Nonscaling Adders and Subtracters for Stochastic Computing Using Markov Chains

Nikos Temenos, *Student Member, IEEE*, and Paul P. Sotiriadis, *Senior Member, IEEE*

***Abstract*— This work presents adder and subtracter architectures for stochastic computing (SC). In contrast to standard approaches, the result of their operation is nonscaling, i.e., $X \pm Y$, and this is achieved via a deterministic operation based on a counting process. These properties result in an improved tradeoff between accuracy and stochastic sequence length, fast convergence, and the potential for cascaded, scale-dependent (e.g., nonlinear) stochastic computations providing with flexibility on the design level. The architectures are modeled using Markov chains (MCs) allowing for detailed understanding of their proper operation supported with analytical derivations. Using modified MC models, the adder and subtracter's internal register size is analytically calculated providing guidelines for its optimal size selection based on accuracy requirements and stochastic input sequences lengths. Both architectures are simulated in MATLAB and are designed in Synopsys to compare their performance to that of existing ones in terms of computational accuracy and hardware resources. Finally, to demonstrate the adder's efficacy, we use it as a building block to realize a 3 × 3 convolution kernel and then perform a standard digital image processing task. The results are compared to those achieved using adder architectures from the SC literature.**

***Index Terms*— Approximate computing, nonscaling addition, nonscaling subtraction, stochastic adders, stochastic computing (SC), stochastic subtracters.**

I. Introduction

With Moore's law stressed and new applications pushing for higher computational efficiency, research is intensified toward alternative numerical computing paradigms. A promising approach is stochastic computing (SC), where numbers and signals are encoded in a probabilistic form [1].

The essence of SC lies in its ability to realize basic arithmetic operations using only a few standard logic cells [2], and in contrast to conventional digital signal processors (DSPs), it is robust to soft errors [3]. SC enables massive parallelism and therefore proves attractive for hardware-demanding designs. These advantages have been exploited in several applications, including the field of digital image processing [4]–[6], neural networks (NNs) [7]–[13], soft polynomial solving and filtering [14], [15], as well as modern error-correcting coding and decoding [3], [16], [17].

Manuscript received January 8, 2021; revised April 7, 2021; accepted May 4, 2021. Date of publication July 20, 2021; date of current version August 30, 2021. This work was supported by the Hellenic Foundation for Research and Innovation (HFRI) through the HFRI Ph.D. Fellowship Grant under Fellowship 1216. *(Corresponding author: Nikos Temenos.)*

The authors are with the Department of Electrical and Computer Engineering, National Technical University of Athens, 15780 Athens, Greece (e-mail: ntemenos@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3095353.

Digital Object Identifier 10.1109/TVLSI.2021.3095353

The DSP cores utilized in the aforementioned applications rely mostly in multiply-and-add operations. The multiplication part is implemented using a single AND or XNOR gate according to the stochastic number representation used. Their addition part in SC though is typically realized by the MUX, which requires an additional random number source for its select signal, besides its inputs. However, the random number source by itself is a large block compared to the SC elements, occupying most of the design's area [18]. In addition, the adder's output is typically scaled by $1/2$, meaning that for a given sequence length, the resolution has dropped by 2. This makes the MUX less attractive for cascaded computations and blocks that rely on exact calculations. The same applies for the MUX implementation of the subtracter.

To address the former issues focusing on computational and design efficiency, several adders [19]–[22] and subtracters [22]–[24] have been published and explored [25] within the context of SC. The adder in [19] is based on the multiplexer's scaling principle but avoids the extra random number source by using a single T flip-flop, increasing also its accuracy. A similar (scaling) approach is presented in [20], but instead of a T flip-flop, it employs a two-state finite-state machine (FSM) to further increase its accuracy. The semistochastic approach explored in [26] uses the parallel input adder originally proposed in [25], which provides computations in a binary format that does not always favor next-stage SC-based computational blocks, for instance nonlinear functions. The nonscaling adder in [21] is based on a two-line representation of a stochastic number carrying its information in two sequences: one for its sign and one for its magnitude. Although it is a promising approach in application level [27], the two-line encoding imposes system design constraints as it requires other operations, e.g., multipliers, to follow this principle as well. Furthermore, the size of its counting unit is only estimated empirically [27]. Similar to the previous adder, the adder (and subtracter with one inverted input) presented in [22] encodes a stochastic number by using the ratio of logic ones and zeros between its input sequences. Yet, its unique representation is incompatible with standard SC formats, while the generation of two sequences for a single stochastic number influences the overall hardware utilization.

Regarding stochastic subtracters, the method in [23], [28], and [29] correlates the input sequences. This, however, requires caution since SC elements are prone to errors caused by correlated inputs [18]. Moreover, if the subtraction is an intermediate operation, regenerating correlated inputs is necessary. Another technique presented in [24] applies iterative logic units to enhance the accuracy of an XNOR gate with one

of its inputs inverted. As expected, it trades hardware resources and latency for accuracy, both depending on the number of stages used.

All the above methods trade circuit run time and/or hardware area for accuracy. In addition, certain of them introduce constraints that reduce the flexibility on the SC design space. Motivated by the aforementioned and to achieve the best of both worlds, we propose nonscaling adder and subtracter architectures for SC. They offer the following advantages: 1) they do not require any random number source; 2) they do not scale the output result; 3) they operate with independent and identically distributed (IID) input sequences (i.e., no specially correlated inputs required); 4) they are compatible with standard SC formats; and 5) they are fast converging, achieving high accuracy with short sequences lengths.

To demonstrate the properties of the two architectures, we model them using Markov chains (MCs), which allows for the derivation of their first-order statistics, the verification of their stochastic operation, and the analytic calculation of the internal register size, providing guidelines for their design based on accuracy requirements.

The remainder of this article is organized as follows. Section II provides a background on the standard notation used and the stochastic number representation. In Section III, the architecture of the proposed stochastic adder is presented and its MC is mathematically analyzed. Based on the proposed adder, Section IV briefly presents the proposed stochastic subtracter and its MC analysis. In Section V, we provide with a high-level estimation of the hardware requirements of the proposed architectures. In Section VI, we show extensive comparisons between the proposed architectures and the state of the art in both accuracy and hardware resources as well as discuss the impact of the results. To demonstrate the adder's efficacy in larger designs, in Section VII, we use it to implement a realistic digital image processing task and compare its effectiveness with other adders and techniques targeting SC implementations as well as the standard binary method. Finally, Section VIII concludes the present work.

II. Stochastic Number Representation

The stochastic number generator (SNG), shown in Fig. 1 [1], [3], is the standard circuit converting a k-bit deterministic number into its stochastic 0 and 1 sequence representation. A pseudorandom number generator uniformly distributed in $\{0, 1, \ldots, 2^k - 1\}$ and typically implemented as a k-bit linear-feedback shift register (LFSR) generates on every clock cycle a k-bit random number that is compared with the deterministic number $B \in [0, 1]$. The bit generation is completed after $N = 2^k$ clock cycles and corresponds to the length of the sequence [2], [3].

The N-bit output sequence generated by the SNG, i.e., $\{X_n\}, n = 1, 2, \ldots, N$, with n being the current time index (or clock cycle), is IID. It represents a nonnegative number in $[0, 1]$ and is known as unipolar format in SC. The probability of the stochastic number is defined as $X \triangleq P_r(X_n = 1) = B/2^k$, which is the normalized value of

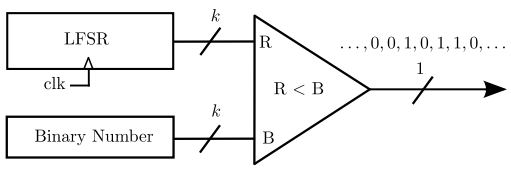

Fig. 1. Stochastic number generator (SNG) circuit [3].

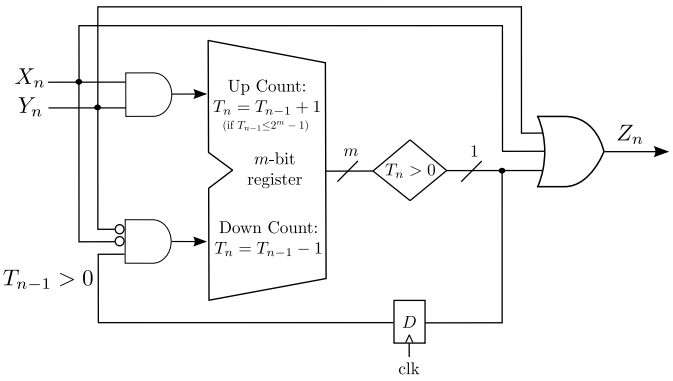

Fig. 2. Proposed stochastic adder architecture. T_n is the m-bit register's state, updated according to (1).

B in k-bit representation and its mean is given as

$$\widetilde{X}_N = \frac{1}{N}(X_1 + X_2 + \cdots + X_N).$$

Negative numbers (bipolar format) can also be represented using the transformation $X \mapsto 2X - 1$, expanding the range to $[-1, 1]$ [1]. For both stochastic number formats, the length of the sequence N is directly associated with the accuracy of the representation, which increases at the cost of additional clock cycles and is considered as SC's essential design tradeoff. In the following, we use the former basic facts to explain the operation of the proposed adder and subtracter architectures.

III. Stochastic Adder

This section introduces the proposed stochastic adder architecture and its mathematical analysis using MCs.

A. Architecture

The proposed adder architecture is shown in Fig. 2. If $\text{OR}(X_n, Y_n) = 1$, then the output is $Z_n = 1$. In the case of $X_n = Y_n = 1$, 1 is also stored and carried in the register (upcount by 1) in order to be outputted in the first future clock cycle n', i.e., $Z_{n'} = 1$, for which $X_{n'} = Y_{n'} = 0$. Moreover, when $X_n = Y_n = 0$, the register is downcounted by 1 if it had a positive prior value.

The procedure of storing 1 s, when $X_n = Y_n = 1$, and carrying them until they can be outputted compensates for the inability of the single-bit output to accommodate the instantaneous value of more than 1.

The above are captured in the schematic of Fig. 2, where the register is m-bit with current state T_n in the set $\mathcal{T}_R \triangleq \{0, 1, 2, \ldots, M - 1\}$, where $M = 2^m$. Note that T_n equals the

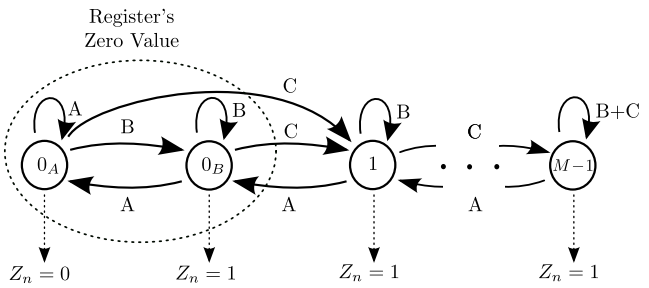

Fig. 3. MC model of the proposed stochastic adder. The register's zero state is represented by two states in the model, 0_A and 0_B. Transition probabilities A, B, and C are given by (3).

number of accumulated logic 1 s "owned" to the output, with initial value $T_0 = 0$ when the operation starts. From Fig. 2, one can conclude that the state of the adder evolves according to the following iteration:

$$T_n = \min\left\{T_{n-1} + X_n Y_n - \left(T_{n-1} > 0\right)\overline{X}_n \overline{Y}_n, M-1\right\} \quad (1)$$

where $\overline{X}_n = \text{NOT}(X_n) = 1 - X_n$ and similarly for $\overline{Y}_n$.

Although the proposed adder is designed to process stochastic sequences, its behavior is deterministic. As shown in Fig. 2, the output is a deterministic function of the inputs without any additional randomization, which could increase uncertainty and degrade precision. Specifically, the adder's output precision is determined by the length, N, of the input sequences, their stochastic properties, and the register's size, m.

B. MC Modeling

The operation of the stochastic adder architecture is modeled by the MC in Fig. 3. To explain its derivation, we note first that we assign two states 0_A and 0_B to the zero value of the register, whereas states 1 to $M-1$ represent the corresponding values of the register. Therefore, the MC state S_n can take the $M+1$ values in the set

$$\mathcal{S} \triangleq \{0_A, 0_B, 1, 2, \ldots, M-1\}. \quad (2)$$

Although using two zero states may appear confusing, it simplifies the analysis significantly because it allows us to relate the output value, Z_n, to the state only (i.e., the output is a function of the MC state and not of the inputs X_n and Y_n).

Let the MC's state be S_{n-1}. Then, the transition to the next state S_n and the output Z_n are determined according to the following transition probabilities:

$$\begin{aligned} A &= P_r(X_n = 0)P_r(Y_n = 0) \\ B &= P_r(X_n = 1) + P_r(Y_n = 1) - 2P_r(X_n = 1)P_r(Y_n = 1) \\ C &= P_r(X_n = 1)P_r(Y_n = 1). \end{aligned} \quad (3)$$

As shown in Fig. 3, there are three kinds of states: A) the two zero states 0_A and 0_B corresponding to register's zero state and also embedding information of the predecessor input-state pair; B) states 1 to $M-2$ capturing a sequential increase/decrease of the register's value; and C) state $M-1$ corresponding to the maximum value of the register that is also the overflow state in the case of $X_n = Y_n = 1$ with probability C. Extensive discussion on the register's overflow procedure and probability estimation follows in Section III-E.

To analyze the behavior of the MC, which captures that of the proposed stochastic adder, we proceed with standard definitions. The $(M+1) \times (M+1)$ transition probability matrix, with state ordering $(0_A, 0_B, 1, 2, \ldots, M-1)$, is defined as

$$H = \Big[P_r(S_{n+1} = s_b | S_n = s_a)\Big]_{s_a, s_b \in \mathcal{S}}$$

where the (s_a, s_b) entry of the matrix is the probability to transition to state s_b from state s_a. Matrix H is written as

$$H = \begin{bmatrix} A & B & C & \ldots\ldots & 0 \\ A & B & C & \ldots\ldots & 0 \\ 0 & A & B & C \ \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & A & B & C \\ 0 & \ldots\ldots & 0 & A & B+C \end{bmatrix}. \quad (4)$$

The probability distribution vector of state S_n defined as

$$p_n^T \triangleq \begin{bmatrix} P_r(S_n = 0_A) \\ P_r(S_n = 0_B) \\ P_r(S_n = 1) \\ \vdots \\ P_r(S_n = M-1) \end{bmatrix} \in [0,1]^{M+1} \quad (5)$$

can be expressed as

$$p_n = p_0 H^n \in [0,1]^{M+1} \quad (6)$$

where

$$p_0 = \big[1, 0, 0, \ldots 0\big] \in [0,1]^{M+1} \quad (7)$$

is the initial distribution vector and represents the starting state of the register $S_0 = 0_A$.

C. Expected Output Value and Error Characteristics

We use the MC model equations from Section II-B to derive the expected value of the adder's output. To this end, we first calculate the expected value of the instantaneous output Z_n. Note that since Z_n depends only on the state S_n, it is zero if and only if $S_n = 0_A$. Therefore, it is

$$\begin{aligned} \mathbb{E}[Z_n] &= P_r(Z_n = 1) = P_r(S_n \in \mathcal{S} - \{0_A\}) \\ &= 1 - p_0 H^n e_1^T \end{aligned} \quad (8)$$

where we used (6) and $e_i = [0, \ldots 0, 1, 0, \ldots, 0] \in \mathbb{R}^{M+1}$ is the ith normal vector. Then, the average value of the output N-bit sequence

$$\widetilde{Z}_N = \frac{1}{N}\Big(Z_1 + Z_2 + \cdots + Z_N\Big) \quad (9)$$

has expected value

$$\mathbb{E}[\widetilde{Z}_N] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[Z_n] = 1 - \frac{1}{N}p_0\left(\sum_{n=1}^{N} H^n\right)e_1^T. \quad (10)$$

Both the expected value of $\{Z_n\}$ and its mean are essential in quantifying the model's accuracy given its inputs $\{X_n\}$, $\{Y_n\}$ and will be used to verify the operation of the architecture.

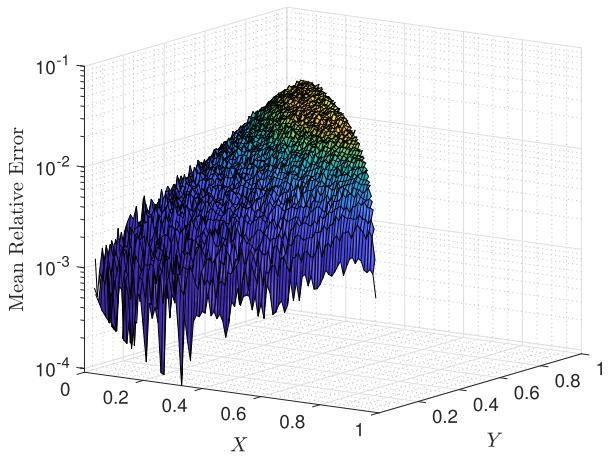

Fig. 4. Mean relative error calculated using (11) with sequence length $N = 32$ and a register of $m = 2$-bits. Results are in log scale.

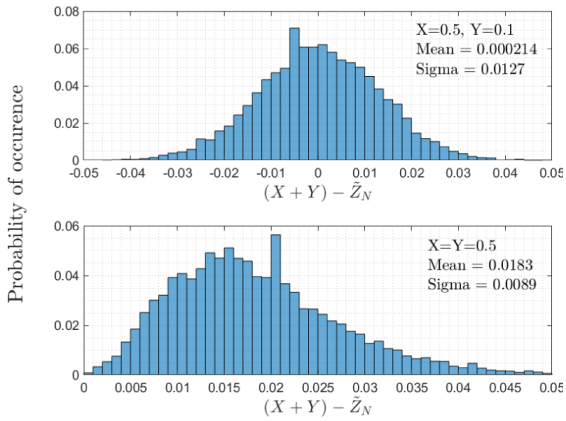

Fig. 5. Distribution of error $\tilde{Z}_N - (X + Y)$ for two selected cases with sequence length $N = 2048$ and register size of $m = 7$ bits.

1) Error Characteristics: To measure the adder's output precision and characterize the error behavior, we use the mean relative error (MRE) metric as follows:

$$Z_{\text{error}} = \mathbb{E}\left|\frac{(X+Y)-\tilde{Z}_N}{(X+Y)}\right| \tag{11}$$

where $\tilde{Z}_N$ is given by (9) and $X, Y \in [0, 1]$ are the probabilities of one of the input sequences. The MRE is estimated numerically. For every pair of input probabilities (X, Y) considered, the simulation is run 10^3 times with IID sequences $\{X_n\}, \{Y_n\}$. The results are shown in Fig. 4, in a logarithmic scale, for sequence length $N = 32$ and a register of $m = 2$-bits. As seen, the error is not uniform and peaks when $X + Y = 1$, especially for $X = Y = 0.5$. It decreases rapidly when moving away from these values.

To further illustrate the error characteristics, Fig. 5 shows the error distribution of $(X + Y) - \tilde{Z}_N$ for the two additions, $(X, Y) = (0.5, 0.1)$ and $(X, Y) = (0.5, 0.5)$, with $N = 2048$, $m = 7$, and 10^4 simulation runs. It is observed that for $(0.5, 0.1)$, the error's mean is about 2×10^{-5}, whereas for edge case of $(0.5, 0.5)$, the mean is increased to about 2×10^{-2}.

D. Verification of Operation

The operation of the proposed architecture as an adder is proven here. As above, for the IID input sequences, we use notation $X \triangleq P_r(X_n = 1)$ and $Y \triangleq P_r(Y_n = 1)$. In addition, we assume that $0 < X, Y < 1$ implying $A, B, C > 0$, as defined in (3). Therefore, the main, first upper, and first lower diagonals of matrix H in (4) are positive implying the following Lemma whose proof is straightforward.

Lemma 1: All entries of H^{M-1} are positive, i.e., $H^{M-1} > 0$.

The result of Lemma 1 implies that $(I + |H|)^{M-1} > 0$, which along with Theorem 1 from [30] below proves that H is irreducible.

Theorem 1: Matrix H is irreducible if and only if $(I + |H|)^{M-1} > 0$, where I is the identity matrix.

Moreover, since H is a stochastic matrix, its spectral radius is $\rho(H) = 1$ having 1 as an eigenvalue. Now, consider vector $v \in \mathbb{R}^{M+1}$ such that

$$v^T = \theta\left[1, \frac{1-A}{A}, \frac{\rho}{A}, \frac{\rho^2}{A}, \ldots, \frac{\rho^{M-1}}{A}\right] \tag{12}$$

where we have set

$$\rho \triangleq \frac{C}{A} = \frac{XY}{(1-X)(1-Y)} \tag{13}$$

$$\theta \triangleq A\frac{\rho-1}{\rho^M-1} \tag{14}$$

and $\underline{1} = [1, 1, \ldots, 1]^T \in \mathbb{R}^{M+1}$ is the column vector of ones.

It can be verified that v^T and $\underline{1}$ are left and right eigenvectors of H corresponding to eigenvalue 1, i.e., $v^T H = v^T$ and $H\underline{1} = \underline{1}$. Moreover, it is $v^T\underline{1} = 1$. From [30, Th. 8.6.1], we get that

$$\lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N} H^n = \underline{1}v^T \tag{15}$$

noting that $\underline{1}v^T$ is an $(M + 1)\times(M + 1)$ rank-one matrix. From (10) and (15), we get $\lim_{N\to\infty}\mathbb{E}[\widetilde{Z}_N] = 1 - p_0\underline{1}v^T e_1^T$. Since $p_0\underline{1} = 1$ and $v^T e_1^T = \theta$, we get $\lim_{N\to\infty}\mathbb{E}[\widetilde{Z}_N] = 1-\theta$, and by replacing θ, we have

$$\lim_{N\to\infty}\mathbb{E}[\widetilde{Z}_N] = 1 - A\frac{\rho-1}{\rho^M-1}. \tag{16}$$

We assume in addition that $X + Y < 1$, which along with $0<X, Y<1$ imply that $0 < \rho < 1$ and so $\lim_{M\to\infty}\rho^M = 0$. Therefore, since $1 - A(1 - \rho) = 1 + C-A = X + Y$, we get

$$\lim_{M\to\infty}\left(\lim_{N\to\infty}\mathbb{E}[\widetilde{Z}_N]\right) = X + Y \tag{17}$$

which proves the correct operation in the limiting case.

The result of (17) is valid for stochastic addition in unipolar format and it is extended directly to bipolar format via the transformation $Z \mapsto 2(Z - 1)$, where $Z = X + Y$ as before.

E. Register's Size and Overflow MC Model

The finite size of the m-bit register and the stochastic sequence length N impact the accuracy of the addition. Consider the definitions in (3) and suppose that the input sequences

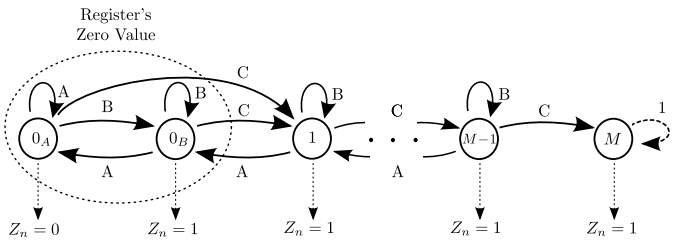

Fig. 6. MC overflow model of the proposed stochastic adder with absorbing state M. Transition probabilities A, B, and C are given by (3).

happen to have long segments of ones simultaneously. This means that at each clock cycle, the register's size increases by 1, i.e., it moves one state rightwise in the MC model in Fig. 3. Apparently, this cannot continue beyond state $M-1$ resulting in loss of counting ones, i.e., overflow. To select the register's size, it is important to investigate how the number of states M relates to overflow occurrences and when this leads to erroneous bits in the output.

1) Overflow MC Model: We start by modifying the MC model in Fig. 3 to get the one in Fig. 6. The difference of the two MC models is an extra state M in Fig. 6, which is absorbing. The overflow of the register appears when (and only when) its state is $M-1$ and the input bits are both ones. In this case, the state remains $M-1$ in the MC in Fig. 3, whereas it transitions to M in Fig. 6. Moreover, in the latter case, the state remains M, forever, indicating that there has been at least one overflow. Note that this is the sole purpose of this model and does not imply any change in the original architecture or the size of the register.

Defining the set of states of the new MC model, $\hat{\mathcal{S}} \triangleq \{0_A, 0_B, 1, 2, \ldots, M\}$, the probability vector of the state $\hat{S}_n$ at time n is defined as

$$\hat{p}_n^T \triangleq \begin{bmatrix} P_r(S_n = 0_A) \\ P_r(S_n = 0_B) \\ P_r(S_n = 1) \\ \vdots \\ P_r(S_n = M) \end{bmatrix} \in [0,1]^{M+2} \tag{18}$$

and the transition probability matrix $\hat{H} \in [0,1]^{(M+2)\times(M+2)}$, assuming the usual state ordering, is the following:

$$\hat{H} = \begin{bmatrix} A & B & C & \ldots & \ldots & \ldots & 0 \\ A & B & C & \ldots & \ldots & \ldots & 0 \\ 0 & A & B & C & \ldots & \ldots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ldots & 0 \\ \vdots & \ldots & 0 & A & B & C & 0 \\ \vdots & \ldots & \ldots & 0 & A & B & C \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 1 \end{bmatrix} \tag{19}$$

where A, B, and C are the probabilities given by (3). Note that $\hat{H}$ and H differ only in their last row.

Assuming that $S_0 = 0_A$ is the initial state of the register, then the initial probability vector is

$$\hat{p}_0 = \big[1, 0, 0, \ldots, 0\big] \in [0,1]^{M+2} \tag{20}$$

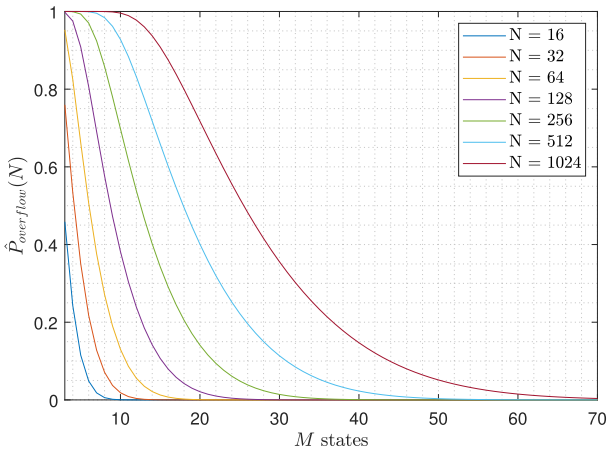

Fig. 7. Probability of overflow $\hat{P}_{\text{overflow}}(n)$, equation (22), for various numbers of register states $M = 3, 4, \ldots, 70$ and output stochastic sequences length N, when $X = Y = 0.5$.

and the probability vector at time n is expressed as

$$\hat{p}_n = \hat{p}_0 \hat{H}^n. \tag{21}$$

The probability that the register has overflowed at least once, before or at cycle n, is $P_r(S_n = M)$ expressed as

$$\hat{P}_{\text{overflow}}(n) \triangleq \hat{p}_0 \hat{H}^n e_{M+2}^T \tag{22}$$

where $e_i = [0, \ldots, 0, 1, 0, \ldots, 0] \in \mathbb{R}^{M+2}$ is the ith normal vector. One can observe that the maximum of $\hat{P}_{\text{overflow}}(n)$ occurs for $P_r(X_n = 1) = P_r(Y_n = 1) = 0.5$, and it decreases when moving away from these values.

A graphical illustration of (22) is shown in Fig. 7 for $P_r(X_n = 1) = P_r(Y_n = 1) = 0.5$, $M = 3, \ldots, 70$ and typical output sequence lengths $N = 2^5, 2^6, \ldots, 2^{10}$. As expected, increasing the number of states M reduces the probability of overflow.

To further investigate the overflow process, we can consider the expected number of transitions to reach the absorption state, N^*. It is given by [31], [32]

$$N^* = p_0 F \underline{1} \tag{23}$$

where p_0 is the initial distribution vector, $\underline{1}$ is the column vector of $M+1$ ones, and $F \in [0,1]^{(M+1)\times(M+1)}$ is the fundamental matrix of the absorbing MC [31], [32] defined as

$$F = (I - \tilde{H})^{-1}. \tag{24}$$

Matrix $\tilde{H}$ is given by the decomposition of the $(M+2)\times(M+2)$ matrix $\hat{H}$ in the form

$$\hat{H} = \left[\begin{array}{c|c} \tilde{H} & R \\ \hline 0 & 1 \end{array}\right]. \tag{25}$$

The expected number of transitions to absorption is used in the following to select the register's size.

TABLE I
BINARY REGISTER SIZE m-BIT ($M = 2^m$) SELECTION

Sequence length N-bits	16	32	64	128	256	512	1024
Register size m-bits	2	2	3	3	4	4	5
$\hat{P}_{overflow}(N)$	0.24	0.53	0.27	0.56	0.29	0.60	0.30

2) Error Due to Overflow: An overflow by itself does not necessarily imply an erroneous output bit. For example, in the extreme case where all input bits of both sequences are one, the register may overflow repeatedly, but the architecture's output, 1, is always correct.

Consider now the following scenario. Start from state 0_A and transition to state $M-1$ strictly monotonically, i.e., with both inputs being one. This requires $M-1$ transitions and there is no overflow. State $M-1$ is critical; the next state may be either $M-2$, if both inputs are zero, or $M-1$ again if at least one input is one. If both inputs are one, then we have the first overflow.

Therefore, the minimum time required for the first overflow to possibly occur is M clock cycles (transitions). For the overflow to be observed at the output, the state of the register must become 0_A to output a zero, which requires a minimum of $M-1$ transitions leftwise from state $M-1$ (i.e., both inputs be zero).

Therefore, the minimum number of clock cycles (transitions) that can (may) result in an erroneous output bit is $N_{\text{er}} = 2M-1$, and therefore, the following condition guarantees that all output bits are correct

$$N < N_{\text{er}}. \tag{26}$$

Typical register sizes, M, imply relatively small values of N_{er} and so of N, if we require zero errors. Therefore, we prefer to relax our requirements and statistically allow some overflows potentially implying some errors. To this end, we use N^* in (23) as a guide to select M. Note that N^* is an implicit function of M, X, and Y.

We start with selecting the value of N. Then, for $X = Y = 0.5$, we derive the values of A, B, and C, and therefore, N^* becomes a function of M only. We chose $M = 2^m$ to be the smallest power of 2 resulting in $N^* \geq N$. The values of m are presented in Table I along with the corresponding probability of overflow, for $X = Y = 0.5$, given by (22).

F. Deviation of the Output From IID

To estimate the statistical deviation of the output sequence $\{Z_n\}$ from the ideal IID, we use the stochastic computing correlation (SCC) metric [3], [28]. For Z_n and Z_k with $k = n + r$ and $r > 0$, it is

$$\text{SCC}(Z_n, Z_k) = \begin{cases} \dfrac{\mathbb{E}[Z_n Z_k] - \mathbb{E}[Z_n]\mathbb{E}[Z_k]}{\min(\mathbb{E}[Z_n], \mathbb{E}[Z_k]) - \mathbb{E}[Z_n]\mathbb{E}[Z_k]}, & \mathbb{E}[Z_n Z_k] > \mathbb{E}[Z_n]\mathbb{E}[Z_k] \\ \dfrac{\mathbb{E}[Z_n Z_k] - \mathbb{E}[Z_n]\mathbb{E}[Z_k]}{\mathbb{E}[Z_n]\mathbb{E}[Z_k] - \max(\mathbb{E}[Z_n] + \mathbb{E}[Z_k] - 1, 0)}, & \text{otherwise.} \end{cases} \tag{27}$$

Note that $\text{SCC}(Z_n, Z_k) \in [-1, 1]$ with zero corresponding to uncorrelated random variables.

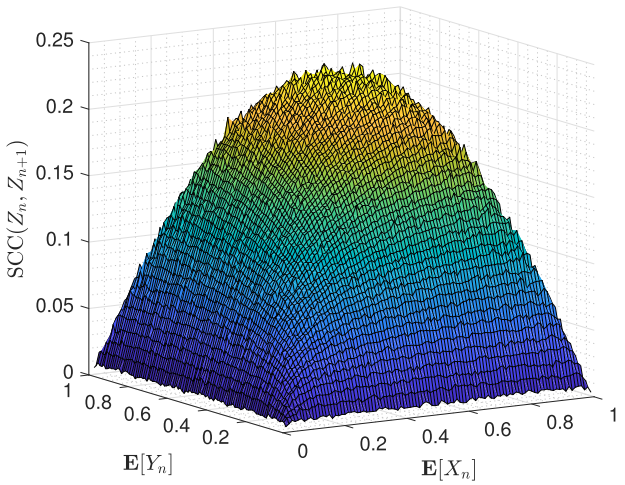

Fig. 8. SCC(Z_n, Z_{n+1}) of IID inputs X, Y with $N = 1024$ and $m = 5$ bit.

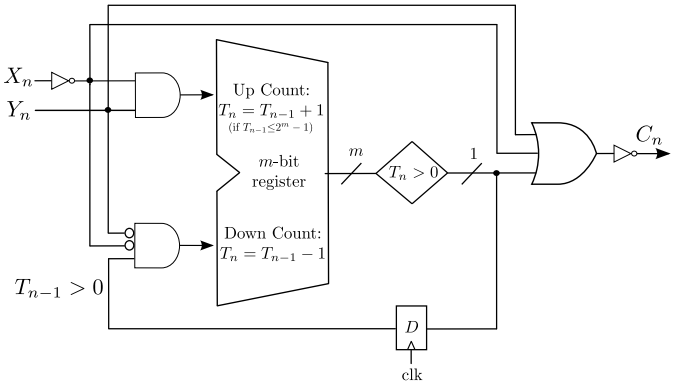

Fig. 9. Proposed stochastic subtracter architecture. T_n is the m-bit register's current state.

Fig. 8 shows SCC(Z_n, Z_{n+1}) for IID input sequences with mean values $X, Y \in [0, 1]$ and $X + Y \leq 1$ when $m = 5$ bits and $N = 1024$. SCC(Z_n, Z_{n+1}) achieves its maximum value of 0.21 when $X = Y = 0.5$, decreasing to zero when moving away from $X = Y = 0.5$. This potentially allows us to use $\{Z_n\}$ and perform the next-stage calculations with a single D flip-flop to achieve the delay of $r = 1$, for instance to calculate $(X + Y)^2$ as $\mathbb{E}[Z_n Z_{n+1}]$.

IV. STOCHASTIC SUBTRACTER

This section introduces the proposed stochastic subtracter architecture and its mathematical modeling, using MCs, following the definitions, analysis, and operating principles of the stochastic adder.

A. Architecture

The proposed subtracter architecture is shown in Fig. 9. It is comprised of the proposed stochastic adder with inverted one input and its output. Therefore, the subtracter operates like the adder with inputs $\overline{X}_n$ and Y_n having probabilities $P_r(\overline{X}_n = 1) = 1 - X$ and $P_r(Y_n = 1) = Y$, respectively. The addition operation implies that $\widetilde{Z}_N \approx 1 - X + Y$ and the output inversion

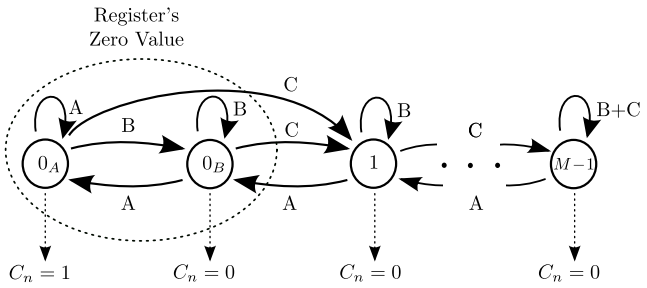

Fig. 10. MC model of the proposed stochastic subtracter. The register's zero state is represented by two states in the model, 0_A and 0_B. Transition probabilities A, B, and C are given by (29).

gives $\widetilde{C}_N = 1 - \widetilde{Z}_N \approx X - Y$. For the subtracter to operate appropriately, it must be $X \geq Y$.

Similar to the stochastic adder, the counter's value T_n (with initial value $T_0 = 0$) belongs to $\mathcal{T}_R \triangleq \{0, 1, 2, \ldots, M-1\}$, where $M = 2^m$ and m is the register's size. The state T_n indicates the number of logic 1 s "owned" to the addition $(1 - X_n) + Y_n$ and evolves according to

$$T_n = \min\Big\{T_{n-1} + \overline{X}_n Y_n - \big(T_{n-1} > 0\big) X_n \overline{Y}_n, M-1\Big\}. \quad (28)$$

B. MC Modeling

The operation of the proposed subtracter architecture is modeled by the MC model shown in Fig. 10. Both zero states 0_A and 0_B represent the zero value of the register, whereas 1 to $M-1$ represent the corresponding nonzero values of the register. Moreover, the state of the MC model S_n belongs to the set of $M+1$ elements given in (2), while the state S_n and its output C_n are determined by the following transition probabilities:

$$\begin{aligned} A &= P_r(X_n = 1) P_r(Y_n = 0) \\ B &= P_r(X_n = 1) P_r(Y_n = 1) + P_r(Y_n = 0) P_r(X_n = 0) \\ C &= P_r(X_n = 0) P_r(Y_n = 1). \end{aligned} \quad (29)$$

The analysis of the MC's behavior can be obtained by using (4), (5), and (7), along with (29) to calculate the probability distribution vector of state S_n after $n = 1, 2, \ldots, N$ steps. Note that although matrix H is the same for both the adder and subtracter, transition probabilities A, B, and C are different. Also, the MC models are the same, except for the output values.

C. Expected Output, Error Characteristics, and Verification of Operation

According to the MC model of Fig. 10, it is $C_n = 1$ if and only if $S_n = 0_A$. Therefore, the expected value of the instantaneous output is

$$\mathbb{E}[C_n] = P_r(C_n = 1) = P_r(S_n = 0_A) = p_0 H^n e_1^T. \quad (30)$$

The average value of the output N-bit sequence is

$$\widetilde{C}_N = \frac{1}{N}(C_1 + C_2 + \cdots + C_N), \quad (31)$$

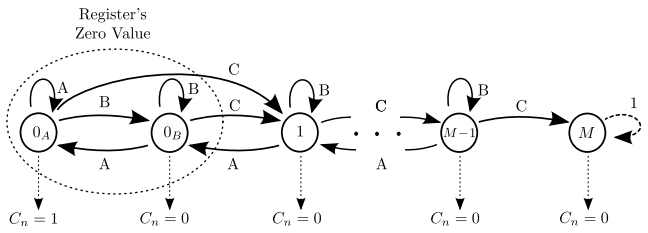

Fig. 11. MC overflow model of the proposed stochastic subtracter with absorbing state M. Transition probabilities are given by (29).

with expected value given by

$$\mathbb{E}[\tilde{C}_N] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[C_n] = \frac{1}{N} p_0 \Big(\sum_{n=1}^{N} H^n\Big) e_1^T. \quad (32)$$

We note that the subtractor's error characteristics follow a similar behavior to those of the adder's error shown in Section III-C. The procedure to verify the operation of the proposed subtracter architecture is identical to that of the adder in Section III-D. Following Lemma 1 and since matrix H is irreducible according to Theorem 1 and corresponding assumptions, we conclude that the operation at the limit case using (32) implies

$$\lim_{M\to\infty}\left(\lim_{N\to\infty}\mathbb{E}[\widetilde{C}_N]\right) = X - Y. \quad (33)$$

Also, it can be shown that bipolar representation $C = X - Y$ of the stochastic subtracter is achieved using $C \mapsto 2C$.

D. Register's Size and Overflow MC Model

The transitions of the subtracter's register values follow the same principles as in the adder. Therefore, for an N-bit sequence, the register does not overflow as long as it satisfies (26), while Fig. 11 shows the MC overflow model.

The state of the MC overflow model, $\hat{S}_n$, transitions within $\hat{\mathcal{S}} \triangleq \{0_A, 0_B, 1, 2, \ldots, M\}$ of cardinality $M+2$, while $\hat{S}_n = M$ is the absorbing state indicating the overflows. As in the adder, the extra state M does not imply an increase in the register's size or a modification in the architecture of Fig. 9. Furthermore, the probability that the register has overflowed at least once, before or at cycle n, is (22). It can be used to select the register's size based on accuracy requirements as explained in the adder's section and summarized in Table I.

E. Deviation of the Output From IID

The subtractor's output SCC is calculated similar to that of the adder, using (27), but with (30) for deriving the expected value of the output $\mathbb{E}[C_n]$. $\mathrm{SCC}(C_n, C_{n+1})$ for IID input sequences with mean values $X, Y \in [0, 1]$ and $X \geq Y$ when $m = 5$ bits and $N = 1024$ is identical to the that of the adder, as shown in Fig. 8.

V. High-Level Hardware Requirements Estimation

In this section, we provide a high-level estimation of the proposed architectures' hardware requirements. It is important to note that the correspondence between the architectures of

Figs. 2 and 9 and the hardware estimation in the following may vary as the latter is heavily dependent on two factors: 1) the optimization tools and 2) the technology used.

A. Total Cell Area

The adder of Fig. 2 is composed of the following gates and instances: 1) ANDX1; 2) 2 INVX1; 3) NAND3 $\times$ 1; 4) m-bit accumulator containing m-bit register and adder (FAX1); 5) m-bit comparator; 6) D-FF; and 7) OR3 $\times$ 1. Note that the comparator's total cell area is reduced compared to the standard one, given the fact that only an equality comparison with the first state is necessary as $Z_n = 0$ implies $S_n = 0$. For instance, for $m = 2$, the comparator is reduced to 3 INVX1's and an ANDX1. The subtracter's total cell area is estimated similar to the adder's, by including two INVX1's.

B. Propagation Delay

By inspecting the architectures is Figs. 2 and 9, one can observe that the propagation delay is the path from the D-FF to the counter's register. It is calculated as the minimum total time required to pass through each of the following logic cells and instances (in order): 1) NAND3 $\times$ 1; 2) INVX1; 3) AND2 $\times$ 1; 4) INVX1; 5) OAI21 $\times$ 1; 6) AND2 $\times$ 1; and 7) m-bit FAX1, where 3–7 correspond to the state update.

C. Energy Consumption

From a high-level system perspective, both the adder and the subtracter are FSMs, and thus, their state update is a function of their previous state and inputs, i.e., $S_n = f(S_{n-1}, X_n, Y_n)$. Consequently, the energy consumed during the update process is also a function of the previous state and inputs, namely $w_n = E(S_{n-1} = s, X_n = x, Y_n = y)$, where $s \in \mathcal{S}$ and $x, y \in \{0, 1\}$, depending on the circuit implementation, loading conditions, technology node, and so on. Since the probability of this transition is $P_r(S_{n-1} = s, X_n = x, Y_n = y)$, the expected *instantaneous* energy consumption is

$$\mathbb{E}[w_n] = \sum_{\substack{s \in \mathcal{S} \\ x, y \in \{0,1\}}}^{N} E(s, x, y) P_r(S_{n-1} = s, X_n = x, Y_n = y). \tag{34}$$

Therefore, the *expected* total energy cost of the operation, assuming N-bit input sequences, is $\sum_{n=1}^{N} \mathbb{E}[w_n]$.

VI. Comparison of the Proposed Architectures With Existing Ones

In this section, we compare the proposed architectures with popular SC adders [1], [19]–[22] and subtracters [22]–[24] in the literature. We measured the performance in terms of accuracy using the mean absolute error (MAE) with simulations in MATLAB. To this end, we considered a grid of pair values (X, Y), assuming the unipolar SC format. For each grid point, we performed 10^3 runs with pairs of IID sequences to derive the corresponding MAE. Then, we averaged over all MAE values of each architecture. The experiment was run for stochastic sequence lengths $N = 2^k$, where $k = 4, \ldots, 10$.

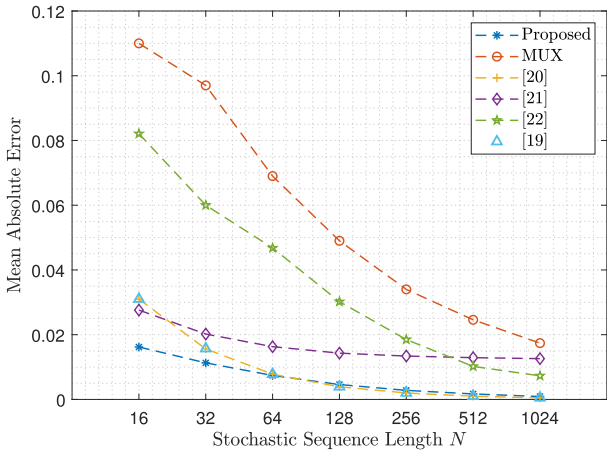

Fig. 12. Comparison of accuracy in MAE of stochastic adders for typical stochastic sequence lengths N.

All designs were synthesized using Verilog HDL in the Xilinx Kintex-7 FPGA kit and fed into the Synopsys Design Compiler using the FreePDK CMOS library at 45 nm [33]. For the comparison, we provide the following estimates: 1) the total area in μm^2; 2) the average power consumption for the max operating frequency in mW; 3) the critical path in ns; and 4) the energy per operation (average power $\times$ the critical path) in pJ.

A. Stochastic Adders

The accuracy, the power $\times$ delay^2, and energy consumption of the adders considered are presented in Figs. 12 and 13, while their detailed hardware requirements, including the area, are cited in Table II. Note that the hardware requirements for the input sequence generation are not included.

1) MUX: We consider the original circuit used for scaled addition (in unipolar format) and scaled subtraction (in bipolar) [1]. It requires large sequence lengths N to achieve acceptable accuracy compared to the other architectures, which reflects the increased total energy consumption. Moreover, the required SNG also impacts both power and energy consumption. This is why it is the least popular approach for addition and subtraction.

2) Adders in [19] and [20]: The adder in [19] uses a T flip-flop to replace the SNG of the original MUX adder, while the adder in [20] employs a 1-bit register along with a two-state FSM to slightly improve on the accuracy.

Compared to the adder in [20], the proposed one requires almost the same area for a register of $m = 2$-bits and slightly more power and energy consumption per operation according to Table II. Compared to the adder in [19], the proposed one has higher power and energy consumption per operation. However, the proposed adder achieves better accuracy than both of them for short sequence lengths according to Fig. 12. Moreover, the nonscaling behavior of the proposed adder benefits cascaded computations since the resolution of the sequence is not reduced by 2 for every adder used. This is discussed further in Section VII.

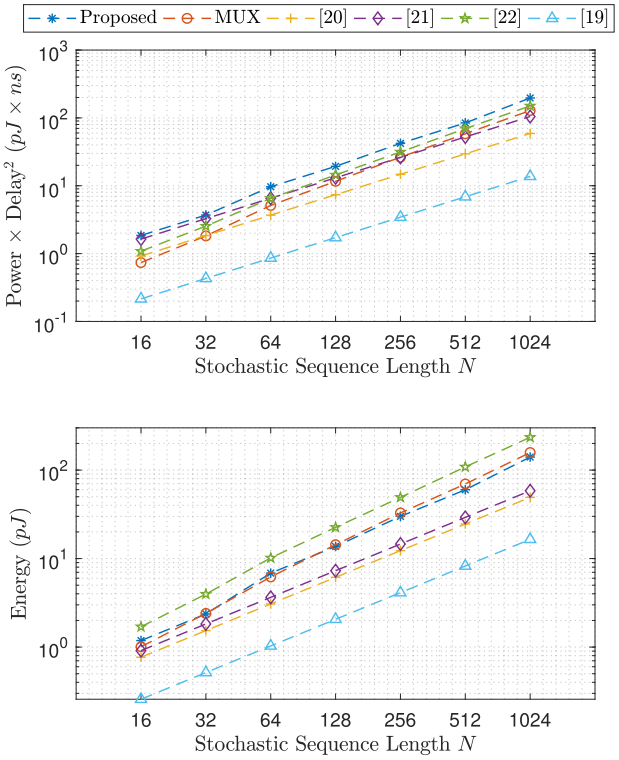

Fig. 13. Comparison of power × delay^2 (pJ × ns) (top) and energy (pJ) (bottom) consumption of stochastic adders for typical stochastic sequence lengths N.

TABLE II

HARDWARE REQUIREMENTS COMPARISON BETWEEN THE PROPOSED ARCHITECTURES AND STATE OF THE ART IN AREA (μm^2), CRITICAL PATH (ns), POWER CONSUMPTION (mW), AND ENERGY (pJ) PER OPERATION

	Register (bit)	Area (μm^2)	Power (mW)	Critical path (ns)	Energy (pJ)
		Stochastic Adders			
Proposed* Adder/Subtracter	$m=2$	59.60	0.053	1.4	0.074
	$m=3$	83.49	0.077		0.108
	$m=4$	98.30	0.084		0.117
	$m=5$	112.61	0.098		0.137
[19]		22.41	0.021	0.8	0.016
[20]		54.39	0.040	1.2	0.048
[21]		92.49	0.071	1.2	0.057
MUX* Adder/Subtracter LFSR size k	$k=4$	74.34	0.079	0.8	0.063
	$k=5$	97.62	0.094		0.075
	$k=6$	122.09	0.122		0.097
	$k=7$	133.71	0.140		0.112
	$k=8$	168.21	0.160		0.128
	$k=9$	177.40	0.171		0.136
	$k=10$	192.20	0.193		0.154
[22]* Adder/Subtracter LFSR size k	$k=4$	83.49	0.106	0.8	0.084
	$k=5$	105.69	0.124		0.099
	$k=6$	126.24	0.159		0.127
	$k=7$	144.54	0.176		0.140
	$k=8$	168.65	0.192		0.153
	$k=9$	178.64	0.212		0.169
	$k=10$	194.75	0.229		0.183
		Stochastic Subtracters			
[24]		41.76	0.063	0.8	0.050
[23]** LFSR size k	$k=4$	105.12	0.14	0.8	0.112
	$k=5$	130.75	0.176		0.140
	$k=6$	166.13	0.229		0.183
	$k=7$	188.65	0.264		0.211
	$k=8$	207.01	0.299		0.239
	$k=9$	229.95	0.331		0.264
	$k=10$	264.54	0.350		0.280

* In these cases the subtracter is obtained with negligible additional hardware requirements (2 NOT gates for the proposed adder and 1 for the rest) and its impact is insignificant in the area, power and energy consumption
** Includes sequence correlation

3) Adder in [21]: The nonscaling adder in [21] assumes a two-line representation of a stochastic number: one to represent the magnitude and one the sign. Here, we use the adder with unipolar format (plus fixed sign) and design parameter "threshold" 2, following the design methodology in [27], to compare it with the other adders. As shown in Fig. 12, it has lower accuracy than the proposed adder and has almost the same power consumption, with energy being its strong point according to Table II. Moreover, the adder in [21] occupies more area compared to the proposed one for register sizes $m = 2, 3$.

4) Adder in [22]: The adder (and subtracter by using a NOT gate in one of its inputs) in [22] uses an encoding of stochastic number into the ratio of the switching activities of two sequences. The four inputs of the adder are pairwise XNORed and then fed to a MUX that uses a modulus 1 counter as its select signal and it's output is the scaled result of the addition. To derive its nonscaling sum, two of the inputs and the output of an SNG are used as inputs to a three-input XNOR. The XNOR's output as well as the MUX form the adder's outputs, while their ratio yields the final (nonscaled) sum.

According to Table II, the overall hardware utilization is taxed due to the additional SNG, which impacts the power and energy consumption as well. Furthermore, to achieve comparable accuracy to that of the other adders, it requires large sequence lengths, as shown in Fig. 12. Its main advantage is that it can be used for both scaling or nonscaling additions and the ratio encoding can be used to directly implement other standard operations as well, e.g., multipliers and dividers. However, the incompatibility of the ratio encoding with other more popular SC encodings results in extra translation hardware complexity when is coupled with other traditional SC numerical architectures.

B. Stochastic Subtracters

The accuracy, the power × delay^2, and energy comparison between the stochastic subtracters are shown in Figs. 14 and Fig. 15, while detailed hardware utilization results are shown in Table II. Since the NOT gate does not degrade the accuracy of the computations for the proposed subtracter as well as the MUX and [22] architectures, the results are identical to the adder. Note that the area, power, and energy consumption of the proposed subtracter is almost the same as those of the proposed adder since the additional two NOT gates have minimal impact.

1) Subtracter in [23]: To realize the operation of subtraction hardware efficiently, the method in [23] correlates two input sequences, using the same LFSR for two different comparators in the SNG stage, and an XOR gate to provide the output sequence.

This architecture has an important key point; consider the following two cases: I) if the first SC operation is the subtraction, one can shape the architecture to effectively generate two signals from SNGs with one LFSR [23], [28] and 2) if on the other hand subtraction is an intermediate SC operation, to effectively use the XOR gate, the subtracter's input sequences must be regenerated in order to have high cross correlation.

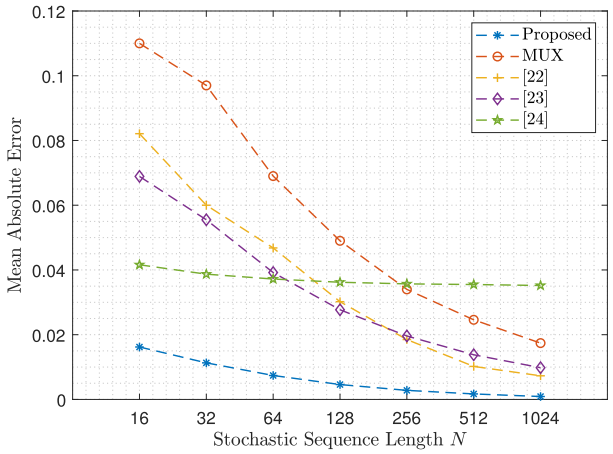

Fig. 14. Comparison of accuracy in MAE of stochastic subtracters for typical stochastic sequence lengths N.

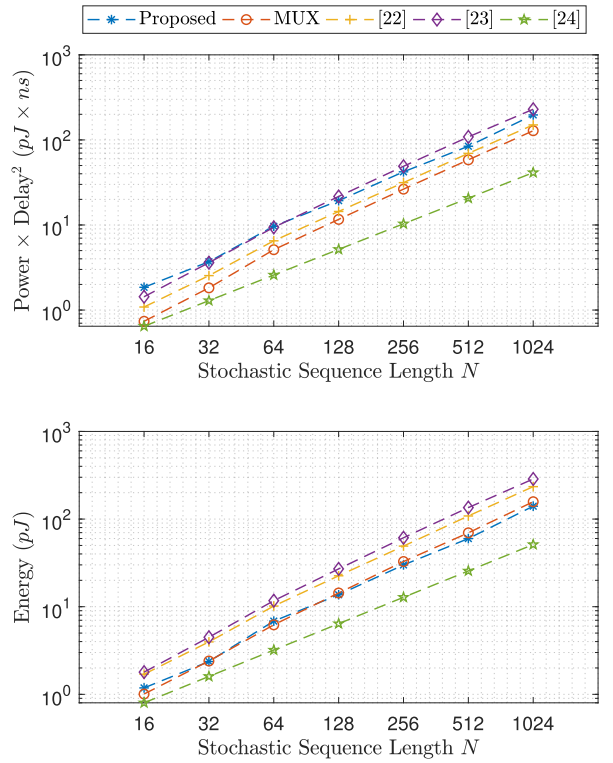

Fig. 15. Comparison of power × delay^2 (pJ × ns) (top) and energy (pJ) (bottom) consumption of stochastic subtracters for typical stochastic sequence lengths N.

Fig. 14 suggests that the work [23] achieves lower accuracy compared to the proposed stochastic subtracter, while it has the advantage of very low power and energy consumption when operating in case 1) above. When the subtraction is an intermediate operation, case 2), the proposed subtracter achieves better overall performance, as shown in Fig. 15 and Table II.

2) Subtracter in [24]: The subtracter in [24] uses an XNOR gate with inputs the two stochastic sequences, one of them inverted, to generate a rough estimate of the subtraction result and cascaded logic stages to improve its accuracy. The number of additional logic stages considered here is 3 but can be further expanded at the cost of additional hardware resources and delay. The proposed stochastic subtracter achieves better accuracy than the one in [24] as shown in Fig. 14, but the latter one has lower power and energy consumption according to Table II.

VII. Application in Digital Image Processing

To demonstrate the proposed adder's effectiveness in cascaded computations, we use it as a building block, along with AND gates for multiplication, to implement a convolution operation. Specifically, the convolution is used in a digital image processing task, which is the filtering of an entire image using a 3×3 box blur kernel. By adjusting appropriately kernel's weight values and by including nonlinear functions, this kernel can be used in NNs.

For the application, we select a grayscale image and represent each pixel with an 8-bit number as it is typically required in image processing. The pixels' and the kernel's values are normalized to range [0, 1] in order to be processed in the SC domain. For the stochastic number representation, we consider typical stochastic sequence lengths, namely $N = 2^k$, with $k = 4, \ldots, 10$ and investigate their effect in the accuracy of the computations. Then, the proposed as well as selected adders discussed in Section VI are used to realize the convolution operation and their performances are reported.

Among the selected adders, we excluded the standard MUX from comparisons as it requires large N lengths to achieve acceptable accuracy as shown in Fig. 12 implying also an increased hardware overhead. The same applies to the adder in [22] due to the fact that the two-sequence encoding of a stochastic number increases the design complexity in cascaded computations and each nonscaling adder is hardware demanding for moderate N lengths according to Table II.

The accuracy comparison of the convolutions based on the selected adders is shown in Table III evaluated with two metrics: 1) the peak signal-to-noise ratio (PSNR) and 2) the structural similarity index measure (SSIM). The first measures the absolute accuracy of computation and is one of the standard metrics used for images, whereas the second one measures the perceived quality of an image with values in [0, 1] (higher means better quality) [34]. In addition, a graphical representation of the computations using the proposed adder is shown in Fig. 16. Moreover, Table IV presents the corresponding hardware resources to realize the convolution kernel.

We note first that the register size used for the proposed adder is $m = 2$, as it does not degrade the result of calculations in this specific application. Moreover, the kernel by itself requires nine multipliers (AND gates) and eight stochastic adders, while its structure is adder tree based.

1) *Convolution Using Adders [19], [20]:* According to Table III, the two scaling adders [19], [20] provide acceptable accuracy and SSIM results when using more than $N = 256$ bit sequence lengths, which is due to sequence resolution drop by 2 after each addition. On the contrary, the convolution using the proposed stochastic adder achieves the same accuracy using only

Fig. 16. Image filtering using 3×3 convolution kernel. From left to right, (a) and (b) correspond to MATLAB and (c)–(i) correspond to the approach using the proposed adder with sequence lengths N. (a) Original image. (b) MATLAB's blur calculation. (c) $N = 16$. (d) $N = 32$. (e) $N = 64$. (f) $N = 128$. (g) $N = 256$. (h) $N = 512$. (i) $N = 1024$.

TABLE III
ACCURACY AND IMAGE QUALITY COMPARISON IN THE CALCULATION OF A 3×3 CONVOLUTION KERNEL USING THE PROPOSED AND STATE-OF-THE-ART STOCHASTIC ADDERS

	Peak-Signal-to-Noise Ratio (PSNR)						
$N = 2^k$	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Proposed	16.61	19.25	22.06	25.08	28.01	30.90	33.94
[20]	<10	13.56	17.91	21.95	25.68	28.67	31.92
[19]	<10	13.01	17.58	21.52	25.71	28.57	31.84
[21]	16.45	19.04	22.08	25.15	28.01	30.96	33.63
[25]	15.75	18.76	21.57	24.80	27.83	30.81	33.84
	Structural Similarity Index Measure (SSIM)						
Proposed	0.210	0.292	0.391	0.512	0.628	0.734	0.831
[20]	<0.2	<0.2	0.231	0.346	0.490	0.620	0.748
[19]	<0.2	<0.2	0.250	0.367	0.504	0.628	0.747
[21]	0.218	0.297	0.391	0.498	0.612	0.725	0.828
[25]	0.209	0.289	0.381	0.487	0.598	0.713	0.811

TABLE IV
COMPARISON OF HARDWARE REQUIREMENTS TO IMPLEMENT THE 3×3 CONVOLUTION KERNEL USING THE PROPOSED AND STATE-OF-THE-ART STOCHASTIC ADDERS IN AREA (μm^2), CRITICAL PATH (ns), POWER (mW) AND ENERGY (pJ) PER OPERATION

	Area (μm^2)	Power (mW)	Critical Path (ns)	Energy (pJ)
Proposed	554.63	0.390	1.6	0.624
[25] Binary Output	443.95	0.157	1.9	0.298
[25] Stochastic Output	661.71	0.254	2	0.509
[19]	207.57	0.104	1.5	0.156
[20]	456.62	0.322	1.6	0.515
[21]	952.20	0.492	1.5	0.738
Conventional Binary	9,017.13	2.440	7.5	18.30

half sequence length, e.g., $N = 128$. This leads to less energy per convolution for the proposed compared to the convolution using [20] due to different lengths required. The convolution is using [19], however, despite the large N value, e.g., 256, and it also achieves good power and energy efficiency. A further advantage of the proposed adder is that it benefits the operations that require nonscaled computations after the convolution stage, whereas adders [19], [20] face upscaling followed by sequence regeneration design challenges.

2) *Convolution Using Adder [21]:* Due to its nonscaling nature, the convolution kernel using the adder in [21] achieves the same accuracy as the proposed approach according to Table III. However, it also has slightly increased energy consumption, making the proposed one more efficient for multiple nonscaling additions.
3) *Convolution Using Adder [25]:* The accumulative parallel counter (APC) is a popular adder capable of adding single-bit sequences in parallel producing binary output and it is used in the SC literature [13], [26] as it benefits multiply-and-accumulate stages. Compared to the proposed approach, it achieves almost the same performance in terms of accuracy. Note, however, that if the convolution is the final operation, i.e., no further operations are required, APC is effective terms of hardware. Otherwise, it requires more area than the proposed approach due to the required binary-to-stochastic converter to rerandomize the output for further computations, e.g., nonlinear functions.
4) *Convolution With Conventional Binary:* Compared to the conventional arithmetic architectures, the proposed approach requires negligible area, which is its strong point and is approximately $\times 16$ less according to Table IV. On the other hand, given the moderate number clock cycles to achieve acceptable results, for instance $N = 64$, its energy consumption is higher, namely 21 pJ.

VIII. CONCLUSION

Two SC nonscaling architectures for addition and subtraction were presented. Their operation principle was proved using detailed mathematical analysis based on MC modeling and an analytical methodology for selecting their register size was introduced. The two architectures have the advantage of not requiring any randomizing source and they operate with standard SC encoding making their use in hardware for complex calculations easy. Compared to existing architectures in the SC literature, it was shown that they achieve good latency versus accuracy tradeoff, which is a bottleneck in SC, allowing both fast and precise computations. Finally, the implementation of a 3×3 convolution kernel using the proposed adder demonstrated its advantages.

REFERENCES

[1] B. R. Gaines, *Stochastic Computing Systems*. Boston, MA, USA: Springer, 1967.
[2] S. R. Faraji and K. Bazargan, "Hybrid binary–unary hardware accelerator," *IEEE Trans. Comput.*, vol. 69, no. 9, pp. 1308–1319, Sep. 2020.
[3] A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1515–1531, Aug. 2018.
[4] P. Li and D. J. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *Proc. IEEE 29th Int. Conf. Comput. Design (ICCD)*, Oct. 2011, pp. 154–161.

[5] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 3, pp. 449–462, Mar. 2014.

[6] A. Alaghi, C. Li, and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2013, pp. 1–6.

[7] A. Morro *et al.*, "A stochastic spiking neural network for virtual screening," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 1371–1375, Apr. 2018.

[8] B. D. Brown and H. C. Card, "Stochastic neural computation. I. Computational elements," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, Sep. 2001.

[9] B. D. Brown and H. C. Card, "Stochastic neural computation. II. Soft competitive learning," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 906–920, Sep. 2001.

[10] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "VLSI implementation of deep neural network using integral stochastic computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2688–2699, Oct. 2017.

[11] S. Liu, H. Jiang, L. Liu, and J. Han, "Gradient descent using stochastic circuits for efficient training of learning machines," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2530–2541, Nov. 2018.

[12] Y. Liu, L. Liu, F. Lombardi, and J. Han, "An energy-efficient and noise-tolerant recurrent neural network using stochastic computing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 9, pp. 2213–2221, Sep. 2019.

[13] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 2809–2824, Aug. 2020.

[14] Y. Liu and K. K. Parhi, "Architectures for recursive digital filters using stochastic computing," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3705–3718, Jul. 2016.

[15] N. Saraf, K. Bazargan, D. J. Lilja, and M. D. Riedel, "IIR filters using stochastic arithmetic," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6.

[16] W. J. Gross and V. C. Gaudet, *Stochastic Computing: Techniques and Applications*. Cham, Switzerland: Springer, 2019.

[17] S. S. Tehrani, W. J. Gross, and S. Mannor, "Stochastic decoding of LDPC codes," *IEEE Commun. Lett.*, vol. 10, no. 10, pp. 716–718, Oct. 2006.

[18] M. H. Najafi, D. Jenson, D. J. Lilja, and M. D. Riedel, "Performing stochastic computation deterministically," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 12, pp. 2925–2938, Dec. 2019.

[19] 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 *Proc. ACM Proc. Design, Autom. Test Eur.*, Laussane, Switzerland, Mar. 2017, pp. 13–18.

[20] P. Ting and J. P. Hayes, "Eliminating a hidden error source in stochastic circuits," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2017, pp. 1–6.

[21] A. Ren *et al.*, "SC-DCNN: Highly-scalable deep convolutional neural network using stochastic computing," in *Proc. ACM 22nd Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, Xi'an, China, Apr. 2017, pp. 405–418.

[22] V. Canals, A. Morro, A. Oliver, M. L. Alomar, and J. L. Rossello, "A new stochastic computing methodology for efficient neural network implementation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 551–564, Mar. 2016.

[23] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Asheville, NC, USA, Oct. 2013, pp. 39–46.

[24] Y. Liu and K. K. Parhi, "Computing polynomials using unipolar stochastic logic," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, pp. 1–30, May 2017.

[25] B. Parhami and C.-H. Yeh, "Accumulative parallel counters," in *Proc. Conf. Rec. 29th Asilomar Conf. Signals, Syst. Comput.*, Nov. 1995, pp. 966–970.

[26] P.-S. Ting and J. P. Hayes, "Stochastic logic realization of matrix operations," in *Proc. 17th Euromicro Conf. Digit. Syst. Design*, Aug. 2014, pp. 356–364.

[27] B. Yuan, Y. Wang, and Z. Wang, "Area-efficient scaling-free DFT/FFT design using stochastic computing," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 12, pp. 1131–1135, Dec. 2016.

[28] V. T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1417–1422.

[29] A. Alaghi and J. P. Hayes, "Fast and accurate computation using stochastic circuits," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–4.

[30] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 1990.

[31] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed. Providence, RI, USA: American Mathematical Society, 1997.

[32] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, 1st ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.

[33] J. E. Stine *et al.*, "FreePDK: An open-source variation-aware design kit," in *Proc. IEEE Int. Conf. Microelectron. Syst. Educ. (MSE)*, San Diego, CA, USA, 2007.

[34] Z. Wang and A. C. Bovik, *Modern Image Quality Assessment*, 1st ed. San Rafael, CA, USA: Morgan and Claypool, 2006.

Nikos Temenos (Student Member, IEEE) received the B.Sc. degree in computer and systems engineering from the Piraeus University of Applied Sciences, Aigaleo, Greece, in 2015, and the M.Sc. degree in microelectronics from the National and Kapodistrian University of Athens, Athens, Greece, in 2017. He is currently working toward the Ph.D. degree at the National Technical University of Athens, Athens.

He has authored or coauthored several IEEE conferences. His main research area includes digital VLSI design, computer arithmetic, as well as algorithms and architectures for stochastic computing targeting field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) technologies.

Mr. Temenos is a regular reviewer for many IEEE transactions and conferences.

Paul P. Sotiriadis (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece, in 1994, the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1996, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2002.

In 2002, he joined Johns Hopkins University, Baltimore, MD, USA, as an Assistant Professor of electrical and computer engineering. He is currently a Faculty Member with the Department of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece. He is also the Director of the Electronics Laboratory, National Technical University of Athens, and a member of the Governing Board of the Hellenic Space Center, the National Space Center of Greece, Chalandri, Greece. In 2012, he joined the faculty of the Department of Electrical and Computer Engineering, National Technical University of Athens. He has authored or coauthored more than 170 research publications, most of them in IEEE journals and conferences, holds one patent, and has contributed chapters to technical books. His research interests include design, optimization, and mathematical modeling of analog and mixed-signal circuits, RF and microwave circuits, advanced frequency synthesis, biomedical instrumentation, and applications of machine learning in application-specific integrated circuit (ASIC) optimization as well hardware realizations of machine learning (ML) algorithms. He has led several projects in these fields funded by U.S. organizations and has collaborations with industry and national labs.

Dr. Sotiriadis has been a member of technical committees of many conferences. He regularly reviews for many IEEE transactions and conferences and serves on proposal review panels. He received several awards, including the 2012 Guillemin-Cauer Award from the IEEE Circuits and Systems Society, the Best Paper Award at the IEEE International Symposium on Circuits and Systems in 2007, the Best Paper Award at the IEEE International Frequency Control Symposium in 2012, the Best Paper Award at the IEEE International Conference on Modern Circuits and Systems Technologies in 2019, and the Best Paper Award at the IEEE International Conference on Microelectronics in 2021. He has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS from 2016 to 2020 and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS from 2005 to 2010. He is also an Associate Editor of the IEEE SENSORS JOURNAL.