A Markov Chain Framework for Modeling the Statistical Properties of Stochastic Computing Finite-State Machines

Nikos Temenos and Paul P. Sotiriadis
Department of Electrical and Computer Engineering
National Technical University of Athens, Greece
E-mail: ntemenos@gmail.com

***Abstract*—A general methodology to derive analytically the statistical properties of Stochastic Computing Finite-State Machines (SFSM) is introduced. The SFSMs, expressed as Moore ones, are modeled using Markov Chains, enabling the derivation in closed form of their output sequences' statistical properties, including their expected value, their auto- & cross-correlation, their auto- & cross-covariance, their variance and standard deviation as well as their mean squared error. A MC overflow/underflow probability model accompanies the methodology, allowing to calculate analytically the expected number of steps before overflows/underflows, setting the guidelines to select the register's size that reduces erroneous bits originating from them. In the proposed methodology both the input sequence length and the number of the SFSMs' states are considered as parameters, accelerating the overall design procedure as the necessity for multiple time-consuming numerical simulations is eliminated. The proposed methodology's accurate modeling capabilities are demonstrated with its application in two SFSMs selected from the SC literature, while comparisons with the numerical experiments justify its correctness.**

***Index Terms*—Stochastic Computing, Stochastic FSMs, Markov Chain Modeling**

I. Introduction

Arithmetic operations on binary stochastic sequences is the basis of the unconventional technique known as Stochastic Computing (SC) [1]. Its encoding and processing of information deviates from the classical binary arithmetic one; SC processes binary-valued numbers in the form of sequences of logic 1s and 0s [1], [2]. Hence, SC's bit-serial processing allows for operations to be realized simply using standard logic gates and cells [2]–[7], while its probabilistic nature makes it tolerant to soft-errors originating from noisy sources [2], [3].

SC's advantages favor applications requiring massive parallelism and hardware-friendly realizations. Neural Networks [8]–[15] and Digital Image Processing [16]–[18] are the two primary benefited fields of application due to the necessity of non-linear elements in their Digital Signal Processing (DSP) cores, besides the fundamental arithmetic operations [8], [10], [16]. These non-linear elements are highly-complex functions and in contrast to the standard binary arithmetic, their realization is done efficiently in SC with the use of Stochastic Finite-State Machines (SFSMs) [10], [12], [17], [19], [20].

The concept of using SFSMs to approximate non-linear functions such as the *tanh*, the *exponential* etc. was introduced in [10]. For the approximations to be feasible, the SFSMs should satisfy the following conditions according to [10]: 1) they have a finite number of ordered states with the first and last one being saturating, meaning that they cannot be exceeded; 2) the transitions within their states are driven by input sequences, with stochastic properties and finite length; and 3) each state communicates with the rest ones. These conditions allow for the operation of a SFSM to be described as an ergodic Markov Chain (MC), enabling the synthesis of functions based on simple logical operations between the states' probabilities [10].

Despite the SFSMs' multiple advantages, they also come with their own weaknesses [10]. In [10], it is mentioned that SFSMs introduce correlations among the bits of the output sequence, which is reasonable given the memory elements required to implement the state machines [10]. However, the calculation of the output's auto-correlation is estimated with numerical experiments [10]. This is also the case for the SFSM's output that approximates the given function, in which, two important factors contribute as well: 1) the number of states and 2) the input sequence length.

The SFSM analysis of [10], is further extended in [17]. Specifically, in [17], MCs are used to formally prove the principle of operation of several SC-based non-linear functions, including the *exponential*, the *tanh* etc. [17]. A fault-tolerance analysis with respect to bit-flips is also considered in [17]. Nevertheless, the SFSM's statistical properties are not investigated.

Stochastic sequence correlation is often caused at the input as discussed in [2], [21]–[23]; the binary-to-stochastic number converters share a common random number source. This allows for certain arithmetic operations to be realized more efficiently as shown in [2], [21]–[23], at the cost of increased correlation between the input and the output sequences. The use of a de-correlator unit composed of D Flip-Flops to reduce correlations is mentioned in [22], but, the analysis is supported by numerical experiments.

With respect to the SFSM's output auto-correlation, it is only investigated in [24]. Its calculation, however, faces modeling difficulties when joint distributions are required and thus it is limited to approximations [24]. The variance in multi-stage SC circuits is analyzed in [25]. Yet, it is approached from a gate-level perspective, without further investigation in

SFSMs.

Motivated by the needs for an in-depth understanding of the SFSMs' statistical properties, in this work we introduce a mathematical framework for their detailed analysis based on MCs. It is a general methodology, in the sense that it can be applied to any SFSM modeled as a MC. The major contribution of this manuscript is the analytical calculation using closed-formed expressions of the following quantities in a SFSM:

- The expected value of the output and the output's mean.
- The auto-correlation and auto-covariance of the output.
- The cross-correlation and cross-covariance of the output with the inputs.
- The variance & the standard deviation of the output's mean.
- The mean squared error of the output's mean.
- The probability of overflows and underflows in the saturating states.
- The expected number of steps before overflows and underflows, used to select the number of states of the SFSM balancing the computational accuracy hardware trade-off.

Once applied to a SFSM, the proposed framework can be an effective tool to: 1) evaluate the correctness of the SFSM's output when approximating a given function; 2) measure the correlation among the bits in the output sequence and to what extent it affects further operations (e.g. multiplication) of the output with itself and the inputs; 3) calculate the expected accuracy of the SFSM's output and to compare it with the experimental numerical results; and 4) select the register's size that balances computational accuracy compared to hardware resources. A further advantage of the proposed framework is that it considers as parameters both the input sequence length and the number of states, which is of utter importance for the modeling of SFSMs; it eliminates the necessity for multiple time-consuming parametric simulations to derive the statistical properties and the register's size, thereby accelerating their design & modeling procedure.

We organize the remainder of the proposed work as follows. In Section II we provide the essential notation of the stochastic numbers, as well as explain the modeling procedure of SFSMs using Markov Chains. In Section III, we introduce the proposed framework based on MCs and explain in-detail the methodology to derive the statistical properties of SFSMs. In Section IV, we investigate the overflow/underflow occurrence based on the number of the SFSM's states and provide guidelines to select the number of states. In Section V, we use the proposed framework to model in detail the statistical properties of two SFSMs selected from the SC literature and we showcase its correctness by comparing its results with those obtained from numerical experiments. Finally, in Section VI we conclude the introduced framework.

II. Stochastic Computing Finite-State Machines

In this section we provide the basic notation & definitions used for the stochastic numbers and then we proceed with the modeling of SFSMs as Markov Chains.

A. Stochastic Number Properties

The standard way to encode binary numbers into stochastic ones is by using a stochastic number generator (SNG). The SNG utilizes a pseudo-random number generator, which produces a sequence of independent and identically distributed (i.i.d.) pseudo-random numbers in $[0,1]$ and compares them sequentially to the input c-bit binary word $b \in [0,1]$ [26]. The comparator outputs a logic 1 if the binary word is larger and 0 otherwise. The bit generation process is completed after $N = 2^c$ clock cycles setting the length of the generated stochastic sequence.

The generated N-bit sequence $X_n, n = 1, 2, \ldots, N$, where n is the time (clock cycle), is such that $X \triangleq P_r(X_n = 1)$ in the range $[0,1]$, known as *unipolar format*, with realization

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

Negative numbers are also supported in SC, with their representation known as *bipolar format*. The encoding from unipolar to bipolar, is done using the transformation $X \mapsto 2X - 1$, expanding the range of the stochastic number to $[-1,1]$.

B. Operation of a Stochastic Computing Processing Block

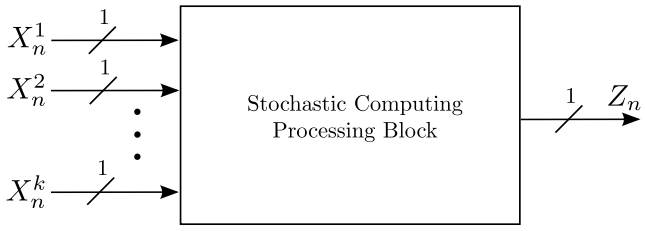

Fig. 1. A multi-input single-output stochastic computing processing block.

From a system-level perspective, a SC processing block (SCPB) is represented by the abstract model of Fig. 1. Typically in SC, it can describe the operation of

1) a combinational logic expression,
2) a sequential logic expression,
3) a higher-level architecture, containing both of the previous processing elements.

Therefore, a SCPB can have many stochastic input sequences $\{X_n^j\}, j = 1, \ldots, k$, each one with probability $X^j = P_r(X_n^j = 1)$, while $\{Z_n\}$ is the output sequence.

The realization of sequential logic circuits and high-level architectures requires memory elements. This means that the SCPB must have a set of internal states $\mathcal{T}_R \triangleq \{0, 1, 2, ..., W-1\}$, where W is the number of states. When counters are used in SC, it is important to note that they may saturate. Assume for example that the states are linearly ordered, i.e. $0 < 1 < 2 <, \ldots, < W-1$ and the goal of the SCPB is to capture an operation of the form $T_n = T_{n-1} + f(X_n^1, \ldots, X_n^k)$, where T_n is the current state. State T_n is constrained in $\mathcal{T}_R$, i.e., within 0 and $W-1$ and what is (typically) realized by the SCPB is the state update process

$$T_n = \max\left\{\min\left\{T_{n-1} + f(X_n^1, \ldots, X_n^k), W-1\right\}, 0\right\}. \tag{2}$$

To provide with better insight on the state update, we proceed with the following example. Assume 3 i.i.d. input sequences $\{X_n^1\}, \{X_n^2\}, \{X_n^3\}$ and the function

$$f(X_n^1, X_n^2, X_n^3) = \text{AND}\left(X_n^1, X_n^2, X_n^3\right) - \text{AND}\left(\overline{X_n^1}, \overline{X_n^2}, \overline{X_n^3}\right)$$

where $\overline{X_n^j} = 1 - X_n^j$. Considering the above, T_n increases its value by 1-bit when and only when all three inputs are simultaneously 1, i.e. $X_n^1 = X_n^2 = X_n^3 = 1$, decreases its value by 1-bit when $X_n^1 = X_n^2 = X_n^3 = 0$ and maintains its previous value otherwise, i.e. $T_n = T_{n-1}$.

The purpose of the counter's register in the previous example, is to remember the cases where all three inputs are 1, so as to "balance" them with the cases where all three inputs are 0.

With respect to the output, Z_n is determined according to the SCPB's operation. In the simplest case of combinational logic, Z_n is straightforward. However, in the case where the SCPB describes a sequential logic circuit or a higher-level architecture, then FSMs are utilized. Therefore, the SCPB's operation can be described using a stochastic FSM (SFSM) and consequently be modeled as a Markov Chain (MC), allowing for the exploration of its long-term stochastic dynamics and the calculation of its statistical properties.

C. Markov Chain Modeling of a Stochastic FSM

A SFSM expresses a behavior that falls into the category of either a Mealy or a Moore FSM. The former implies that the current output Z_n is a function of the inputs and the state, whereas the latter implies that Z_n is determined solely by the current state. Although the conversion from one FSM behavior to another is a feasible and standard task [27], as shown with the example in Fig. 2, here we consider only Moore-based FSMs. This is because relating the current state to the output only, makes the mathematical modeling, analysis and design of SFSMs using MCs more tractable.

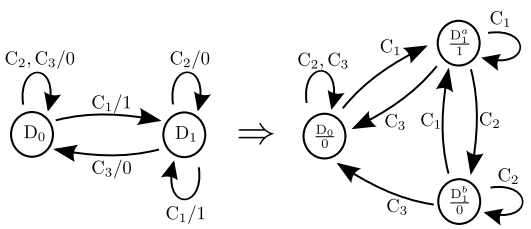

Fig. 2. Conversion example of a stochastic Mealy (left) to Moore (right) FSM. State D_1 in the Mealy is separated into two states in the Moore D_1^a, D_1^b outputting 1 and 0 respectively. In this example, transition probabilities C_1, C_2, C_3, are arbitrary selected, but, determined by two stochastic input sequences $\{X_n^1\}$,$\{X_n^2\}$.

A SFSM can be described by a MC model, with an example shown in Fig. 3. The MC of Fig. 3 is used as reference to explain the modeling procedure of a SFSM, but, note that *any* MC can be used. The MC of Fig. 3 has a total of M states and its current state, S_n, takes values within the set

$$\mathcal{S} \triangleq \{0, 1, 2, \ldots, M-2, M-1\}. \tag{3}$$

Therefore, the MC's current state S_n, is described as a function of the previous state and the inputs, i.e. $S_n = F(S_{n-1}, X_n^j)$ and thus the output is a function of the state, i.e. $Z_n = G(S_n)$.

Considering that a counter is used as memory element, there is a difference between the counter's total number of states, W, and the MC's number of states, M; the MC has *at least* as many states as the counter has, i.e. $M \geq W$, meaning that the mapping from the MC states to those of the counter is surjective, but, not necessarily injective. This can be based on many factors, such as the conversion from a Mealy SFSM behavior to a Moore one, the counter's register type, for instance a shift-register, the SCPB's number of inputs etc.

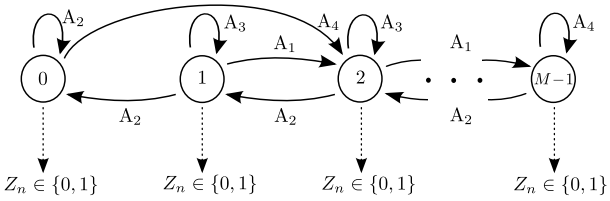

Fig. 3. Example of a Markov Chain model describing the operation of a stochastic FSM. Transition probabilities A_j are defined by a boolean function and determine the state's transition (see example below). The output Z_n is related to the current state, expressing the FSM's behavior as a Moore one, outputting 0 or 1.

Proceeding to the MC's behavior and assuming that transitions occur from a state σ_i to any other one σ_j, with $\sigma_i, \sigma_j \in \mathcal{S}$, then the $(M \times M)$ transition probability matrix is defined as $H \triangleq [P_r(S_n = \sigma_j | S_{n-1} = \sigma_i)]$. Considering that the transitions from one state σ_i to another σ_j are determined by the inputs $X_n^1, X_n^2, \ldots, X_n^k$ of the SCPB, then the transition probabilities $A_j, j = 1, \ldots, l$ could be any boolean function, such as AND, OR, XOR etc, as

$$A_j = P_r\left(f_j(X_n^1, X_n^2, \ldots, X_n^k)\right). \tag{4}$$

To further explain how A_j are determined, consider the following example. Suppose that the MC's state S_{n-1} at time index $n-1$, transitions as follows

- If $X_n^1 = X_n^2 = 1$ and $S_{n-1} > 0$, then $S_n = S_{n-1} + 1$,
- If $X_n^1 = X_n^2 = 0$ and $S_{n-1} > 0$, then $S_n = S_{n-1} - 1$,
- If $\text{XOR}(X_n^1, X_n^2) = 1$ and $S_{n-1} > 0$, then $S_n = S_{n-1}$,
- If $\text{OR}(X_n^1, X_n^2)$=1 and $S_{n-1} = 0$, then $S_n = S_{n-1} + 2$,
- If $\text{OR}(X_n^1, X_n^2)$=1 and $S_{n-1} = M-1$, then $S_n = S_{n-1}$.

Based on the above, the transition probabilities can be described as $A_1 = P_r(\text{AND}(X_n^1, X_n^2) = 1)$, $A_2 = P_r(\text{NOR}(X_n^1, X_n^2) = 1)$ and $A_3 = P_r(\text{XOR}(X_n^1, X_n^2) = 1)$. They can be used along with the MC model of Fig. 3 and the state ordering $(0, 1, \ldots, M-1)$, to express H as in (5), where $A_4 = A_1 + A_3$.

$$H = \begin{bmatrix} A_2 & 0 & A_4 & \ldots & \ldots & 0 \\ A_2 & A_3 & A_1 & 0 & \ldots & 0 \\ 0 & A_2 & A_3 & A_1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & A_2 & A_3 & A_1 \\ 0 & \ldots & \ldots & 0 & A_2 & A_4 \end{bmatrix}. \tag{5}$$

Note that since H is stochastic, it satisfies $\sum_{j=1}^{M} H_{i,j} = 1$, where (i, j) represents the i-th row and j-th column of the matrix H. The probability distribution vector of state S_n, is defined as

$$p_n^T \triangleq \begin{bmatrix} P_r(S_n = 0) \\ P_r(S_n = 1) \\ P_r(S_n = 2) \\ \vdots \\ P_r(S_n = M-1) \end{bmatrix} \in [0,1]^M \tag{6}$$

and for $n = 1, 2, \ldots, N$ steps it is derived as

$$p_n = p_0 H^n \in [0,1]^M. \tag{7}$$

Here, p_0 denotes the initial distribution vector representing the starting state of the MC, S_0, which can take any value within $\mathcal{S}$. It is expressed as

$$p_0 = e_i \in [0,1]^M, \tag{8}$$

where $e_i = [0, \ldots, 1, \ldots, 0] \in \mathbb{R}^M$ is the i-th standard vector.

Before we proceed with the analysis in the next section, it is important to note that we consider only MCs that are *irreducible*; they have the property that starting from any state σ_i, it is possible to transition to any other one σ_j, regardless of the number of transition steps. Such MCs have a unique stationary distribution p_F [28], i.e. $p_n \to p_F$ as $n \to \infty$, where also $p_F H = p_F$

III. Statistical Modeling of Stochastic FSMs

In this section, we use the MC modeling to derive analytically the statistical properties of SFSMs.

A. Expected Value

To derive the first-moment statistics, one can observe first from the MC model of Fig. 3, that Z_n is related to the state only; each state outputs either 0 or 1, based on the SFSM's operation. It is convenient therefore, to partition $\mathcal{S}$ into two subsets $\mathcal{S}_1$ and $\mathcal{S}_0$, such that $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_0$, $\mathcal{S}_1 \cap \mathcal{S}_0 = \{\}$, where $S_n \in \mathcal{S}_1 \Rightarrow Z_n = 1$ and $S_n \in \mathcal{S}_0 \Rightarrow Z_n = 0$.

Considering the above and also the equations describing the MC (5), (7) and (8), the expected value of the instantaneous output Z_n is calculated as

$$\mathbb{E}[Z_n] = P_r(Z_n = 1) = P_r\left(S_n \in \mathcal{S}_1\right) = p_0 H^n q^T, \tag{9}$$

with $q \in \mathbb{R}^M$ defined as

$$q \triangleq \sum_{i \in \mathcal{S}_1} e_i, \tag{10}$$

where q represents the set of states outputting 1. To give a better intuition behind the calculation of (9) and the definition of q in (10), suppose that the states 0 and 1 are the only ones outputting 1. Then $\mathcal{S}$ is partitioned into $\mathcal{S}_1 = \{0, 1\}$ and $\mathcal{S}_0 = \{2, \ldots, M-1\}$ and thus $q = [1, 1, 0, \ldots, 0]$.

According to (1), the average of the N-bit output sequence is

$$\widetilde{Z}_N = \frac{1}{N}\Big(Z_1 + Z_2 + \cdots + Z_N\Big), \tag{11}$$

and using (9) its expected value is calculated as

$$\mathbb{E}[\widetilde{Z}_N] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[Z_n] = \frac{1}{N} p_0 \left(\sum_{n=1}^{N} H^n\right) q^T. \tag{12}$$

Both $\mathbb{E}[Z_n]$ and $\mathbb{E}[\widetilde{Z}_N]$ are also essential in the calculation of the second-moment statistics in the following subsection.

B. Auto-Correlation & Auto-Covariance

The auto-correlation of the output $\{Z_n\}$ for time lag $r \geq 0$ is

$$\begin{aligned} R_Z(n+r, n) \triangleq& \mathbb{E}[Z_{n+r} Z_n] = P_r(Z_{n+r} = 1, Z_n = 1) \\ =& \sum_{j_1, j_2 \in \mathcal{S}_1} P_r(S_{n+r} = j_2, S_n = j_1) \\ =& \sum_{j_1, j_2 \in \mathcal{S}_1} P_r(S_n = j_1) P_r(S_{n+r} = j_2 | S_n = j_1) \\ =& \sum_{j_1, j_2 \in \mathcal{S}_1} (p_0 H^n e_{j_1}^T)(e_{j_1} H^r e_{j_2}^T) \\ =& p_0 H^n Q H^r q^T, \end{aligned} \tag{13}$$

where q is given by (10) and $Q \in \mathbb{R}^{M \times M}$ is

$$Q \triangleq \sum_{j_1 \in \mathcal{S}_1} e_{j_1}^T e_{j_1} = \text{diag}(q). \tag{14}$$

The auto-covariance of the output $\{Z_n\}$ is calculated using (9) and (13) as

$$\begin{aligned} C_Z(n+r, n) \triangleq& \mathbb{E}\big[(Z_{n+r} - \mathbb{E}[Z_{n+r}])(Z_n - \mathbb{E}[Z_n])\big] \\ =& R_Z(n+r, n) - \mathbb{E}[Z_{n+r}]\mathbb{E}[Z_n] \\ =& p_0 H^n Q H^r q^T - p_0 H^{n+r} q^T p_0 H^n q^T. \end{aligned} \tag{15}$$

C. Cross-Correlation & Cross-Covariance

We recall that Z_n and S_n depend only on $\{X_n^j\}$, $j = 1, 2, \ldots, k$ and not on their future values. Moreover, the random variables of the input sequences $\{X_n^j\}$ are assumed to be independent to each other, since they originate from different random number sources. To this end, we derive the cross-correlation of the output $\{Z_n\}$ with the a single input $\{X_n\}$ as

$$R_{ZX}(n, n+r) \triangleq \mathbb{E}[Z_n X_{n+r}] = P_r(Z_n = 1, X_{n+r} = 1) \tag{16}$$

To proceed further, we distinguish cases for r,

- For $r = 0$,

$$\begin{aligned} R_{ZX}(n, n) =& P_r(Z_n = 1, X_n = 1) \\ =& \sum_{\sigma \in \mathcal{S}} P_r(Z_n = 1, X_n = 1, S_{n-1} = \sigma) \\ =& \sum_{\substack{\sigma \in \mathcal{S}, \\ \sigma_1 \in \mathcal{S}_1}} P_r(S_n = \sigma_1, X_n = 1, S_{n-1} = \sigma) \\ =& \sum_{\substack{\sigma \in \mathcal{S}, \\ \sigma_1 \in \mathcal{S}_1}} P_r(S_n = \sigma_1 \mid X_n = 1, S_{n-1} = \sigma) \\ & P_r(X_n = 1) P_r(S_{n-1} = \sigma) \\ =& p_0 H^{n-1} (H \circ V) q^T P_r(X_n = 1), \end{aligned} \tag{17}$$

where matrix $H \circ V$ is the point-wise (Hadamard) product of H with V, where $V \in \{0,1\}^{M\times M}$ is such that $v_i, j = 1$ if and only if the transition from the i-th to the j-th state is done with $X_n = 1$.

- For $r \geq 1$, since Z_n and X_{n+r} are independent we have

$$\begin{aligned} R_{ZX}(n, n+r) =& P_r(Z_n = 1)P_r(X_{n+r} = 1) \\ =& p_0 H^n q^T P_r(X_{n+r} = 1). \end{aligned} \tag{18}$$

Summarizing,

$$R_{ZX}(n, n+r) = \begin{cases} p_0 H^{n-1}(H \circ V)q^T P_r(X_n = 1), & r = 0 \\ p_0 H^n q^T P_r(X_{n+r} = 1), & r > 0 \end{cases} \tag{19}$$

The cross-covariance between the output $\{Z_n\}$ and the input $\{X_n\}$ sequences

$$C_{ZX}(n, n+r) = R_{ZX}(n, n+r) - \mathbb{E}[Z_n]\mathbb{E}[X_{n+r}], \tag{20}$$

is derived directly from (9) and (19) and the definition $X = P_r(X_n = 1)$ giving

$$C_{ZX}(n, n+r) = \begin{cases} X p_0 H^{n-1}(H \circ V - H)q^T, & r = 0 \\ 0, & r > 0 \end{cases} \tag{21}$$

D. Variance and Standard Deviation

The variance of $\tilde{Z}_N$ from (11) is calculated using the expression (15) as follows

$$\begin{aligned} \text{Var}(\tilde{Z}_N) =& \mathbb{E}\big[(\tilde{Z}_N - \mathbb{E}[\tilde{Z}_N])^2\big] \\ =& \frac{1}{N^2} \sum_{i,j=1}^{N} \mathbb{E}\big[(Z_i - \mathbb{E}[Z_i])(Z_j - \mathbb{E}[Z_j])\big] \\ =& \frac{1}{N^2} \sum_{i,j=1}^{N} C_Z(i,j) \\ =& \frac{1}{N^2} \left[\sum_{i=1}^{N} C_Z(i,i) + 2 \sum_{i>j}^{N} C_Z(i,j) \right] \\ =& \frac{1}{N^2} \Bigg[p_0 \sum_{i=1}^{N} H^i Q q^T - \sum_{i=1}^{N} \left(p_0 H^i q^T\right)^2 \\ & + 2 \Bigg(\sum_{j=1}^{N-1} \sum_{i=j+1}^{N} p_0 H^j Q H^{(i-j)} q^T \\ & - \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \left(p_0 H^i q^T\right)\left(p_0 H^j q^T\right) \Bigg) \Bigg], \end{aligned} \tag{22}$$

while the standard deviation is obtained as $\sigma_{\tilde{Z}_N} = \sqrt{\text{Var}(\tilde{Z}_N)}$.

E. Error Analysis

To investigate the output accuracy of a SFSM, one can calculate analytically the Mean Squared Error (MSE) between the output's mean value $\tilde{Z}_N$ and the actual value of the computation $\bar{Z}$. It is calculated as

$$\begin{aligned} \text{MSE}(\tilde{Z}_N) &= \mathbb{E}\left[(\tilde{Z}_N - \bar{Z})^2\right] \\ &= \mathbb{E}\left[\tilde{Z}_N^2 - 2\tilde{Z}_N\bar{Z} + \bar{Z}^2\right] \\ &= \mathbb{E}\left[\tilde{Z}_N^2\right] - 2\mathbb{E}[\tilde{Z}_N]\bar{Z} + \bar{Z}^2 \\ &= \text{Var}(\tilde{Z}_N) + \mathbb{E}[\tilde{Z}_N]^2 - 2\mathbb{E}[\tilde{Z}_N]\bar{Z} + \bar{Z}^2, \end{aligned} \tag{23}$$

where the analytical expressions (12) and (22) are used.

IV. Register's Number of States Selection

The registers utilized by the SCPBs are typically used to store and "remember" logic 1s based upon a counting process. Ideally, with finite input sequence length and infinite number of states, the counting is performed perfectly, i.e. without loss of 1s. In practice, however, this is not feasible given the register's finite number of states; if they are too few, the counting process results in overflows or underflows that may degrade the output's accuracy.

Consider the following scenario: starting from any initial state of the register, e.g. $T_0 \in \mathcal{T}_R$, the SCPB's inputs are such that they force T_n to perform a walk within states $0, \ldots, W-1$. Eventually, T_n will reach either of its saturating states $W-1$ or 0 and may visit them repeatedly. This can cause overflows or underflows given that states $W-1$ and 0 cannot be exceeded to allow for further counting and correctly storing of logic 1s. Therefore, it is important to investigate how the number of states W are related to overflows/underflows and when this impacts the accuracy of the output sequence.

A. SFSM Overflow/Underflow Modeling

To explain the modeling procedure of overflows/underflows, consider the MC of Fig. 3 and suppose that its current state S_n is $M-1$ (or 0). The *overflows/underflows* occur when and only when the next combination of inputs at time index $n+1$, force the MC's state to return to itself, i.e. $S_{n+1} = S_n$, where it should transition to $S_{n+1} = S_n+1$ or $S_{n+1} = S_n-1$ instead. However, states M and -1 do not exist and as expected, the MC of Fig. 3 does not allow for overflows/underflows to be modeled. Therefore, we modify it to the one shown in Fig. 4 which contains two extra *absorbing* states M_a, M_b so as to capture the overflows/underflows. Note that both states M_a, M_b are used for modeling purposes only and do not imply any change of the register's states or size.

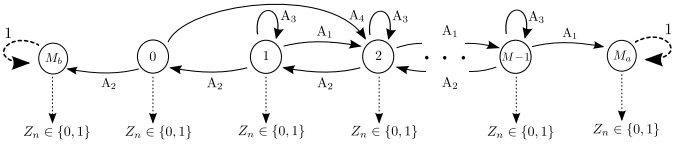

Fig. 4. Example of the Markov Chain overflow/underflow model with absorbing states M_a, M_b corresponding to that of Fig. 3.

Based on the MC model of Fig. 4 one can calculate the *probability* of overflows/underflows in states M_a, M_b. First, the set of the MC's states is defined as $\hat{\mathcal{S}} \triangleq$

$\{0, 1, 2, ..., M-1, M_a, M_b\}$ and assuming a state order $(0, 1, 2, ..., M-1, M_a, M_b)$ then the transition probability matrix $\hat{H} \in [0,1]^{(M+2)\times(M+2)}$ is written as

$$\hat{H} = \begin{bmatrix} 0 & 0 & A_4 & \dots & \dots & 0 & 0 & A_2 \\ A_2 & A_3 & A_1 & 0 & \dots & 0 & 0 & A_4 \\ 0 & A_2 & A_3 & A_1 & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & A_2 & A_3 & A_1 & 0 & 0 \\ 0 & \dots & \dots & 0 & A_2 & A_3 & A_1 & 0 \\ 0 & \dots & \dots & 0 & 0 & 0 & 1 & 0 \\ 0 & \dots & \dots & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (24)$$

The MC's current state $\hat{S}_n$ probability distribution vector is

$$\hat{p}_n^T \triangleq \begin{bmatrix} P_r(\hat{S}_n = 0) \\ P_r(\hat{S}_n = 1) \\ \vdots \\ P_r(\hat{S}_n = M-1) \\ P_r(\hat{S}_n = M_a) \\ P_r(\hat{S}_n = M_b) \end{bmatrix} \in [0,1]^{M+2} \quad (25)$$

and is calculated as

$$\hat{p}_n = \hat{p}_0 \hat{H}^n, \quad (26)$$

with initial distribution vector

$$\hat{p}_0 = e_i \in [0,1]^{M+2}. \quad (27)$$

Considering the above, the probability that the MC has overflowed/underflowed by clock cycle n in states M_a and M_b is $P_r(\hat{S}_n = M_a)$ and $P_r(\hat{S}_n = M_b)$ respectively and is calculated as

$$\Big[P_r(\hat{S}_n = M_a), P_r(\hat{S}_n = M_b)\Big] = \hat{p}_0 \hat{H}^n [e_{M+1}^T, e_{M+2}^T]. \quad (28)$$

B. Expected number of Steps before Overflows/Underflows

It is reasonable to further investigate the overflow/underflow process, especially when the operation of the SFSM and consequently that of the SCPB restrains their occurrence. For this reason, we calculate the expected number of transitions *before the first overflow/underflow*, i.e. before states M_a or M_b are reached. We write matrix $\hat{H}$ in its canonical form [29], [30] as

$$\hat{H} = \left[\begin{array}{c|c} \tilde{H} & R \\ \hline 0_{2,M} & I_2 \end{array}\right], \quad (29)$$

where $\tilde{H} \in [0,1]^{M\times M}$, $R \in [0,1]^{M\times 2}$, $I_2 \in [0,1]^{2\times 2}$ and $0_{2,M} \in [0,1]^{2\times M}$. Using $\tilde{H}$, the fundamental matrix of the absorbing MC [29], [30] is calculated as

$$F = (I_M - \tilde{H})^{-1} \in \mathbb{R}^{M\times M}. \quad (30)$$

Considering that the initial state, S_0, can be any within the states $\mathcal{S}$ of the MC of Fig. 3, then the expected number of transitions *before* the MC is absorbed is

$$N^* = p_0 F \underline{1}, \quad (31)$$

where $\underline{1} \in \mathbb{R}^M$ is the column vector of all ones and p_0 is given by (8). The potentially negative impact of overflows/underflows and the importance of N^* in the register's state selection, is discussed in the following subsection.

C. Guidelines to select the number of states

According to the SCPB's operation and the counting process itself, an overflow/underflow does not always result in an erroneous bit at the output. To give a better insight on this, we consider two cases for a MC with a finite number of states M:

- The MC's current state S_n starts from the initial state $S_0 = 0$, transitions within $\mathcal{S}$, and *is allowed* to transition to its saturating states, visiting them repeatedly as well. Typically in SC, such MC describes the operation of a SFSM that approximates an asymptotically bounded function and actually benefits from the overflows/underflows, for instance the stochastic *tanh* [10].
- The MC's current state S_n starts from the initial state $S_0 = 0$, transitions within $\mathcal{S}$, but, *is not allowed* to repeatedly visit the last state, $M-1$ which is a saturating one. Such MC describes the operation of a SFSM that captures the bit-differences from the input sequences and stores them cumulatively in a register, but, does not benefit from overflows as they may result in erroneous bits at the output sequence [20], [31].

From the above, it is reasonable for a SFSM to have the number of its states carefully selected so as to limit the use of registers taxing on the hardware resources. In this direction, one can use the expression of N^* in (31) as a guideline to select M and hence the register's size. First, one has to select the stochastic sequence length N, the number of states M and the input probabilities $X^j = P_r(X_n^j = 1), j = 1, \dots, k$. Since N^* is a function of the inputs and the number of states $M = 2^w$, a reasonable register's size $\hat{w}$ can be selected

$$\hat{w} = \min\Big\{w \in \mathbb{N} \mid \min_{(X^1,\dots,X^k)} N^*(X^1, \dots, X^k, 2^w) \geq N\Big\}. \quad (32)$$

V. Modeling Examples

In this section we show how the the proposed MC framework can be applied to model in detail the statistical properties of two SFSMs, selected from the SC literature. To demonstrate our framework's accurate modeling, we compare its results with those obtained from the numerical calculations for 10^4 runs with i.i.d. inputs, all conducted using Matlab.

A. Modeling Example 1: Stochastic Tanh

Architecture: The first modeling example we consider is the stochastic tanh function (STanh) introduced in [10]. Its architecture is shown in Fig. 5, where $\{X_n\}$ is the i.i.d. input sequence and $\{Z_n\}$ is the output. If $X_n = 1$, then the w-bit register's current value T_n is increased by 1-bit, whereas in the opposite case, i.e. $X_n = 0$, it is decreased by 1-bit.

The up & down counting of T_n occurs within $\mathcal{T}_R \triangleq \{0, 1, \dots, W-1\}$, where W is the total number of states. Here, the up & down counting is realized using a ripple binary counter, able to count up to $W = 2^w$ states, where w is the register's size, but, note that it can also be realized by a shift-register. The first and last states, 0 and $W-1$, are saturating, which means that they cannot be exceeded.

Therefore, considering (2), with initial state $T_0 = W/2$, T_n is updated as

$$T_n = \max\left\{\min\left\{T_{n-1} + X_n - \overline{X}_n, W-1\right\}, 0\right\}.$$

The instantaneous value of the output Z_n, is determined by the state's current value as $Z_n = T_n \geq W/2$. According to the analysis in [10] and considering the above, for an input X representing a stochastic number in bipolar format, the configuration shown in Fig. 5 approximates the Tanh$(\cdot)$ function as STanh$(W, X) \approx$ Tanh$(XW/2)$.

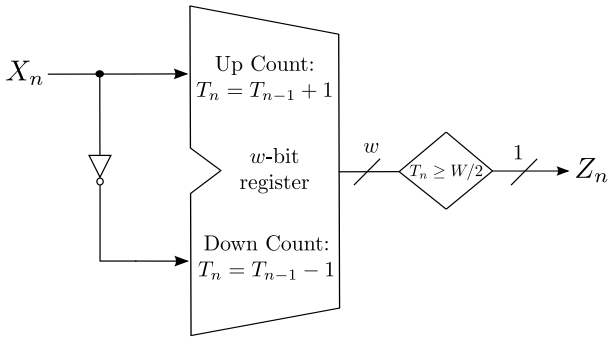

Fig. 5. Architecture of the stochastic tanh function.

Markov Chain Modeling: The operation of the STanh architecture shown in Fig. 5 can be described by the MC model of Fig. 6. Its states have an one-to-one correspondence with the register's ones and therefore the MC's current value S_n transitions within $\mathcal{S} = \{0, 1, \ldots, M-1\}$. The transition probabilities are

$$\begin{aligned} A_1 &= P_r(X_n = 1) \\ A_2 &= P_r(X_n = 0) = 1 - P_r(X_n = 1) \end{aligned} \tag{33}$$

and can be used to describe the transition probability matrix $H \in \mathbb{R}^{M\times M}$ as

$$H = \begin{bmatrix} A_2 & A_1 & 0 & \ldots & \ldots & 0 \\ A_2 & 0 & A_1 & 0 & \ldots & 0 \\ 0 & A_2 & 0 & A_1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & A_2 & 0 & A_1 \\ 0 & \ldots & \ldots & 0 & A_2 & A_1 \end{bmatrix}. \tag{34}$$

Assuming an initial distribution vector $p_0 = e_{M/2} \in \mathbb{R}^M$, the MC's probability distribution vector p_n is calculated using (7).

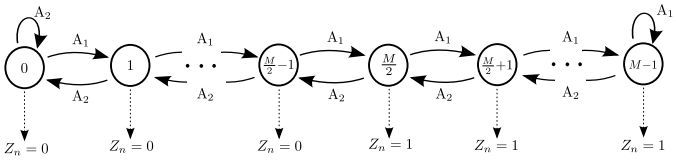

Fig. 6. Markov Chain model describing the operation of the stochastic tanh function. Transition probabilities are given by (33).

First-Moment Statistics: Considering the MC model of Fig. 6, $\mathcal{S}$ can be separated into $\mathcal{S}_0 = \{0, \ldots, M/2-1\}$ and $\mathcal{S}_1 = \{M/2, \ldots, M-1\}$. Therefore, using (10), q is expressed as

$$q = \sum_{i=M/2+1}^{M} e_i, \tag{35}$$

allowing for $\mathbb{E}[Z_n]$ and $\mathbb{E}[\tilde{Z}_N]$ to be calculated using (9) and (12) respectively. A graphical representation of $\mathbb{E}[\tilde{Z}_N]$ approximating the STanh function, is shown in Fig. 7 parameterized with $M = 4$ states and $N = 64$-bit sequence length.

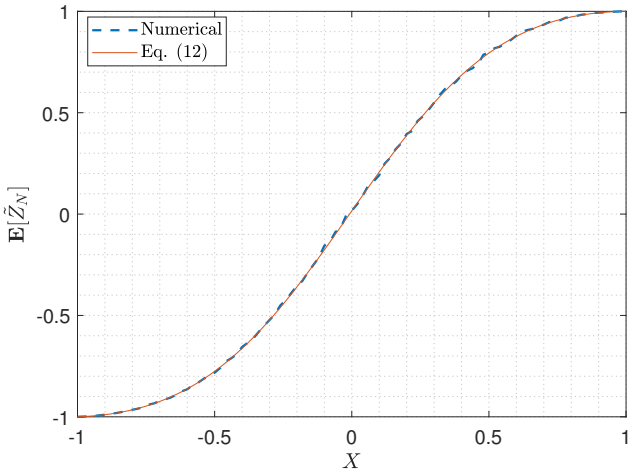

Fig. 7. Expected value of the stochastic tanh's output mean $\mathbb{E}[\tilde{Z}_N]$ calculated using (12), parameterized with $M = 4$ states and sequence length $N = 64$. For the numerical calculations, 10^4 i.i.d. runs for each point are considered.

Second-Moment Statistics: To derive the second-moment statistics, one can start from the calculation of the autocorrelation $R_Z(n+r, n)$ using (13). Note that $Q \in \mathbb{R}^{M\times M}$ is obtained from (14), where the vector q is used from (35). Once $R_Z(n+r, n)$ is calculated, it can be used to derive the autocovarinace $C_Z(n+r, n)$ using (15). In Fig. 8, $C_Z(n+r, n)$ is plotted, for $M = 4$ states, input sequence length $N = 256$ and two time lags $r = 0, 1$. As one can observe, $C_Z(n+r, n)$ peaks when $X = 0$ (bipolar format) and is reduced when the delay is increased from 0 to 1 samples. The variance of the output's mean Var$(\tilde{Z}_N)$ is calculated using (22). In Fig. 9 we demonstrate this calculation using $M = 4$ states and input sequence length $N = 64$.

Mean Squared Error: The MSE is calculated using (23), in which $\bar{Z} =$ Tanh$(XM/2)$. The results are shown in Fig. 10 for $M = 4$ states and input sequence length $N = 64$.

Overflow/Underflow Modeling: The modeling of overflows/underflows is achieved using the MC model of Fig. 11, which contains the two absorbing states M_a, M_b. With state ordering $(0, 1, 2, ..., M-1, M_a, M_b)$ and the transition probabilities from (33), the transition probability matrix $\hat{H} \in$

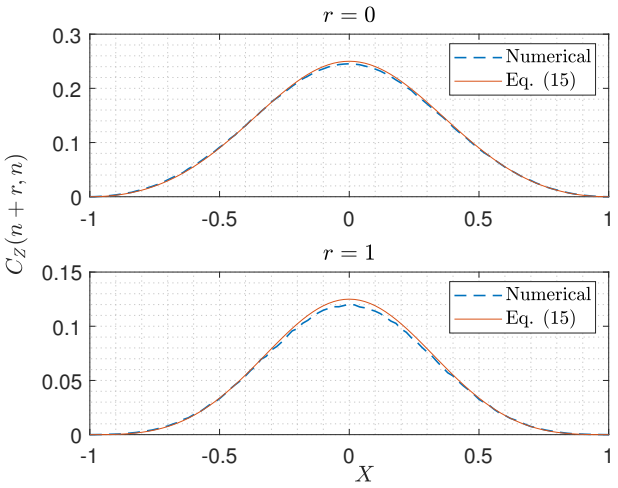

Fig. 8. Auto-Covariance $C_Z(n+r,n)$ of the stochastic tanh's output calculated using (15), parameterized with $M = 4$ states, sequence length $N = 256$ and time lags $r = 0, 1$. For the numerical calculations, 10^4 i.i.d. runs for each point are considered.

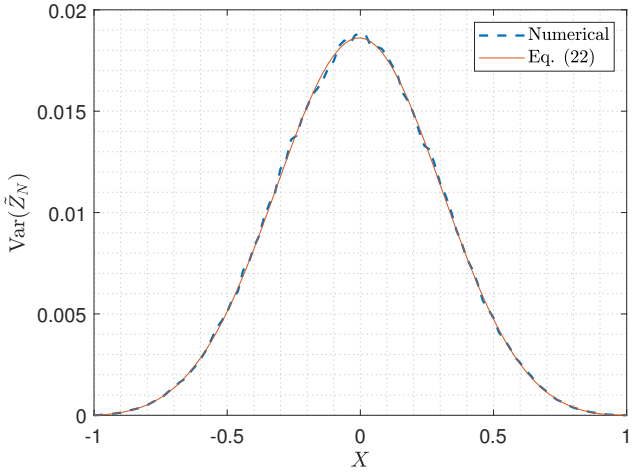

Fig. 9. Variance $\mathrm{Var}(\tilde{Z}_N)$ of the stochastic tanh's output mean calculated using (22), parameterized with $M = 4$ states and sequence length $N = 64$. For the numerical calculations, 10^4 i.i.d. runs for each point are considered.

$\mathbb{R}^{(M+2)\times(M+2)}$ becomes

$$\hat{H} = \begin{bmatrix} 0 & A_1 & 0 & \dots & \dots & 0 & 0 & A_2 \\ A_2 & 0 & A_1 & 0 & \dots & 0 & 0 & 0 \\ 0 & A_2 & 0 & A_1 & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & A_2 & 0 & A_1 & 0 & 0 \\ 0 & \dots & \dots & 0 & A_2 & 0 & A_1 & 0 \\ 0 & \dots & \dots & 0 & 0 & 0 & 1 & 0 \\ 0 & \dots & \dots & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (36)$$

Using (36), the probability distribution vector $\hat{p}_n$ is calculated from the expression given in (26), where the initial distribution vector is $\hat{p}_0 = e_{M/2}$. In addition, Fig. 12 shows the probability of overflow/underflow calculated using (28) for $X = 0.5$ (unipolar format), sequence length $N = 64$ and increasing number of states $M = 4, \dots, 32$.

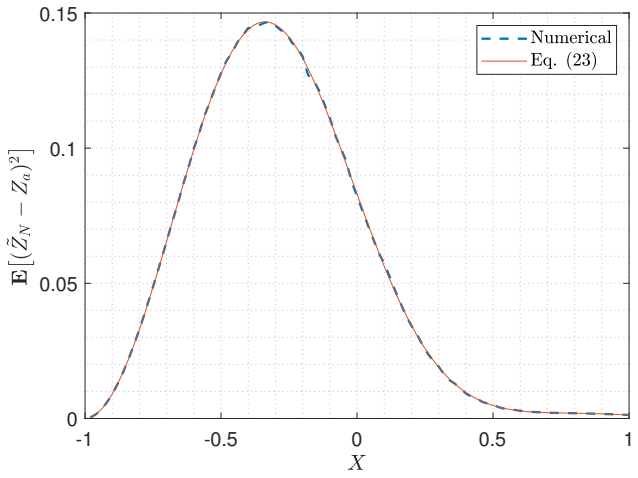

Fig. 10. Mean Squared Error of the stochastic tanh's output mean calculated using (23) for $M = 4$ states and input sequence length $N = 64$. For the numerical calculations, 10^4 i.i.d. runs for each point are considered.

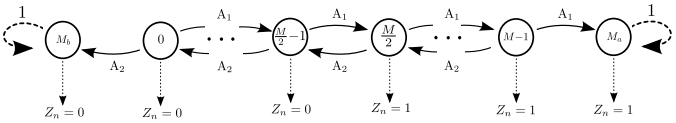

Fig. 11. Markov Chain overflow/underflow model of the stochastic tanh function. Transition probabilities are given by (33).

Register's size selection: The matrix $\hat{H}$ from (36) can be used to derive the fundamental matrix F according to (30). Then, the expected number of steps before overflows N^* can be calculated using (31), considering that $p_0 = e_{M/2} \in \mathbb{R}^M$. In Fig. 13, N^* is plotted, parametrized with sequence length $N = 32$ and state sizes $M = 8, 16, 32$. It can be observed that the condition $N^* \geq N$ from (32), is satisfied only when $M = 16, 32$ states are used.

One can conclude that the advantage of modeling the expected number of steps before overflows N^* is twofold; on the one hand, it allows to accurately select the number of states that reduce the overflow/underflow occurrence, while on the other it prevents from selecting an unnecessarily large number of states, taxing on the hardware resources.

B. Modeling Example 2: Stochastic Adder

Architecture: The second modeling example we consider, is the non-scaling adder introduced in [31]. Its architecture is shown in Fig. 14, where $\{X_n^1\}, \{X_n^2\}$ are i.i.d. input sequences and $\{Z_n\}$ is the output. Its principle operation is based upon the storing of logic ones in a w-bit register when $X_n^1 = X_n^2 = 1$ so as to output them in a future clock cycle n' for which $X_{n'}^1 = X_{n'}^2 = 0$. The register's current value T_n, up and down counts within $\mathcal{T}_R = \{0, 1, \dots, W-1\}$, where $W = 2^w$ is the total number of states. Therefore, T_n's accumulating behavior is expressed as [31]

$$T_n = \min\left\{T_{n-1} + X_n^1 X_n^2 - \left(T_{n-1} > 0\right)\overline{X}_n^1 \overline{X}_n^2, W-1\right\}.$$

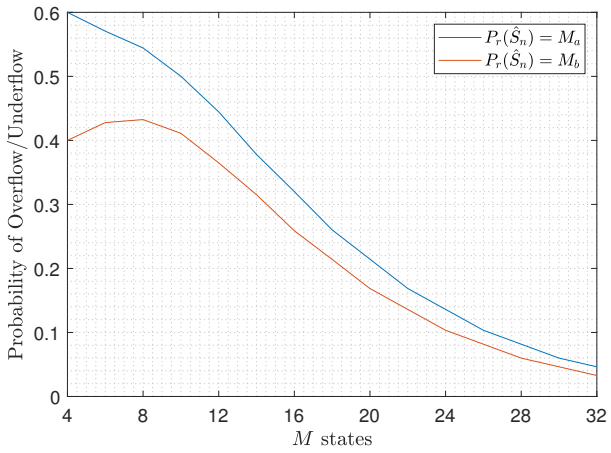

Fig. 12. Probability of overflow/underflow of the stochastic tanh calculated using (28) for increasing number of states $M = 4, \ldots, 32$, input $X = 0.5$ and sequence length $N = 64$.

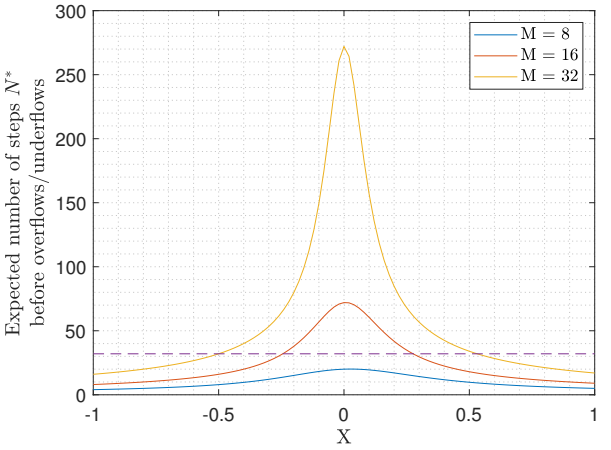

Fig. 13. Expected number of steps before overflows/underflows N^* of the stochastic tanh calculated using (32), for $M = 8, 16, 32$ states and sequence length $N = 32$ (dashed line). The guideline $N^* \geq N$ allows for reduced overflow/underflow occurrence.

From the architecture of Fig. 14, the instantaneous output can be described as $Z_n = X_n^1 + X_n^2 + T_{n-1} > 0$. Note that for the architecture's proper operation, it holds $0 \leq X^1 + X^2 \leq 1$.

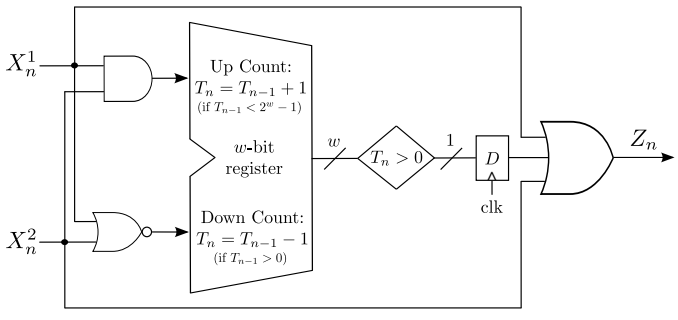

Fig. 14. Architecture of the stochastic adder [31].

Markov Chain Modeling: The MC model of Fig. 15 describes the operation of the adder's architecture. Here, the register's initial value 0 is represented by two states in the model, 0_A and 0_B, so as for its SFSM to be expressed as a Moore one. Hence, its current state S_n transitions within $M+1$ values within $\mathcal{S} = \{0_A, 0_B, 1, 2, ..., M-1\}$, while its transition probabilities are

$$\begin{aligned} A_1 &= P_r(X_n^1 = 0)P_r(X_n^2 = 0) \\ A_2 &= P_r(X_n^1 = 1) + P_r(X_n^2 = 1) - 2P_r(X_n^1 = 1)P_r(X_n^2 = 1) \\ A_3 &= P_r(X_n^1 = 1)P_r(X_n^2 = 1). \end{aligned} \tag{37}$$

They can be used to write the transition matrix $H \in \mathbb{R}^{(M+1)\times(M+1)}$ as

$$H = \begin{bmatrix} A_1 & A_2 & A_3 & \ldots & \ldots & 0 \\ A_1 & A_2 & A_3 & \ldots & \ldots & 0 \\ 0 & A_1 & A_2 & A_3 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & A_1 & A_2 & A_3 \\ 0 & \ldots & \ldots & 0 & A_1 & A_2 + A_3 \end{bmatrix}. \tag{38}$$

Since the MC's initial state is $S_0 = 0_A$, here the initial distribution vector is $p_0 = e_1 \in \mathbb{R}^{M+1}$ and thus the states' probability distribution vector p_n is calculated using (7) [31].

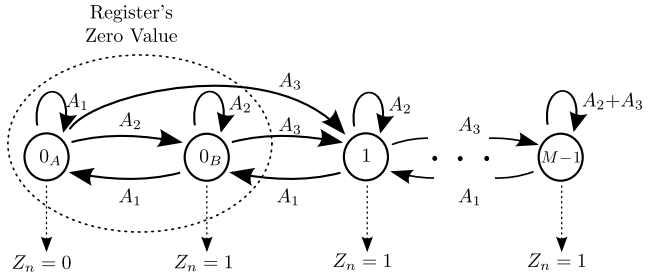

Fig. 15. Markov Chain model describing the operation of the stochastic adder. Transition probabilities are given by (37).

First-Moment Statistics: Observing the MC model of Fig. 15, one can see that $S_n = 0_A \Rightarrow Z_n = 0$, separating $\mathcal{S}$ into $\mathcal{S}_0 = \{0_A\}$ and $\mathcal{S}_1 = \{1, 2, \ldots, M-1\}$. Therefore, vector $q = [0, 1, \ldots, 1] \in \mathbb{R}^{M+1}$ is expressed as

$$q = \sum_{i=2}^{M+1} e_i, \tag{39}$$

and can be used to calculate $\mathbb{E}[Z_n]$ and $\mathbb{E}[\tilde{Z}_N]$ using (9) and (12) respectively [31]. For two inputs $X^1, X^2 \in [0, 1]$, the expected value of the output's mean is shown in Fig. 16, parametrized with $M = 8$ states and $N = 64$-bit sequence length. From Fig. 16, it can be seen that the distribution of the output's mean, $\mathbb{E}[\tilde{Z}_N]$, calculated using (12), matches the one obtained from the numerical experiments, verifying also the correctness of the additions for two inputs $X^1, X^2 \in [0, 1]$, such that $0 \leq X^1 + X^2 \leq 1$.

Second-Moment Statistics: The auto-correlation $R_Z(n+r, n)$ is calculated using (13), considering the vector q from (39) and can be used to calculate $C_Z(n+r, n)$ from the expression (15). The auto-covariance $C_Z(n+r, n)$ is illustrated in Fig. 17, for two inputs $X^1, X^2 \in [0, 1]$, parametrized with $M = 8$ states, input sequence length $N = 64$ and a delay $r = 1$. One can observe that the auto-covariance peaks when $X^1 = X^2 = 0.5$ with a negligible value of approximately 0.03 and gradually

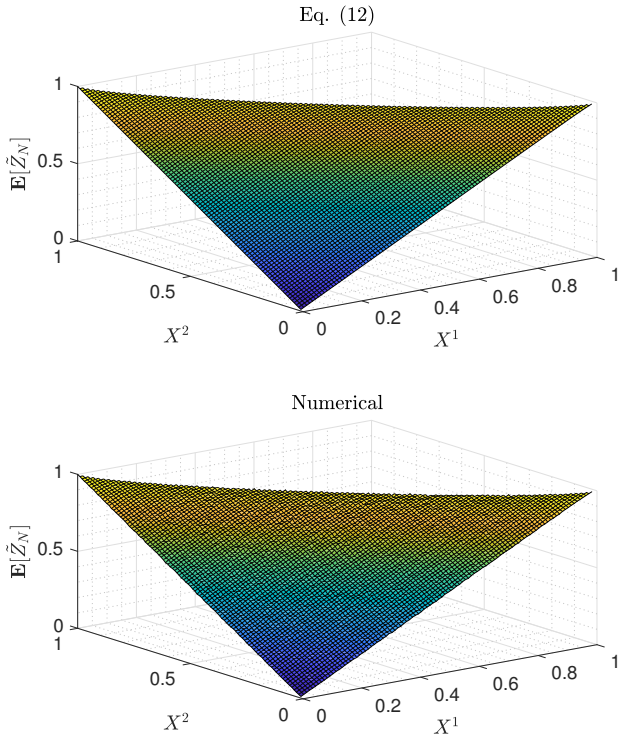

Fig. 16. Expected value of the stochastic adder's output mean $\mathbb{E}[\tilde{Z}_N]$. Top: calculated using (12), parametrized with $M = 8$ states and sequence length $N = 64$. Bottom: Numerical calculations for 10^4 i.i.d. runs for each point.

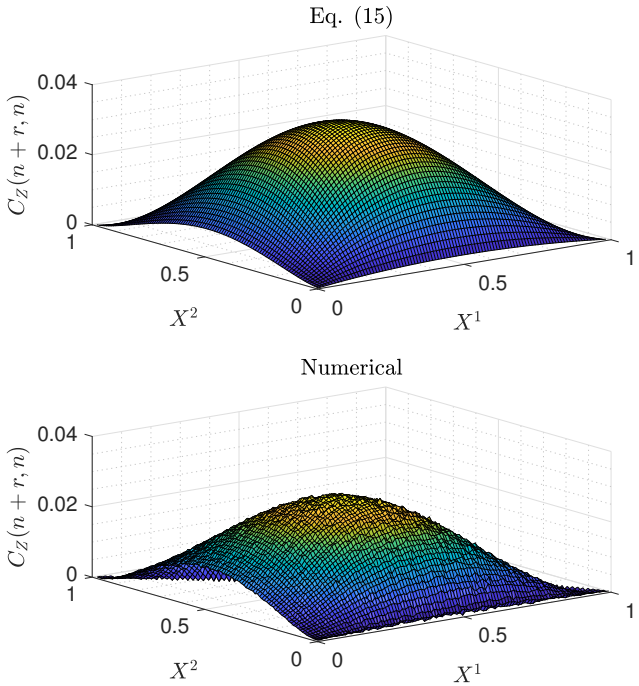

Fig. 17. Auto-Covariance $C_Z(n + r, n)$ of the stochastic adder's output. Top: Calculated using (15), parametrized with $M = 8$ states, sequence length $N = 64$ and delay $r = 1$. Bottom: Numerical calculations for 10^4 i.i.d. runs for each point.

decreases when moving away from these values. Notice that the results obtained from the analytic calculation, follow the ones from the numerical experiments.

Considering $C_Z(n + r, n)$, the variance of $\tilde{Z}_N$, Var$(\tilde{Z}_N)$, can be calculated using the expression (22). In Fig. 18 it is demonstrated for $M = 8$ states and input sequence length $N = 64$. From Fig. 18, it is observed that the analytic calculation of Var$(\tilde{Z}_N)$ follows closely the one obtained from the numerical experiments, where the results have values with order of magnitude up to 10^{-3}.

Mean Squared Error: The MSE of the adder's output mean can be calculated using (23), where $\bar{Z} = X^1 + X^2$. In Fig. 19 the MSE is shown for $M = 8$ states, sequence length $N = 64$ and inputs $X^1, X^2 \in [0, 1]$. From Fig. 19, it is noticeable that the analytic calculation of the MSE$(\tilde{Z}_N)$ using (23) matches the one obtained from the numerical experiments.

Overflow Modeling: The procedure to model overflows deviates from the previous SFSM example. Here, we are interested in "how far" the MC's current value S_n can transition, corresponding to "how many" additional logic 1s are stored from the counting process. Therefore, the modeling of overflows becomes one-sided, in the sense that only one absorbing state is used, M_a. In Fig. 20, the adder's MC overflow model is shown.

With state ordering $(0_A, 0_B, 1, \ldots, M-1, M_a)$, the transition probability matrix $\hat{H} \in \mathbb{R}^{(M+2)\times(M+2)}$ is written using the transition probabilities given in (37) as

$$\hat{H} = \begin{bmatrix} A_1 & A_2 & A_3 & \ldots & \ldots & \ldots & 0 \\ A_1 & A_2 & A_3 & \ldots & \ldots & \ldots & 0 \\ 0 & A_1 & A_2 & A_3 & \ldots & \ldots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ldots & 0 \\ \vdots & \ldots & 0 & A_1 & A_2 & A_3 & 0 \\ \vdots & \ldots & \ldots & 0 & A_1 & A_2 & A_3 \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 1 \end{bmatrix}, \quad (40)$$

and using $\hat{p}_0 = e_1 \in \mathbb{R}^{M+2}$, the probability distribution vector after N steps is calculated with the expression (26). Considering that only one absorbing state is used, then the probability of overflow is [31]

$$P_r(S_n = M_a) = \hat{p}_0 \hat{H}^n e_{M+2}^T \quad (41)$$

In Fig. (21), the adder's probability of overflow is graphically illustrated, for inputs with values $X^1 = X^2 = 0.5$, increasing number of states $M = 4, \ldots, 32$ and sequence lengths $N = 16, 32, 64, 256$. As expected, an increase on the number of states, reduces the probability of overflow.

Register's size selection: For the calculation of the expected number of steps before overflows N^*, the matrix $\hat{H}$ from (40) is used along with the initial distribution vector $p_0 = e_1 \in \mathbb{R}^{M+1}$. In Fig. 22, N^* is plotted for inputs $X^1 = X^2 = 0.5$, increasing number of states $M = 4, \ldots, 32$ and stochastic sequence lengths $N = 16, 32, 64, 128, 256$.

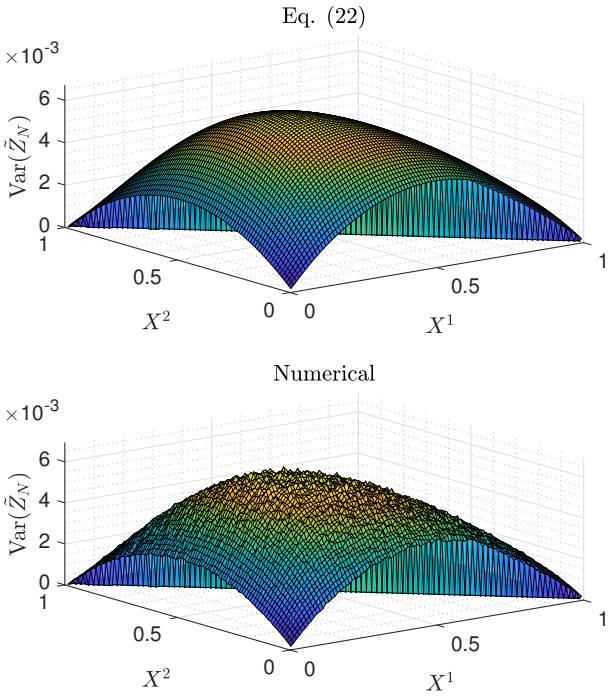

Fig. 18. Variance of the stochastic adder's output mean $\text{Var}(\tilde{Z}_N)$. Top: calculated using (22), parametrized with $M = 8$ states and sequence length $N = 64$. Bottom: Numerical calculations for 10^4 i.i.d. runs for each point.

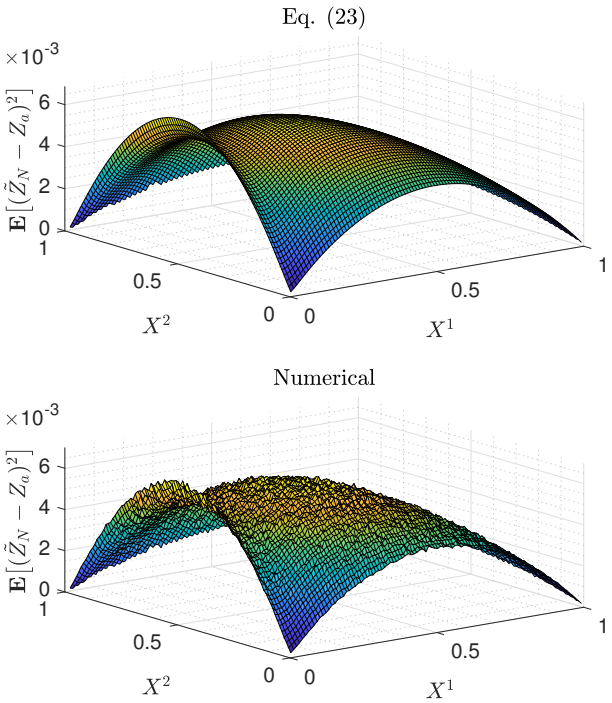

Fig. 19. Mean Squared Error of the stochastic adder's output mean $\text{MSE}(\tilde{Z}_N)$. Top: calculated using (23), parametrized with $M = 8$ states and input sequence length $N = 64$. Bottom: Numerical calculations for 10^4 i.i.d. runs for each point.

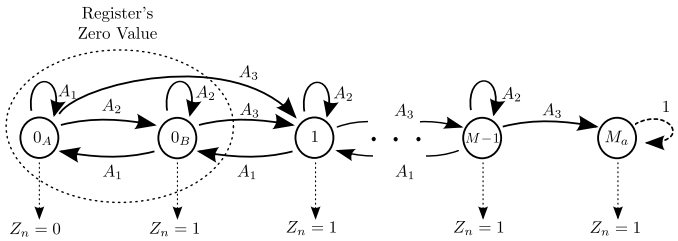

Fig. 20. Markov Chain overflow model of the stochastic adder. Transition probabilities are given by (37).

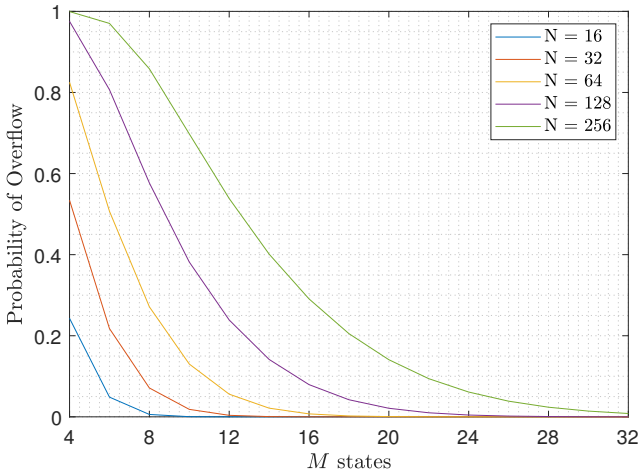

Fig. 21. Probability of overflow of the stochastic adder calculated using (41), for inputs $X^1 = X^2 = 0.5$, increasing number of states $M = 4, \ldots, 32$ and increasing sequence lengths N.

It can be seen from Fig. 22 that for small values of N, namely $N = 16, 32$, a slight increase on the number of states, for instance from $M = 4$ to $M = 8$, has negligible difference on the condition to be satisfied, $N^* \geq N$. However, this is not the case for large values of N, for instance $N = 128$ and more, in which an increase of the number of states (from $M = 4$ to $M = 8$) and hence the register's size (from 2-bit to 3-bit), is necessary to satisfy $N^* \geq N$.

C. Execution Time Performance

To highlight the time efficiency of our proposed framework in the modeling of the SFSMs' statistical properties, we compare its execution times with those obtained from the numerical experiments. For the STanh, we use 10^2 input values uniformly distributed in $[0, 1]$ and for the Stochastic Adder we use 10^4 input values uniformly distributed in $[0, 1] \times [0, 1]$. Regarding the numerical experiments, we conduct 10^4 and 10^5 runs with i.i.d input sequences of length $N = 64$-bits for each input value we consider. To measure the relative performance we use the speedup metric, which is the ratio of the execution time of the numerical experiments, L_N, over that of the analytical modeling one, L_M, i.e. Speedup $= L_\text{N}/L_\text{M}$. The execution times are used to calculate the time saving metric as $(L_N - L_M)/L_N * 100\%$. We also cite the Mean Absolute Error (MAE), which is the absolute difference between the averaged output of the numerical experiments for 10^4 and 10^5

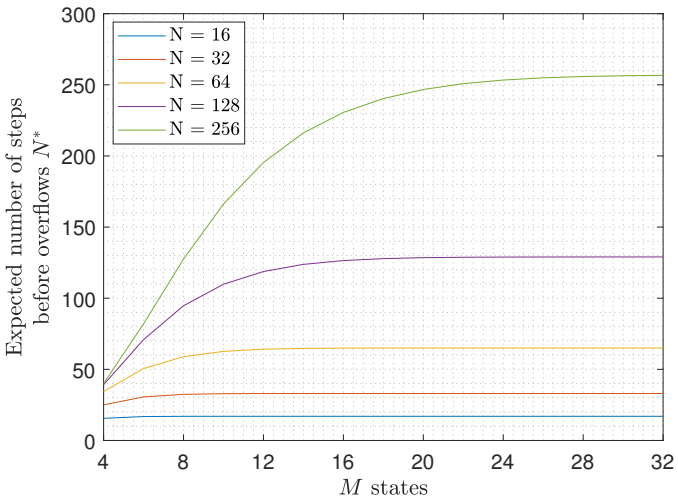

Fig. 22. Expected number of steps before overflows N^* of the stochastic adder calculated using (32), for $M = 4, \ldots, 32$ states, inputs $X^1 = X^2 = 0.5$ and increasing sequences lengths N. The guideline $N^* \geq N$ allows for reduced overflow occurrence.

runs and the analytical modeling output, summed over all the uniformly distributed input values.

Table I, presents the numerical simulation and analytical calculation execution times of the two SFSMs. When 10^5 runs are used, it is observed that the calculation of the expected value and the auto-correlation using the proposed framework, result in substantial time savings for both the STanh, 99.47% and 99.90% respectively, and the Stochastic Adder, 99.94% and 99.96% respectively. With respect to the calculation of the variance and the MSE, the analytical modeling of our proposed method yields significant time savings, corresponding to 95.07% and 94.93% respectively for the STanh and 92.21% and 91.91% respectively for the Stochastic Adder. Decreasing the number of runs to 10^4, is expected to increase the MAE and the execution time, at the cost however of reducing the numerical experiments' approximations.

VI. Conclusion

In this work a general methodology for the analytic calculation of the statistical properties of Stochastic Computing Finite-State Machines was presented. It was shown that expressing the behavior of any SFSM as a Moore one and model it as a MC, makes the calculation of its first and second moment statistics feasible using closed-form expressions, all parametrized on the input sequence length and the number of states. The proposed methodology was used in the detailed modeling of two applicable SFSMs selected from the SC literature. Comparisons of the proposed framework with the numerical experiments, demonstrated its effectiveness in the accurate modeling of the SFSMs' statistical properties.

Acknowledgment

The research work was supported by the Hellenic Foundation for Research and Innovation (HFRI) under the HFRI PhD Fellowship grant (Fellowship Number:1216).

TABLE I
Execution Times (s) for the Modeling of two SFSMs: the STanh and the Stochastic Adder

	Numerical Experiments(s)	Analytical Modeling(s)	Speedup	MAE	Time Savings (%)
STanh 10^4 runs					
Exp. Value	4.41	0.24	18.37	1.1×10^{-3}	94.55
Auto-Correlation	19.84	0.26	76.30	6.6×10^{-3}	98.58
Variance	6.76	3.35	2.01	1.7×10^{-4}	50.44
MSE	6.87	3.42	2.00	2.6×10^{-4}	50.12
Stochastic Adder 10^4 runs					
Exp. Value	303.80	1.62	187.5	5.1×10^{-4}	99.46
Auto-Correlation	302.89	2.28	132.49	3.8×10^{-3}	99.24
Variance	318.07	249.08	1.26	2.2×10^{-4}	21.69
MSE	320.19	251.15	1.27	1.1×10^{-4}	21.56
STanh 10^5 runs					
	Numerical Experiments(s)	Analytical Modeling(s)	Speedup	MAE	Time Savings (%)
Exp. Value	45.37	0.24	189.04	3.5×10^{-4}	99.47
Auto-Correlation	270.85	0.26	1041.73	9.8×10^{-4}	99.90
Variance	68.06	3.35	20.31	3.1×10^{-5}	95.07
MSE	67.56	3.42	19.75	8.2×10^{-5}	94.93
Stochastic Adder 10^5 runs					
Exp. Value	3.09×10^3	1.62	1.91×10^3	1.6×10^{-4}	99.94
Auto-Correlation	6.54×10^3	2.28	2.87×10^3	8.3×10^{-4}	99.96
Variance	3.20×10^3	249.08	1.28×10^3	4.9×10^{-5}	92.21
MSE	3.11×10^3	320.19	1.24×10^3	1.7×10^{-5}	91.91

References

[1] B. R. Gaines, *Stochastic Computing Systems*. Springer, Boston, MA, 1967.

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

[3] W. Qian, M. D. Riedel, H. Zhou, and J. Bruck, "Transforming probabilities with combinational logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 9, pp. 1279 – 1292, Sep. 2011.

[4] W. J. Gross and V. C. Gaudet, *Stochastic Computing: Techniques and Applications*. Springer, International Publishing, 2019.

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

[6] A. Alaghi and J. P. Hayes, "Strauss: Spectral transform use in stochastic circuit synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 11, Nov. 2015.

[7] N. Temenos and P. P. Sotiriadis, "Deterministic finite state machines for stochastic division in unipolar format," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Seville, Spain, Oct. 2020.

[8] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 2809 – 2824, Jul. 2021.

[9] A. Morro, V. Canals, A. Oliver, M. L. Alomar, F. Galán-Prado, P. J. Ballester, and J. L. Rosselló, "A stochastic spiking neural network for virtual screening," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 1371 – 1375, Apr. 2018.

[10] B. D. Brown and H. C. Card, "Stochastic neural computation i: Computational elements," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 891–905, Sep. 2002.

[11] ——, "Stochastic neural computation ii: Soft competitive learning," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 906–920, Sep. 2002.

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

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

[14] Y. Liu, L. Liu, F. Lombardi, and J. Han, "An energy-efficient and noise-tolerant recurrent neural network using stochastic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 9, pp. 2213 – 2221, Jun. 2019.
[15] 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 *ACM Proceedings of the Conference on Design, Automation & Test in Europe*, Laussane, Switzerland, Mar. 2017.
[16] P. Li and D. J. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *IEEE 29th International Conference on Computer Design (ICCD)*, Amherst, MA, USA, Oct. 2011.
[17] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 2, no. 3, pp. 449–462, Apr. 2014.
[18] A. Alaghi, C. Li, and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *IEEE 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, USA, May 2013.
[19] M. Lunglmayr, D. Wiesinger, and W. Haselmayr, "Design and analysis of efficient maximum/minimum circuits for stochastic computing," *IEEE Transactions on Computers*, vol. 69, no. 3, Mar. 2020.
[20] N. Temenos and P. P. Sotiriadis, "Stochastic computing max & min architectures using markov chains: Design, analysis, and implementation," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 29, no. 11, pp. 1813 – 1823, 2021.
[21] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *IEEE 31st International Conference on Computer Design (ICCD)*, Asheville, NC, USA, Oct. 2013.
[22] V. T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, Mar. 2018.
[23] Y. Liu, M. Parhi, M. D. Riedel, and K. K. Parhi, "Synthesis of correlated bit streams for stochastic computing," in *IEEE 50th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, Nov. 2016.
[24] T. J. Baker and J. P. Hayes, "Impact of autocorrelation on stochastic circuit accuracy," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Miami, FL, USA, Jul. 2019.
[25] C. Ma, S. Zhong, and H. Dang, "Understanding variance propagation in stochastic computing systems," in *IEEE International Conference on Computer Design (ICCD)*, Montreal, QC, Canada, Oct. 2012.
[26] S. L. J. Han, "Toward energy-efficient stochastic circuits using parallel sobol sequences," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, pp. 1326 – 1339, Jul. 2018.
[27] M. Ciletti and M. M. Mano, *Digital Design Global Edition*, 6th ed. Pearson, 2018.
[28] J. R. Norris, *Markov Chains*, 1st ed. Cambridge University Press, 1997.
[29] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed. American Mathematical Society, 1997.
[30] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, 1st ed. Society for Industrial and Applied Mathematics, 2000.
[31] N. Temenos and P. P. Sotiriadis, "Nonscaling adders and subtracters for stochastic computing using markov chains," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 29, no. 9, pp. 1612 – 1623, 2021.

Nikos Temenos (S'13) received the B.Sc. degree in Computer and Systems Engineering from Piraeus University of Applied Sciences, Greece, in 2015 and the M.Sc. in Microelectronics from the National and Kapodistrian University of Athens, Greece, in 2017. He is currently working towards the Ph.D. degree at National Technical University of Athens.

His main research area includes digital VLSI design, computer arithmetic as well as algorithms and architectures for Stochastic Computing targeting FPGA and ASIC technologies. He has authored and co-authored several IEEE conferences and is a regular reviewer for many IEEE transactions and conferences.

Paul P. Sotiriadis (SM'18) is a faculty member of the Electrical and Computer Engineering Department of the National Technical University of Athens, Greece, the Director of the Electronics Laboratory of the University and a member of the governing board of the Hellenic Space Center, the National space center of Greece. He received the Diploma degree in Electrical and Computer Engineering from same Department, the M.S. degree in Electrical Engineering from Stanford University, California USA and the Ph.D. degree in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge USA, in 2002. In 2002, he joined the Johns Hopkins University as Assistant Professor of Electrical and Computer Engineering.

In 2012, he joined the faculty of the Electrical and Computer Engineering Department of the National Technical University of Athens. He has authored and coauthored more than 180 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 interconnect networks in deep-submicrometer technologies. He has led several projects in these fields funded by U.S. organizations and has collaborations with industry and national labs.

He has received several awards, including the 2012 Guillemin-Cauer Award from the IEEE Circuits and Systems Society, a Best Paper Award in the IEEE International Symposium on Circuits and Systems 2007, a Best Paper Award in the IEEE International Frequency Control Symposium 2012 and a Best Paper Award in the IEEE International Conference on Modern Circuits and Systems Technologies 2019. Dr. Sotiriadis is an Associate Editor of the IEEE Sensors Journal, has served as an Associate Editor of the IEEE Transactions on Circuits and Systems I: Regular Papers (2016-2020) and the IEEE Transactions on Circuits and Systems II: Express Briefs (2005-2010), and 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.