edge devices should also perform efficiently in error-prone environments. Even though the probabilistic nature of SC makes it inherently resistant to transient or soft errors, there's a crucial gap in understanding the impact of SC on the error resilience of digital filters, compared to traditional binary computing. In this paper, we aim to bridge this gap by studying the impact of error tolerance by injecting bit-flips into the filter input stream. Our study considers several SNG types, such as LFSR-based and Sobol-based, and the use of correlation-enhanced multiplexer (CeMux)-based adders, and their impact on the error-tolerance, hardware area, and power consumption of SC FIR and IIR filters intended for electrocardiogram (ECG) filtering applications. Additionally, we assess the accuracy of various SC filters using the root mean square error (RMSE) metric.

The remainder of the paper is organized as follows: Section II provides the necessary background for stochastic computing, and FIR and IIR filters. Section III introduces our proposed stochastic filter designs. Section IV presents performance results and Section V concludes the paper

II. Background

A. Stochastic Computing

SC [1] has garnered attention for offering compact, error-tolerant, and low-power implementations of complex arithmetic functions. An SN is a sequence of k pseudo-random bits; an SN's value is determined by its 1s count. SNs come in two common formats: unipolar, representing values in the range [0, 1], and bipolar, extending the range to [–1, 1]. A unipolar SN's value is k_1 / k, where k_1 represents the number of 1s present, and a bipolar SN's value is $(k_1 - k_0) / k$, where k_0 is the number of 0s present. To illustrate, the SN $X_1 = 10111011$ has a unipolar value of 6/8, whereas, in bipolar SN representation, its value becomes 1/2.

One distinctive feature of SC is its inherent error tolerance as the order of 1s and 0s in an SN doesn't impact its numerical value. For instance, a single-bit error in X_1, changing its value from 6/8, may result in the closest representable numbers of either 5/8 or 7/8. In contrast, the conventional binary format

This work was supported in part by Deutsche Forschungsgemeinschaft (DFG), project number PO 1220/12-1. John P. Hayes was supported by the U.S. National Science Foundation under Grant CCF-2006704.

979-8-3503-5934-3/24/$31.00 ©2024 IEEE

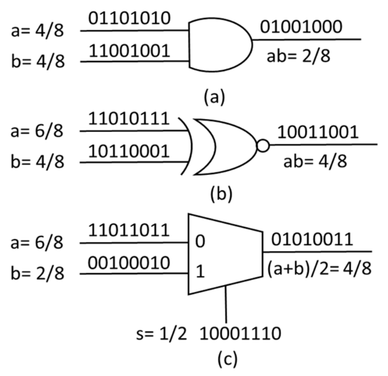


Figure 1: Unipolar multiplication (a), bipolar multiplication (b), and scaled addition using a multiplexer (mux) (c).

represents 6/8 as 0.110, and a single-bit flip can lead to a substantial error, especially if a high-order bit is flipped. For example, changing from 0.110 to 0.010 causes the value to shift from 6/8 to 2/8. Another important property of SC is correlation, which refers to dependency between SNs' bit patterns. Correlation can either benefit or hinder applications: unintended correlation often leads to biased operations and erroneous results, while correlation can be sometimes be deliberately exploited to enhance SC behavior [16]. The stochastic cross-correlation (SCC) metric proposed by [16], assesses the correlation between the bits of two SNs, by quantifying the expected overlap of occurrences of 1s. This can then be exploited for designing efficient SC circuits.

In SC, basic operations can be implemented with simple logic gates. Unipolar multiplication uses AND gates and bipolar multiplication uses XNOR gates. Scaled additions, performed by a mux, keep the sum within the desired range for both unipolar and bipolar SNs. SNs are generated using SNGs, often built around pseudo-random number generators like LFSRs. For a binary number $B \in [0, 1]$, the SNG will produce an k-bit unipolar SN with an expected value of B. For a bipolar SN with a value B_b, the input binary number is set to $B = (B_b + 1)/2$. Converting SNs back to binary is achieved through a stochastic-to-binary (S2B) module, typically composed of a counter that tallies the number of 1s in the SN. Fig. 1 depicts these SC operations.

Despite its numerous advantages, SC has downsides, with longer computation times being a notable example. The exponential increase in bit-stream length, coupled with a marginal increase in precision, contributes to these longer computation times. Hence, when considering SC for a specific application, a careful evaluation of its advantages and disadvantages is crucial.

B. FIR Filters

The finite impulse response (FIR) filter, as illustrated in Fig. 2(a), is used in audio, picture processing, communications systems, and biomedical engineering, and processes a limited number of input and output samples. In discrete-time FIR filters of order M, the impulse response spans $M + 1$ samples, settling to zero. The output sequence in a typical (causal) FIR filter is a weighted sum of the M most recent input samples. The equation for an M–th order discrete-time FIR filter is:

$$y[n] = b[0] \cdot x[n] + b[1] \cdot x[n\text{–}1] + \cdots + b[M] \cdot x[n\text{–}M] \quad (1)$$

Here $y[n]$ is the output sequence, $x[n]$ represents the input sequence, and $b[i]$ are the filter coefficients for $i = 0, 1, ..., M$. The hardware implementation of (1) involves adders, multipliers, and delay units often realized by D flip-flops.

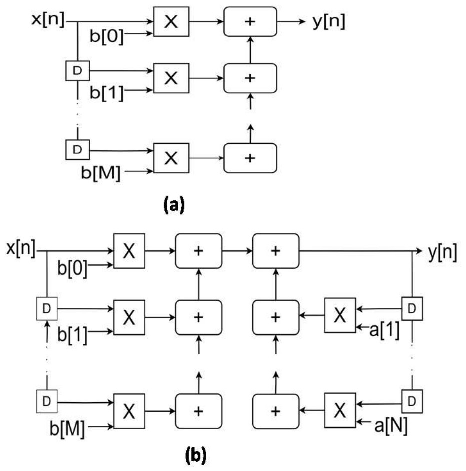


Figure 2: A traditional binary FIR filter (a) and IIR filter (b).

C. IIR Filters

Infinite impulse response (IIR) filters, depicted in Fig. 2(b), play a key role in some digital signal processing applications. Unlike FIR filters, IIR filters exhibit a recursive (feedback) structure, allowing them to achieve a desired frequency response with fewer coefficients. The general equation of an IIR filter is given by:

$$y[n] = b[0] \cdot x[n] + b[1] \cdot x[n\text{–}1] + \cdots + b[M] \cdot x[n\text{–}M] - a[1] \cdot y[n\text{–}1] - a[2] \cdot y[n\text{–}2] - \cdots - a[N] \cdot y[n\text{–}N] \quad (2)$$

Here $y[n]$ is the output sequence, $x[n]$ represents the input sequence, M is the order of the feedforward coefficients $b[i]$ and N is the order of the feedback coefficients $a[j]$, where $i = 0, 1, ..., M$ and $j = 1, 2, \ldots, N$. The implementation of IIR filters in hardware uses adders, multipliers, and D flip-flops. Adders perform summations of input samples and weighted coefficients, multipliers handle coefficient multiplication, and D flip-flops introduce delays in the signal path to achieve the recursive behavior. This hardware configuration allows the IIR filter to efficiently process input signals in real-time, making it suitable for various applications with constrained resources. The key advantage of IIR filters over FIR filters lies in their feedback property, which enables similar responses with fewer coefficients. This makes them suitable for memory-constrained devices and real-time applications. However, the feedback introduces stability problems not found in FIR filters, requiring careful analysis during design. Errors in one output can propagate and impact subsequent samples, posing a major design challenge for approximate methods like SC.

III. Proposed Stochastic Filters

In this section, we present detailed descriptions of our proposed stochastic-based FIR and IIR filters. We employ a careful selection of SNGs, including LFSR- and Sobol-based variants, and leverage the correlation impact of CeMux-based adders [12]. This results in creating three distinct architectures for both FIR and IIR filters. Our design strategically combines different SNGs and adder structures to investigate diverse aspects of performance and efficiency in stochastic-based filter implementations.

A. SC FIR Filter

The proposed implementation approach for SC FIR digital filters is outlined in Fig. 3, featuring two key stages, multiplication, and a mux adder tree for scaled addition of partial products. The design adheres to the structure described in (1), where one input vector $b[i]$ corresponds to the filter

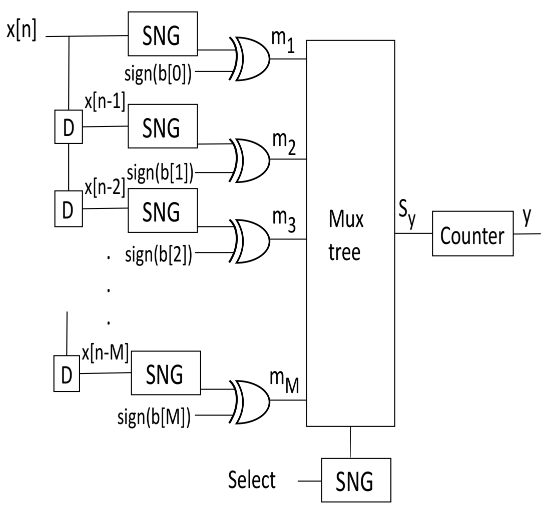


Figure 3: Proposed SC FIR filter, where the SNG and mux tree blocks are modified to generate the different SC FIR filters proposed in this work.

coefficients, and the other $x[n]$ represents the input signal in 2's complement format with $x[n-i]$ are the delayed inputs that are derived from the delay line. The workflow of an M–th order SC FIR filter begins with input sampling, followed by processing through delay units. Once the samples are acquired, the SC calculation unit operates, computing one bit per cycle until the desired output sequence length is reached. This process continues, collecting additional data for subsequent rounds of calculation.

SNs for delayed inputs can be generated by two methods. The first involves delaying the binary version of $x[n]$ and converting the delayed inputs to SNs via SNGs. The second method delays the SN version of $x[n]$ and directly feeds the delayed bit streams to the next module without using extra SNGs. In our work, we selected the first method to avoid having several registers of width equal to the stochastic bit-stream, which can consume excessive area. The first method requires registers of smaller widths (tens of bits) for binary numbers. The resulting stochastic bits and coefficients are subsequently processed serially to generate the final output.

The coefficients $b[i]$ are first converted to absolute values. To implement the absolute values of coefficients, unbiased stochastic sequences are assigned to the mux's select inputs. In an unbiased sequence, there is an equal chance of selecting each input. Additionally, XOR gates at the data inputs handle coefficient signs, inverting inputs for negative coefficients and acting as buffers for positive coefficients. Thus, the coefficients can be weighted by repeating the inputs appropriately. The output of the XOR gate is m_i, which is then fed to the mux tree adder. An input can be connected to multiple mux inputs to assign more weight to it. For instance, in the case of a weighted mux tree adder, as shown in Fig. 4, m_2 is connected to five mux inputs. Following Fig. 3, m_2 is associated with the same input $x[n-1]$, which means the probability of selecting $x[n-1]$ is 5/8; $b[1]$ is either 5/8 or –5/8. The output of the mux tree is then converted from SN to binary using a counter.

The SNG plays a key role in the efficient design of the SC FIR filter for improving the accuracy of the SNs and reducing the area of the overall circuit. In our work, we experiment with two different SNGs and compare their performance.

1) LFSR-based SC FIR (LFSR-SF)

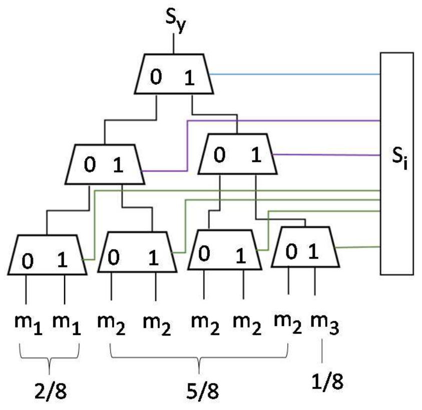


Figure 4: 3-input weighted hardwired mux tree [6] used as an adder.

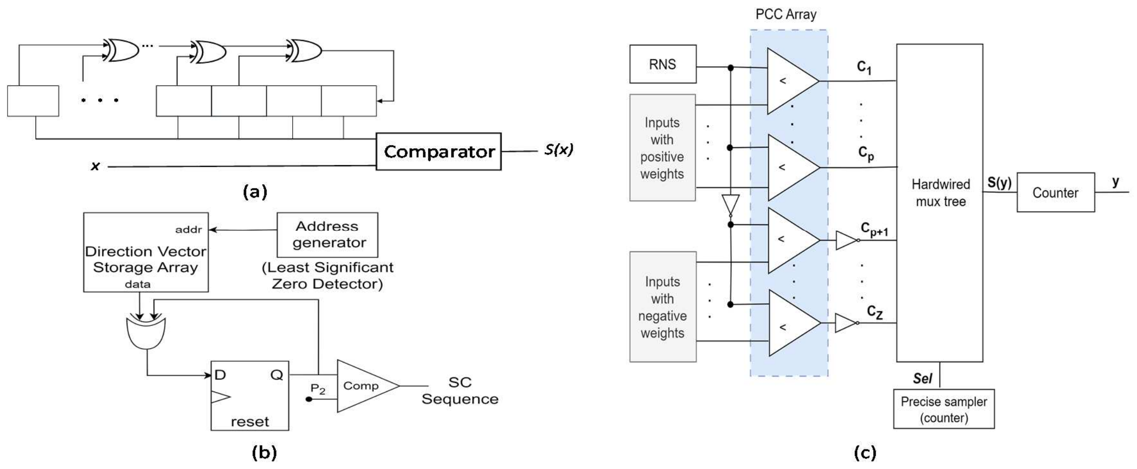
Our LFSR-SF filter design has an LFSR-based SNG, as illustrated in Fig. 5(a). An LFSR is a shift register with feedback, generating pseudo-random bit sequences. The length of the sequence depends on the register's stages. This generated bit sequence is determined by the LFSR's feedback and initial state. For SN generation, specific weights are associated with LFSR outputs, enabling the conversion of binary numbers to stochastic representations. For instance, a 4-bit LFSR can generate a 16-bit SN from a 4-bit unsigned binary number. Extra combinational logic is added to insert the all-zero state into the maximum-length nonzero state sequence of the LFSR. The SNG employs weight generation, converting binary number b to an SN bit-by-bit with the assigned weights. To enhance accuracy, seeds (initial values) of different SNGs are set to distinct values, avoiding the correlation issues associated with the reuse of the same seed.

While LFSR-based SNGs are prevalent in SC, studies have shown that these SNGs and the associated S2B circuits can account for over 80% of the overall circuit cost [17]. This partially mitigates the benefit of low resource utilization in SC, necessitating the exploration of various RNS types for SC applications.

2) Sobol-based SC FIR (Sobol-SF)

Originally used to expedite Monte-Carlo (MC) integration convergence [18], low discrepancy (LD) sequences like the Halton and Sobol have gained attention due to their reduced errors. This feature extends to SC because of the similarity between SN generation and MC sampling [15]. Sobol sequences exhibit smaller discrepancies than Halton since they eliminate the base-conversion overhead associated with Halton sequences. Sobol sequences also improve the accuracy of stochastic circuits and use shorter sequences than conventional LFSR-based SNGs. In this paper, we use the Sobol design shown in Fig. 5(b) for an energy-efficient SNG implementation.

Sobol sequence generation is a meticulous process involving the selection of direction numbers to determine the progression of the sequence in each dimension. These direction numbers are intermediate variables that are required for generating a Sobol sequence. They are constructed using carefully chosen primitive polynomials, adding a layer of precision to the entire sequence generation. The initial set of direction numbers (c_s) is derived from the binary representations of odd numbers smaller than 2^s. Through a series of bit-wise operations on these initial vectors, subsequent direction vectors are computed, ensuring a structured and high-quality Sobol sequence. Once all the



Comparator
S(x)
x
(a)
addr
Direction Vector Storage Array
data
Address generator
(Least Significant Zero Detector)
D
Q
reset
P2
Comp
SC Sequence
(b)
PCC Array
RNS
Inputs with positive weights
Inputs with negative weights
C1
Cp
Cp+1
CZ
Hardwired mux tree
S(y)
Counter
y
Sel
Precise sampler (counter)
(c)

Figure 5: LFSR-based SNG, where *x* denotes the binary input number and *S(x)* is the corresponding SN (a), Sobol-based SNG based on Gray code [15] (b), CeMux design that achieves full positive correlation [12] (c).

necessary direction vectors have been computed the *n*-th term x^n of a Sobol sequence is derived as:

$$x^n = b_1 c_1 \oplus b_2 c_2 \oplus \ldots \oplus b_{N-1} c_{N-1} \oplus b_N c_N \tag{3}$$

where $b_N b_{N-1} \ldots b_2 b_1$ is the Gray code representation of n.

3) CeMux-based SC FIR (CeMux-SF)

Our LFSR-SF and Sobol-SF filter designs use a mux tree for the addition operation, thereby encountering accuracy challenges due to random fluctuation errors. An alternative to a mux-based adder is the approximate parallel counter (APC)-based adder [19]. However, [12] shows that its proposed CeMux-based adder outperforms both mux and APC-based adders.

The CeMux structure, illustrated in Fig. 5(c), implements weighted addition using an XNOR multiplier array and a mux tree. In general, correlation among mux tree data inputs reduces errors caused by the mux selection process. When the SCC is +1 between all mux data inputs, it achieves full correlation, enhancing the mux computation's accuracy [12]. CeMux optimizes this by using a single RNS for data input SNs, which is connected to the probability conversion (PCC) array. The PCC array consists of *k*-bit comparators comparing data inputs with positive weights to the RNS value and comparing inputs with negative weights to the inverted RNS value. The array's output is a set of *Z* SNs, where *Z* is the number of inputs. A *k*-bit counter is assembled, with the *i*-th MSB connected to the select line of all muxes on the *i*-th level of the mux tree. The output of this tree is CeMux's output SN $S(y)$. This is then converted back to the binary output *y* using a counter. Our CeMux-SF filter design employs an LFSR-based SNG along with the CeMux adder proposed in [12] with minor modifications. While CeMux-based adders have demonstrated superior performance [12], relying on positive input correlation, some applications experience soft errors, that affect the proper correlation of mux inputs. We also analyze the performance of CeMux-SF in the presence of such errors.

B. SC IIR

In [8], the authors proposed lattice-based SC IIR filters for better performance. However, their architecture relies on multiple binary multipliers, leading to increased hardware complexity and resource utilization. Hence, for an efficient stochastic implementation of IIR filters, we follow the direct-form structure, which can then be implemented as the cascaded form by decomposing the high-order transfer function into a product of first- and second-order sections. This leads to efficient and more accurate SC IIR designs [20].

The SC IIR filter system, as illustrated in Fig. 6, is composed of two main parts: the feedback and feedforward modules responsible for performing the mathematical operation described by (2). The feedforward module computes the scaled product between the input vector $x[n]$ and the feedforward coefficients $b[i]$. The feedback module computes the operation between the feedback coefficients $a[i]$ and the output $y[n]$. In the SC IIR filter, the delayed inputs $x[n-i]$, as

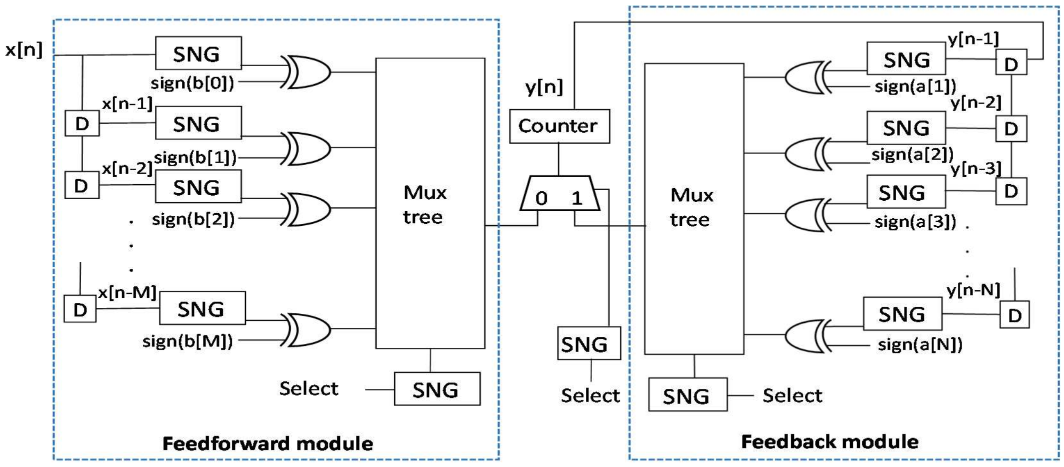


Figure 6: Proposed SC IIR filter where the SNG and mux tree blocks are modified to generate the different SC IIR filters introduced in this work.

well as the delayed outputs $y[n-j]$, can be obtained by either converting the input $x[n]$ and the output $y[n]$ and then passing through the delay element or first passing them through delay elements and then using SNGs for conversion. As mentioned in Sec. III.A, the second method introduces severe latency and storage concerns. These are even more pronounced for SC IIR filters as now apart from the input data, the output data also needs to be stored. Hence, instead of directly converting the input and the output signals into stochastic bit-streams before passing them through the delay line, the binary input signal first traverses the delay line. Each signal from the delay line is then individually converted into a stochastic bitstream, thus reducing latency and storage costs.

Both SC IIR modules comprise delay elements, SNGs, XOR gates, and mux trees. The outputs of these modules are combined by a two-way mux to obtain output SN which is converted to a binary number by a counter to obtain the final filter output $y[n]$. This result is then fed back to process a new sample. The inputs and selection lines to the modules are converted from binary to SN via SNGs. Like our FIR filter design, we have three different SC IIR filters: LFSR-based SC IIR (LFSR-SI), Sobol-based SC IIR (Sobol-SI), and CeMux-based SC IIR (CeMux-SI). The design for each is similar to the corresponding FIR filter explained in Sec. III.A.

IV. Experimental Results

To evaluate the performance of our SC FIR and SC IIR filter designs LFSR-SF, Sobol-SF, CeMux-SF, LFSR-SI, Sobol-SI, and CeMux-SI filters, we present the case study of electrocardiogram (ECG) filtering. We incorporate random noise into a benchmark ECG signal for testing [21]. Traditional digital filters, while effective, pose significant computational demands on ECG monitors. We use MATLAB to determine the coefficients of a lowpass filter with a cutoff frequency of 0.1π rad/sample. The purpose of this lowpass filter is to effectively eliminate high-frequency noise from the ECG signal.

A. Error Tolerance

The precision of an SN is defined by the bit stream [1]. Quantization errors, similar to those in traditional binary filters, occur due to this limited sequence length. SC also suffers from fluctuation errors, as the RNS in the SNG adds uncertainty to SNs. There are also correlation errors that arise from systematic non-zero cross-correlation among two or more SNs and non-zero autocorrelation of SNs in sequential circuits. In contrast, SC is known to be intrinsically tolerant to soft errors. In this section, we explore the extent of soft error tolerance in SC digital filters compared to conventional binary filters. We introduce errors into our filter design by inserting bit flips at the filter's input. The resulting errors can manifest themselves as single-bit, multi-bit, or burst errors. In SC, single-bit errors are considered negligible as their impact on SN errors is very low. As a result, single-bit errors are not addressed in this analysis. However, both multi-bit and burst errors are of significant concern for both binary and SC digital filters. Multi-bit errors involving simultaneous alterations of multiple bits, generally pose a greater threat than burst errors as they are not localized and can corrupt a number in a wider variety of ways. In the binary case, burst errors can have a more pronounced impact on circuit performance if they are concentrated in the MSB region. To ensure a balanced analysis of error tolerance, we consider a multi-bit error scheme. XOR gates are used to inject the errors into the target circuit. Each SC filter design from Sec. III is simulated using the derived filter coefficients and a noisy ECG signal as input. Table 1 provides insights into the effect of increasing percentages of bit flips on the RMSE (root mean square error) for different FIR and IIR filters. We highlight in bold the best RMSE for each error rate. The RMSE is calculated by 10,000 simulation runs and varying the error rate percentage. We use an 8-bit binary filter as our reference with the SC filters having an SN length of 1024 bits. We use 24th-order FIR and 6th-order IIR filters for analyzing the error tolerance of our filter designs, as they have the best performance in an error-free environment, as seen in Table II.

From Table 1, it can be observed that, at zero error rate, binary FIR and IIR filters deliver better performance than their stochastic counterparts, and the CeMux-based versions, with SCC of +1, outperform the LFSR and Sobol-based filters. At a 0.5% error rate, the CeMux-SF and CeMux-SI have SCC of +0.88 and +0.92 respectively, and either outperform or are comparable to Sobol-SF and Sobol-SI. This demonstrates that the correlation has a positive impact on the error resilience of SC filters at lower error rates. However, when the error rate reaches 1%, the SCC of CeMux-SF and CeMux-SI drop to +0.25 and +0.21 respectively, and they lose the advantage of full correlation, resulting in a significant degradation of their performance. The SCC further drops to -0.13 and -0.19 at an error rate of 2% for CeMux-SF and CeMux-SI respectively, worsening their performance. Sobol-based filters emerge as the most error-tolerant with higher error rates. They consistently outperform the LFSR-based filters, as the SNs generated by the Sobol sequence, and their multiplication results are more accurate than those by LFSR. Thus, in an error-prone environment, Sobol-based SC filters perform best. It is also worth noting, that at higher error rates, SC FIR filters have better performance than SC IIR filters, due to the impact of the feedback nature of the IIR filters.

A. Performance Analysis

We use the Synopsys Design Compiler with the Nangate 45nm open cell library to synthesize the different filters and estimate their respective area and power consumption. Table

Table 1. The Error Tolerance of all the proposed filters.

Error rate (%)	FIR filter (RMSE)				IIR Filter (RMSE)			
	Binary	**LSFR-SF**	**Sobol-SF**	**CeMux-SF**	**Binary**	**LSFR-SI**	**Sobol-SI**	**CeMux-SI**
0	**0.0269**	0.0683	0.0504	0.0386	**0.0153**	0.0562	0.0439	0.0295
0.1	0.0407	0.0702	0.0528	**0.0408**	0.0376	0.0589	0.0452	**0.0314**
0.25	0.0946	0.0725	0.0531	**0.0412**	0.0904	0.0605	0.0461	**0.0338**
0.5	0.1274	0.0734	0.0559	**0.0456**	0.1485	0.0681	**0.0508**	0.0574
1	0.5645	0.0816	**0.0608**	0.0746	0.5738	0.0891	**0.0626**	0.0982
1.5	0.9702	0.0979	**0.0643**	0.0850	0.8987	0.0964	**0.0801**	0.1003
2	1.3281	0.1182	**0.0701**	0.1226	1.5182	0.1193	**0.0868**	0.1397

TABLE II. THE ROOT MEAN SQUARE ERROR, AREA, AND POWER CONSUMPTION OF ALL THE FILTERS.

FIR filter						IIR Filter					
Order	**Performance**	**Binary**	**LSFR-SF**	**Sobol-SF**	**CeMux-SF**	**Order**	**Performance**	**Binary**	**LSFR-SI**	**Sobol-SI**	**CeMux-SI**
10	**RMSE**	0.0415	0.0981	0.0722	0.0571	2	**RMSE**	0.0407	0.0870	0.0591	0.0428
	Area (µm^2)	1032	571	508	**451**		**Area (µm^2)**	513	322	195	**108**
	Power (µW)	30.47	13.67	12.70	**11.01**		**Power (µW)**	19.82	7.53	6.14	**4.37**
16	**RMSE**	0.0369	0.0783	0.0614	0.0406	4	**RMSE**	0.0287	0.0751	0.0512	0.0347
	Area (µm^2)	1276	764	635	549		**Area (µm^2)**	648	505	232	275
	Power (µW)	39.45	15.81	14.92	13.55		**Power (µW)**	21.56	8.96	6.99	7.31
24	**RMSE**	**0.0269**	0.0683	0.0504	0.0386	6	**RMSE**	**0.0153**	0.0562	0.0439	0.0295
	Area (µm^2)	1652	982	851	729		**Area (µm^2)**	953	641	458	394
	Power (µW)	47.33	19.58	18.04	16.38		**Power (µW)**	23.45	10.48	7.45	8.04

II illustrates the performance of the binary and stochastic FIR and IIR filters through their RMSE, power consumption, and area in an error-free environment. CeMux-based SC filters are the smallest FIR and IIR filters respectively, because they use a single SNG for all the mux inputs and SNGs take up around 80% of the SC filter circuit. They achieve an average of 57% area reduction compared to their respective binary filters. The Sobol- and LFSR-based filters achieve around 50% and 40% area reduction, respectively. CeMux-SF and CeMux-SI also exhibit the lowest power consumption, achieving a 70% reduction. Sobol-SF and Sobol-SI reduce power consumption by 62% and LFSR-SF and LFSR-SI reduce it by 58%.

V. CONCLUSION

In this work, we have analyzed the impact of two different SNG types, LFSR-based and Sobol-based, as well as that of the correlation-enhanced stochastic mux adder CeMux on error tolerance, area, and power consumption of SC-based FIR and IIR filters. Our study highlights the remarkable error resilience of SC FIR and SC IIR filters, which experience only 1.3X and 1.9X performance degradation, respectively, in a 2% bit-flip scenario, in contrast to 49X and 66X decline in their respective binary counterparts. SC demonstrates a significant 49% area advantage and an impressive 64% improvement in power efficiency in an error-free environment, albeit at the cost of accuracy reduction compared to traditional binary implementations. Our analysis reveals that at higher error rates, Sobol-based filters exhibit significantly better error tolerance (40% on average) owing to their accurate SN generation. In terms of area and power, CeMux-based filters outperform Sobol-based filters by 15% and 9%, respectively.

In conclusion, our research suggests that CeMux-based filters are preferable for resource-constrained applications, while Sobol-based SC filters are better suited for error-tolerant applications. The study underscores the trade-offs that exist among area, power efficiency, and fault tolerance in different SC filters, and highlights the impact of correlation and SNG choice. Additionally, it demonstrates key potential advantages of SC over traditional binary computing.

VI. REFERENCES

[1] J. G. Proakis, and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*, USA: Prentice-Hall, 1996.

[2] A. Alaghi, W. Qian and J. P. Hayes, "The promise and challenge of stochastic computing," in IEEE Trans. Comput.-Aided Des. Integr. Circuits Sys., vol. 37, pp. 1515-1531, 2018.

[3] P. Li, and D. J. Lilja, "Using stochastic computing to implement digital image processing algorithms," ICCD, 2011.

[4] A. Ren, J. Li, Z. Li, C. Ding, X. Qian, Q. Qiu, et al, "SC-DCNN: Highly-scalable deep convolutional neural network using stochastic computing," ACM SIGOPS Oper. Syst. Rev., 2017.

[5] R. Sengupta, I. Polian and J. P. Hayes, "Stochastic computing architectures for lightweight LSTM neural networks," DDECS, 2022.

[6] W. J. Gross, V. C. Gaudet, and A. Milner, "Stochastic implementation of LDPC decoders," ACSSC, 2005.

[7] R. Wang, J. Han, B. F. Cockburn, and D. G. Elliott, "Design, evaluation and fault-tolerance analysis of stochastic FIR filters," Microelectronics Reliability, vol. 57, pp. 111-127, 2016.

[8] Y. Liu, and K. K. Parhi, "Architectures for recursive digital filters using stochastic computing," IEEE Trans. Sig. Proc., vol. 64, pp. 3705-3718, 2016.

[9] K. J. Ahmed, B. Yuan, and M. J. Lee, "High-accuracy stochastic computing-based FIR filter design," ICASSP, pp. 1140-1144, 2018.

[10] Z. Wang, and T. Ban, "Design, implementation, and evaluation of stochastic FIR filters based on FPGA," Circuits Syst. Signal Process, vol. 42, pp. 1142–1162, 2023.

[11] H. Ichihara, T. Sugino, S. Ishii, T. Iwagaki, and T. Inoue, "Compact and accurate digital filters based on stochastic computing," IEEE Trans. Emerg. Topics Comput., vol. 7, pp. 31-43, 2019.

[12] T. J. Baker, and J. P. Hayes, "CeMux: Maximizing the accuracy of stochastic Mux adders and an application to filter design," ACM Trans. Des. Autom. Electron. Syst., vol. 27, pp. 1-26, 2022.

[13] S. Liu, and J. Han, "Toward energy-efficient stochastic circuits using parallel Sobol sequences," IEEE Trans. VLSI Sys., vol. 26, pp. 1326-1339, 2018.

[14] I. L. Dalal, D. Stefan, and J. Harwayne-Gidansky, "Low discrepancy sequences for Monte Carlo simulations on reconfigurable platforms," ASAP, pp. 108–113, 2008.

[15] S. Liu, and J. Han, "Energy efficient stochastic computing with Sobol sequences," DATE, pp. 650–653, 2017.

[16] A. Alaghi, and J. P. Hayes, "Exploiting correlation in stochastic circuit design," ICCD, 2013.

[17] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," IEEE Trans. Computers, vol. 60, pp. 93-105, 2011.

[18] H. Niederreiter, *Random Number Generation and quasi-Monte Carlo Methods,* USA: Society for Industrial and Applied Mathematics, 1992.

[19] B. Parhami, and C. Yeh, "Accumulative parallel counters," ACSSC, 1995.

[20] N. Onizawa, S. Koshita, S. Sakamoto, M. Kawamata and T. Hanyu, "Evaluation of stochastic cascaded IIR filters," ISMVL, pp. 224-229, 2017

[21] G. B. Moody, W. E. Muldrow, and R. G. Mark, "A noise stress test for arrhythmia detectors," Computers in Cardiology, vol. 1, pp. 381-384, 1984.