Stochastic Computing Max & Min Architectures Using Markov Chains: Design, Analysis, and Implementation

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

***Abstract*— Max & min architectures for stochastic computing (SC) are introduced. Their key characteristic is the utilization of an accumulator to store the signed difference between the two inputs, without randomizing sources. This property results in fast-converging and highly accurate computations using short sequence lengths, improving on the latency–accuracy tradeoff of existing SC max–min architectures. The operation of the proposed architectures is modeled using Markov Chains, resulting in in-depth analysis, the derivation of their statistical properties, and guidelines for selecting the register's size to achieve overall design optimization. The computational accuracy and the hardware requirements of the proposed architectures are compared to those of existing ones in the SC literature, using MATLAB and Synopsys Tools. The efficacy of the proposed architectures is demonstrated by realizing a 3 × 3 median filter and using it in an image processing application.**

***Index Terms*— Stochastic computing (SC), stochastic max, stochastic median filter, stochastic min, unconventional computing.**

I. Introduction

UNCONVENTIONAL computing paradigms have recently gained attention as computationally efficient alternatives to binary computing in many emerging applications implemented in VLSI hardware [1]–[5]. Among many, Stochastic Computing (SC) is considered an effective approach [1], [2], [6]–[8].

In SC, information is processed in probabilistic terms by encoding real numbers as stochastic sequences [9], [10]. Its probabilistic nature, therefore, makes it robust to soft-errors [2]. In addition, SC realizes fundamental arithmetic operations and highly complex functions using very simple and compact logic cells, thereby reducing dramatically the hardware area requirements compared to the traditional binary arithmetic [2], [8], [11], [12]. Beyond its strong points, SC requires several computational cycles to achieve high computational accuracy, which impacts its power and energy efficiency [6], [12], [13]. Hence, to make the best of it, achieving low latency combined with increased computational accuracy in SC designs is of primary concern.

On application level, SC's advantages are favored in the emerging fields of neural networks (NNs) [4], [6], [14]–[18] and digital image processing [3], [19], [20]. These fields require nonlinear blocks as an essential part of their digital signal processing (DSP) cores (besides the basic arithmetic operations), implemented by finite state machines (FSMs) [3], [7], [15], [19]. Stochastic FSMs can realize effectively nonlinear functions, such as the tanh, the exponential, the linear gain, and the max & min [3], [4], [7], [15], [19].

The max & min are very popular nonlinear functions [1], especially in max-pooling operations, and thus, their efficient implementation is significant within SC's context. Current max & min architectures include the following ones.

The architecture by Lee *et al.* [21] realizes the stochastic max & min by correlating [22] the input sequences using a three-state FSM and then a single gate to produce the output, depending on the desired function (max or min). The FSM's number of states limits the accuracy of the output since it can only store logic ones according to the FSM depth used.

Another architecture, by Li *et al.* [3], uses MUXs and the FSM-based tanh function [15] to realize the max & min. One of the two MUXs, though, uses an additional hardware-demanding binary-to-stochastic converter to generate the MUX's select signal (besides its inputs), thus increasing the hardware requirements [12]. Furthermore, the dependence of the FSM's number of states with the input sequence length requires numerical simulations beforehand to derive the register's size that yields the highest computational accuracy.

Following Li *et al.* [3], the approach by Yu *et al.* [23] replaces the binary-to-stochastic converter with an XOR to reduce the hardware overhead, keeping the rest of the processing structure.

A recent method to realize the max & min is proposed by Lunglmayr *et al.* [7]. Instead of tanh-based FSM as in [23], it uses a shift register to store the ones from one of its inputs, and its least significant bit (LSB) produces a logic 1 if it has saturated up to the LSB. Similar to [23], the size of shift register that yields the highest computational accuracy is derived with numerical simulations according to the stochastic sequence length used. Moreover, if the shift register's size is not selected accurately, the output's accuracy is reduced, as shown in [7].

Manuscript received March 2, 2021; revised May 28, 2021; accepted July 26, 2021. Date of publication October 7, 2021; date of current version November 3, 2021. This work was supported by the Hellenic Foundation for Research and Innovation (HFRI) under the HFRI PhD Fellowship Grant (Fellowship Number: 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.2021.3114424.

Digital Object Identifier 10.1109/TVLSI.2021.3114424

Motivated by the design challenges of the former methods combined with the necessity for fast computations in SC, we propose a different approach for max & min. The proposed architectures utilize an accumulator to capture and store the signed bit differences between their two input sequences, without additional random sources, making their operation deterministic. This results in fast convergence and at the same time highly accurate computations using short input sequence lengths.

The above properties are demonstrated by modeling the architectures using Markov Chains (MCs), allowing us to: 1) explain their operating principles in detail; 2) derive the first-order statistics of their output and prove their proper operation at the limit; 3) calculate analytically the probability of overflow; and 4) provide guidelines to select their register's size based on accuracy requirements.

This article is organized as follows. In Section II, we provide a background on the notation and the statistical properties of the stochastic numbers. In Section III, we introduce the proposed stochastic max architecture and analyze it mathematically using MCs. Based on the proposed max, in Section IV, we present the proposed min along with its analysis. In Section V, we present extensive comparisons between the proposed architectures and existing ones selected from the SC literature in terms of computational accuracy and hardware resources. In Section VI, we demonstrate the architectures' effectiveness by using the proposed max and min to realize a 3×3 median filter, perform a standard image processing task, and compare it with the standard binary approach. Finally, in Section VII, we conclude this work.

II. Stochastic Number Representation

SC requires an interface to encode binary numbers into stochastic ones, a process typically done using stochastic number generators (SNGs) [2], [9], [24]. Their operation is based upon the comparison on each clock cycle of the value of a pseudorandom number generator of k-bits (for instance, a linear feedback shift register (LFSR)) with a fixed value $b \in [0, 1]$ encoded as a k-bit binary word. According to the comparator's result, the SNG outputs a logic 0 or 1, and the bit generation is completed after $N = 2^k$ clock cycles, corresponding to the length of the generated stochastic sequence.

The (random) bits in the N-bit sequence are assumed to be independent identically distributed (i.i.d.) Bernoulli random variables, i.e., $X_n, n = 1, 2, \ldots, N$, where n is the time index. Assuming the unipolar format encoding, the stochastic number's value is

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

within $[0, 1]$, with expected value $X \triangleq P_r(X_n = 1)$. For negative stochastic number representation, known as the bipolar format, the transformation $X \mapsto 2X - 1$ expands the range of the stochastic number to $[-1, 1]$.

III. Stochastic MAX Architecture and Analysis

In this section, we introduce the proposed stochastic max architecture, and we present its mathematical modeling and analysis using MCs.

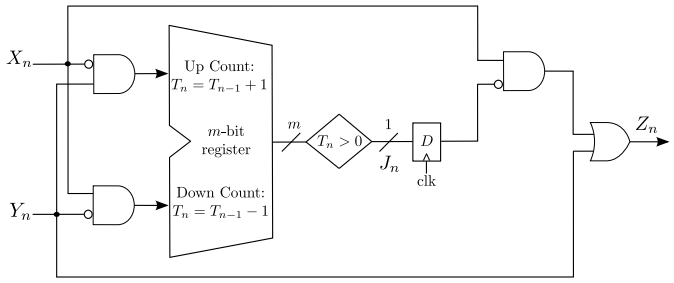

Fig. 1. Proposed stochastic max architecture. T_n is the m-bit register's state, updated according to (1).

A. Architecture

The proposed stochastic max architecture is shown in Fig. 1, where $\{X_n\}$ and $\{Y_n\}$ are the stochastic input sequences and $\{Z_n\}$ is the output. Ideally, if, for some n, it is $Y_n > X_n$, then the m-bit register's value is increased by 1 (up count), whereas, if $Y_n < X_n$, it is decreased by 1 (down count). If $Y_n = X_n$, the register's value remains unchanged. Also, we assume the initial value $T_0 = 0$. One could say that the m-bit register's purpose is to count the *signed* bit-differences between its two inputs.

It is important to note that the up & down counting of the m-bit register is *saturating*, meaning that states 0 and $M - 1$ cannot be exceeded, and it is always $T_n \in \mathcal{T}_R$ where $\mathcal{T}_R \triangleq \{0, 1, 2, \ldots, M - 1\}$, with $M = 2^m$ being the total number of states. Hence, from the architecture of Fig. 1, we conclude that the state T_n evolves according to

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

where $\overline{X}_n = 1 - X_n$ and $\overline{Y}_n = 1 - Y_n$.

The architecture's output Z_n is determined as follows: if X_n and Y_n are both 0 or both 1, then Z_n is 0 or 1, respectively; if $Y_n > X_n$, then $Z_n = 1$; if $Y_n < X_n$, then $Z_n = 1$; if the register was zero in the previous cycle, i.e., if $T_{n-1} = 0$, it is $Z_n = 0$ otherwise.

Defining J_n to be 1 if $T_n > 0$ and zero otherwise, and by inspecting the architecture in Fig. 1, the output Z_n can be expressed as

$$Z_n = Y_n + X_n \overline{J_{n-1}}. \quad (2)$$

The deterministic behavior of the proposed architecture in Fig. 1 is captured by (1) and (2). Specifically, the output Z_n is a function of the inputs and the state T_n without any additional randomization from any source. As such, the resolution of $\{Z_n\}$ is only limited by the length of the N-bit stochastic input sequences and the register's size.

B. Markov Chain Modeling

To model the operation of the proposed architecture with the stochastic inputs, we consider two MC models. The first one is more simple and allows us to easily model the transitions of the state. However, it is not convenient for modeling the output, which is a function of the previous state and the current inputs. To simplify the derivation of output statistics, we extend the first model by doubling the number of states so that the output depends only on the current state. Both models

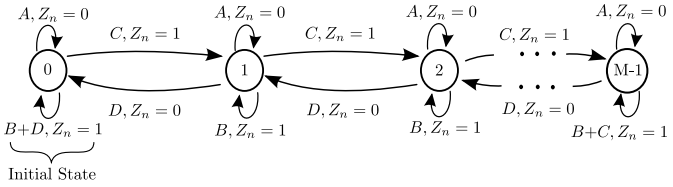

Fig. 2. MC model of the proposed stochastic max architecture. Output Z_n is determined by the state's transition according to transition probabilities A, B, C, and D given by (4).

are helpful in explaining different aspects of the architecture's behavior and are discussed in the following.

1) Markov Chain Model: The first MC model is shown in Fig. 2. It describes the max architecture's operation and corresponds to a Mealy FSM. The model's M states have the obvious one-to-one correspondence with the register's states. The MC model's state S_n at time index n, starting from $S_0 = 0$, transitions within the set

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

If the MC's current state is S_{n-1} at time index $n-1$, then inputs X_n and Y_n along with S_{n-1} determine the output Z_n and the next state S_n. The transition probabilities A, B, C, and D are

$$\begin{aligned} A &= P_r(X_n = 0)P_r(Y_n = 0) \\ B &= P_r(X_n = 1)P_r(Y_n = 1) \\ C &= P_r(X_n = 0)P_r(Y_n = 1) \\ D &= P_r(X_n = 1)P_r(Y_n = 0). \end{aligned} \tag{4}$$

To proceed with the analysis of the MC's behavior, we define the $M \times M$ transition probability matrix

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

where $P_r(S_{n+1} = s | S_n = \sigma)$ is the transition probability from state σ to state s, at time index n, and $\sigma, s = 0, 1, \ldots, M-1$. From (4), it is

$$H = \begin{bmatrix} 1-C & C & 0 & \ldots & \ldots & 0 \\ D & A+B & C & 0 & \ldots & 0 \\ 0 & D & A+B & C & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & D & A+B & C \\ 0 & \ldots & \ldots & 0 & D & 1-D \end{bmatrix}. \tag{5}$$

The probability distribution vector of state S_n is defined as

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

For $n = 1, 2, \ldots, N$ steps, it is expressed as

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

where p_0 is the initial distribution vector representing the starting state of the register, i.e., $S_0 = 0$, i.e.,

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

The analysis of the MC model of Fig. 2 is used to derive guidelines for the register's size, presented in Section III-E.

2) Extended Markov Chain Model: Despite its simplicity, the MC model of Fig. 2 is not convenient for the analysis of the statistics of the output. Instead, we can double the number of its states to get the MC model of Fig. 3. This extended MC model corresponds to a Moore FSM, relating the output value Z_n only to the state.

Each register state is represented by two states in the model of Fig. 3. The states of the model are classified into two subsets; the first one, $\mathcal{S}_a \triangleq \{0_a, 1_a, \ldots, (M-1)_a\}$ containing the states that output $Z_n = 0$, and the second one, $\mathcal{S}_b \triangleq \{0_b, 1_b, \ldots, (M-1)_b\}$ containing the states that output $Z_n = 1$. The MC's state $\tilde{S}_n$ transitions within the $2M$ states in

$$\tilde{\mathcal{S}} \triangleq \mathcal{S}_a \cup \mathcal{S}_b = \{0_a, 0_b, 1_a, 1_b, \ldots, (M-1)_a, (M-1)_b\} \tag{9}$$

according to inputs X_n, Y_n and with initial state $\tilde{S}_0 = 0_a$.

The transition probability matrix $\tilde{H} \in [0, 1]^{2M \times 2M}$ of the model in Fig. 3 is expressed using A, B, C, and D from (4), the definitions $F \triangleq B + D$ and $U \triangleq B + C$, and the state ordering $\big(0_a, 0_b, 1_a, 1_b, \ldots, (M-1)_a, (M-1)_b\big)$ as follows:

$$\tilde{H} = \begin{bmatrix} A & F & 0 & C & 0 & & & \ldots & & & 0 \\ A & F & 0 & C & 0 & & & \ldots & & & 0 \\ D & 0 & A & B & 0 & C & 0 & \ldots & & & 0 \\ D & 0 & A & B & 0 & C & 0 & \ldots & & & 0 \\ 0 & 0 & D & 0 & A & B & 0 & C & 0 & \ldots & 0 \\ 0 & 0 & D & 0 & A & B & 0 & C & 0 & \ldots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & & \ldots & & 0 & D & 0 & A & B & 0 & C \\ 0 & & \ldots & & 0 & D & 0 & A & B & 0 & C \\ 0 & & \ldots & & & & 0 & D & 0 & A & U \\ 0 & & \ldots & & & & 0 & D & 0 & A & U \end{bmatrix}. \tag{10}$$

The probability distribution vector of state $\tilde{S}_n$ is defined as

$$\tilde{p}_n^T \triangleq \begin{bmatrix} P_r(\tilde{S}_n = 0_a) \\ P_r(\tilde{S}_n = 0_b) \\ P_r(\tilde{S}_n = 1_a) \\ P_r(\tilde{S}_n = 1_b) \\ \vdots \\ P_r(\tilde{S}_n = (M-1)_a) \\ P_r(\tilde{S}_n = (M-1)_b) \end{bmatrix} \in [0, 1]^{2M} \tag{11}$$

and it is expressed as

$$\tilde{p}_n = \tilde{p}_0 \tilde{H}^n \in [0, 1]^{2M} \tag{12}$$

where the initial state of the register $\tilde{S}_0 = 0_a$ is given by

$$\tilde{p}_0 = \big[1, 0, 0, \ldots, 0\big] \in [0, 1]^{2M}. \tag{13}$$

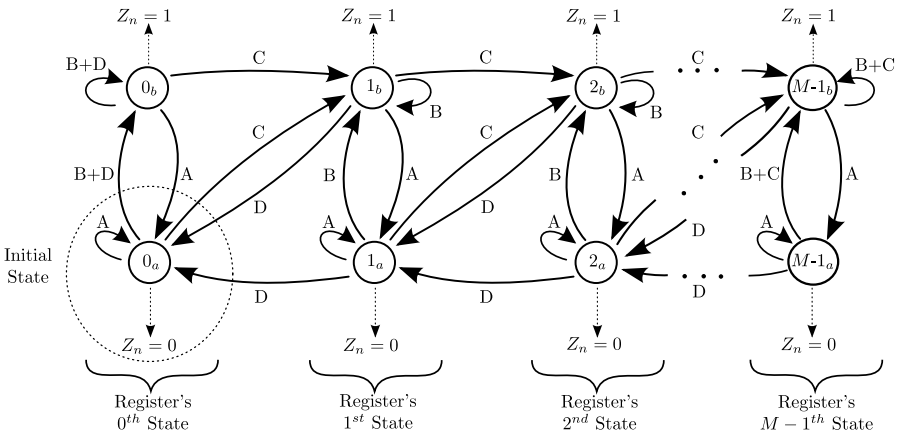

Fig. 3. Extended MC model of the proposed stochastic max architecture with transition probabilities given by (4). Each register state is represented by two states in the model and is classified into two subsets of states: upper ones outputting $Z_n = 1$ and lower ones outputting $Z_n = 0$. Subscripts a and b denote in which subset $\tilde{S}_n$ is currently into. Transition probabilities A, B, C, and D are given by (4).

C. First-Order Statistics and Proof of Operation

To derive the first order statistics of the max architecture, we use the MC model of Fig. 3 along with (10), (12), and (13). Based on the model, we use the fact that $Z_n = 1$ if and only if $\tilde{S}_n \in \mathcal{S}_b$. Therefore, the expected value of the output Z_n is

$$\mathbb{E}[Z_n] = P_r(Z_n = 1) = P_r\left(\tilde{S}_n \in \mathcal{S}_b\right) = \tilde{p}_0 \tilde{H}^n q_e^T \tag{14}$$

with q_e (ones in the even-indexed positions) defined as

$$q_e \triangleq [0, 1, 0, 1, \ldots, 0, 1] \in [0, 1]^{2M}. \tag{15}$$

The average of the N-bit output sequence is

$$\widetilde{Z}_N = \frac{1}{N}\left(Z_1 + Z_2 + \cdots + Z_N\right) \tag{16}$$

and using (14), its expected value is written as

$$\mathbb{E}[\widetilde{Z}_N] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[Z_n] = \frac{1}{N}\tilde{p}_0\left(\sum_{n=1}^{N}\tilde{H}^n\right)q_e^T. \tag{17}$$

Using the expected value of the architecture's mean $\mathbb{E}[\widetilde{Z}_N]$, it can be verified that $\tilde{Z}_N$ converges to $\max\{X, Y\}$. Specifically, in the Appendix, it is shown that, for $X, Y \in (0, 1)$ and $X \neq Y$, it is

$$\lim_{M\to\infty}\left(\lim_{N\to\infty}\mathbb{E}[\tilde{Z}_N]\right) = \begin{cases} X, & X > Y \\ Y, & X < Y. \end{cases} \tag{18}$$

D. Error Profile

We measure the accuracy of the proposed max's output and show the distribution of error for different inputs $X, Y \in [0, 1]$ using the mean absolute error (MAE) as

$$Z_{\text{error}} = \mathbb{E}\left|\tilde{Z}_N - \max\{X, Y\}\right| \tag{19}$$

where $\tilde{Z}_N$ is given by (16) and $X, Y \in [0, 1]$. The MAE is calculated numerically for pairs of i.i.d. input sequences (X, Y), while the simulation is performed 10^3 times for each pair considered. In Fig. 4, the MAE results are shown with sequence length $N = 64$ and a $m = 2$-bit register. One can observe that the error peaks when $X = Y = 0.5$ and gradually decreases when moving away from this pair values.

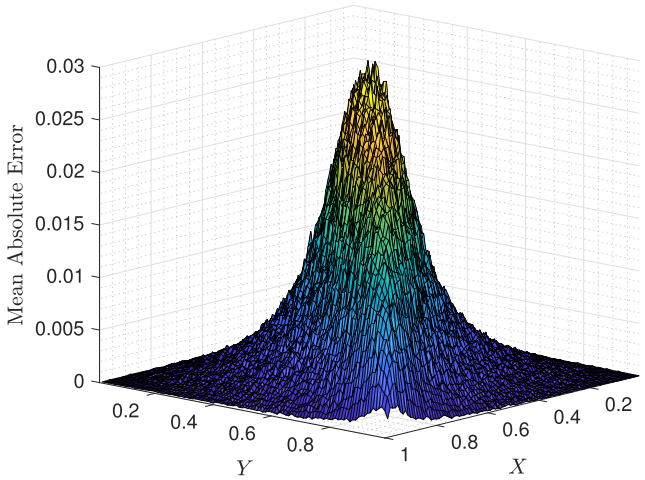

Fig. 4. MAE calculated using (19) parameterized with sequence length $N = 64$ and register size $m = 2$-bits. Simulations are performed for 10^3 runs on each pair (X, Y).

E. Register's Size and Overflow Markov Chain Model

According to the architecture of Fig. 1, the m-bit register counts the time indices for which $X_n < Y_n$. A potential case for the inputs is when $\{Y_n\}$ happens to have large segments of all ones, while, simultaneously, $\{X_n\}$ has large segments of all zeros. This means that, at each time index, the counter will increase its value by 1-bit, whereas, in the MC model of Fig. 2, the current state S_n will transition up to state $M - 1$.

The size of the m-bit register and, consequently, the M states in the MC model are both finite; further storing of 1s beyond their respective sizes is not feasible and results in overflow. To this end, it is important to investigate how the number of states M is related to overflows, as well as when overflows lead to erroneous bits in the output.

1) Overflow Calculation: To model the overflow occurrence, we modify the existing model of Fig. 2 into the one

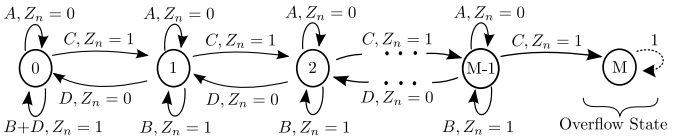

Fig. 5. Overflow MC model of the proposed stochastic max architecture. Absorbing state M represents the overflow. Transition probabilities A, B, C, and D are given by (4).

shown in Fig. 5, and note that it is preferred over the MC model of Fig. 3. This is because we focus on the MC's transitions within its states, and the simplicity of the MC model of Fig. 2 makes it suitable for this purpose.

The two models, in Figs. 2 and 5, are identical, except that the second one has one more state, M, which is absorbing. The purpose of this state is to capture the existence of a register's overflow. This occurs at a time index n when the state of the model in Fig. 2 is $S_n = M-1$, and the inputs are $X_n = 0$ and $Y_n = 1$, corresponding to probability C. In this case, the state of the model in Fig. 2 remains in $M-1$, whereas the one on the model in Fig. 5 is captured in M. It is important to note here that the extra state does not imply an increase in the register's size, and it is used for modeling purposes only.

To calculate the probability of overflow, we define first the set of states $\hat{\mathcal{S}} \triangleq \{0, 1, 2, \ldots, M\}$, which has $M+1$ values. The transition probability matrix $\hat{H} \in [0,1]^{(M+1)\times(M+1)}$ with state ordering $(0, 1, \ldots, M)$ is

$$\hat{H} = \begin{bmatrix} 1-C & C & 0 & \ldots & \ldots & 0 \\ D & A+B & C & 0 & \ldots & 0 \\ 0 & D & A+B & C & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & D & A+B & C \\ 0 & \ldots & \ldots & 0 & 0 & 1 \end{bmatrix}. \tag{20}$$

The probability vector of the state $\hat{S}_n$ is

$$\hat{p}_n^T \triangleq \begin{bmatrix} P_r(\hat{S}_n = 0) \\ P_r(\hat{S}_n = 1) \\ \vdots \\ P_r(\hat{S}_n = M) \end{bmatrix} \in [0,1]^{M+1} \tag{21}$$

and is expressed as

$$\hat{p}_n = \hat{p}_0 \hat{H}^n \in [0,1]^{M+1} \tag{22}$$

where the initial probability vector is

$$\hat{p}_0 = \big[1, 0, 0, \ldots, 0\big] \in [0,1]^{M+1}. \tag{23}$$

From (21), the probability that the register has overflowed *at least once* until time index $n \leq N$ is

$$\hat{P}_{\text{overflow}}(n) \triangleq P_r(\hat{S}_n = M) = \hat{p}_0 \hat{H}^n e_{M+1}^T \tag{24}$$

where $e_{M+1} = \big[0, \ldots, 0, 1\big] \in \mathbb{R}^{M+1}$.

In addition to (24), we use the model in Fig. 5 to calculate the expected number of transitions[1] before the first overflow

[1]i.e. the expected value of the time index n^* when the first overflow takes place.

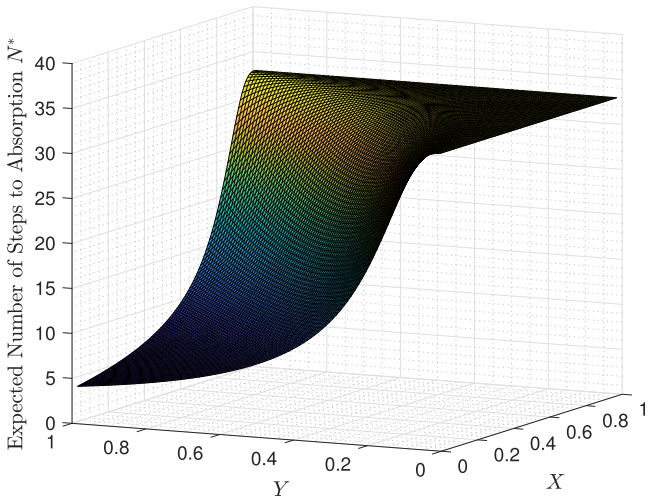

Fig. 6. Expected number of steps to absorption, calculated using (27), with stochastic sequence length $N = 32$ and number of states $M = 4$, parameterized on the inputs' expected values X and Y.

or, equivalently, the reach of the absorption state M. We start by writing matrix $\hat{H}$ in its canonical form [25], [26] as

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

where $\bar{H} \in [0,1]^{M\times M}$ and $R \in [0,1]^{M\times 1}$. Then, the fundamental matrix $F \in [0,1]^{M\times M}$ of the absorbing MC is

$$F = (I - \bar{H})^{-1} \tag{26}$$

where $I \in [0,1]^{M\times M}$ is the identity matrix. Since the initial state of the MC model in Fig. 5 is $\hat{S}_0 = 0$, the *average* number of transition steps to reach the absorbing state is given by [25], [26]

$$N^* = p_0 F \underline{1} \tag{27}$$

where p_0 is defined in (8) and $\underline{1} \in [0,1]^{M\times 1}$ is the column vector of ones.

Fig. 6 presents the expected number of steps before absorption calculated using (27). The stochastic sequence length is $N = 32$, the number of states is $M = 4$, and the input expected values $X, Y \in [0,1]$ are sweeped parametrically. As expected, N^* increases with X and decreases with Y, which is intuitively in agreement with the MC model in (24) and the transition probabilities in (4).

2) Errors Due to Overflow: Although the finite number of states may result in an overflow, an overflow does not necessarily imply an erroneous bit in the output Z_n. Consider, for example, the case when after the first overflow, the state S_n remains within $\mathcal{S} - \{0\}$, as shown in Fig. 2.

Now, consider the following extreme scenario. The state starts from $S_0 = 0$, $X_n = 0$, and $Y_n = 1$ for $n = 1, 2, \ldots, M-1$ implying $S_n = n$ and $n = 0, 1, 2, \ldots, M-1$. Then, again $X_M = 0$ and $Y_M = 1$ implying $S_M = M-1$ and an overflow. After that, assume the sequences $X_n = 1$ and $Y_n = 0$, $n = M+1, M+2, \ldots, 2M-1$ implying $S_n = 2M-n-1$ for the same n values. So far, the output is always $Z_n = 1$ independently of the overflow. Observe that $S_{2M-1} = 0$, and if the overflow had not occurred, e.g., if the states were $M+1$ instead of $M-1$,

TABLE I
MINIMUM REGISTER SIZE m-BIT ($M = 2^m$) SATISFYING $N^* \geq N$

Sequence length N-bits	16	32	64	128	256	512	1024
Register size m-bits	1	2	2	3	3	4	5

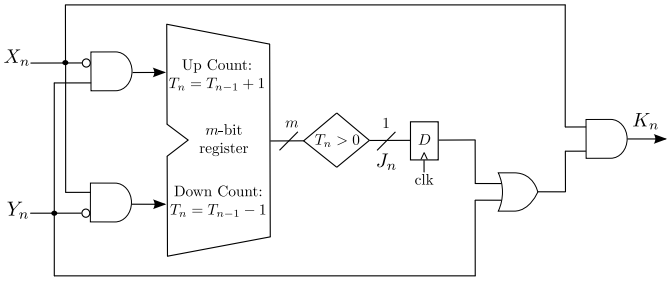

Fig. 7. Proposed stochastic min architecture. T_n is the m-bit register's state, updated according to (1).

then it would be $S_{2M-1} = 1$. Finally, assume that $X_{2M} = 1$ and $Y_{2M} = 0$, which implies that $Z_{2M} = 1$, whereas, if the overflow had not occurred, it would have been $Z_{2M} = 0$. One can easily prove that the above describes the shortest sequence (input and state) resulting in an erroneous output bit.

Therefore, the maximum sequence length N for which the *probability of error* is always zero, independently of the input values, is $N_{\text{errfree}} = 2M - 1$.

Although N_{errfree} is always desirable, in cases when N is required to be large, it will also force the register's states M to be large as well and, therefore, will increase the hardware resources. Instead, one can relax the error-free requirement allowing for overflows that may result in a few incorrect output bits. In this direction, we can use the expression of N^* in (27) to select M (note that M defines the size of the matrices involved).

Fig. 4 shows that the highest MAE appears when $X = Y = 0.5$. This relates to the appearance of overflows converted into erroneous output bits. Assuming that $X = Y = 0.5$, we first select the value of N, and then, we choose $M = 2^m$ to be the smallest power of 2 resulting in $N^* \geq N$. The values of the derived register's sizes m are cited in Table I parameterized on N.

IV. STOCHASTIC MIN ARCHITECTURE AND ANALYSIS

In this section, we follow the stochastic MAX's principles to introduce the proposed stochastic MIN architecture and its mathematical analysis using MCs.

A. Architecture

The proposed min architecture is shown in Fig. 7. Again, the m-bit register is used to count the number of cases $Y_n > X_n$ minus the number of cases $Y_n < X_n$. Therefore, the accumulator's current value T_n starts from $T_0 = 0$ and belongs in the set $\mathcal{T}_R \triangleq \{0, 1, 2, \ldots, M-1\}$, which has a total of $M = 2^m$ states and is updated according to (1). Similar to the max architecture, states 0 and $M-1$ constrain the values' range of T_n.

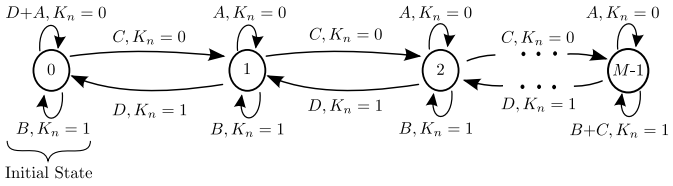

Fig. 8. MC model of the proposed stochastic min architecture. Output K_n is determined by the state's transition according to transition probabilities A, B, C, and D given by (4).

In contrast to the max architecture, the output K_n here is determined as follows: if X_n and Y_n are both 0 or 1, then K_n has the same value 0 or 1, respectively; if $Y_n > X_n$, then K_n always outputs 0; and if $Y_n < X_n$, then $K_n = 1$ if and only if the register's previous value was $T_{n-1} > 0$, and $K_n = 0$ otherwise. Summarizing the former cases and also considering the architecture in Fig. 7 and the definition $J_n = T_n > 0$, the instantaneous output K_n is expressed as

$$K_n = X_n\big(Y_n + J_{n-1}\big). \tag{28}$$

B. Markov Chain Modeling

The operation of the proposed min architecture is modeled using the MC model in Fig. 8. The MC's current state S_n transitions within its M states in the set $\mathcal{S}$ given by (3), while its probability distribution vector after N steps is calculated using (4), (5), (7), and (8).

1) Extended Markov Chain Model: The MC model in Fig. 8 is converted to that in Fig. 9, which allows to relate its current state $\tilde{S}_n$ to the output K_n by classifying the model's states into the subsets $\mathcal{S}_a$ and $\mathcal{S}_b$ that always output $K_n = 1$ and $K_n = 0$, respectively. Furthermore, $\tilde{S}_n$ transitions within $2M$ states in the set given by (3), with initial value $\tilde{S}_0 = 0_a$. Assuming the states' ordering $\big(0_a, 0_b, \ldots, (M-1)_a, (M-1)_b\big)$, the transition probability matrix of the MC model is given by the following equation where we have defined $U \triangleq B + C$ and $W \triangleq D + A$:

$$\tilde{H} = \begin{bmatrix}
B & W & 0 & C & 0 & & & \ldots & & & 0 \\
B & W & 0 & C & 0 & & & \ldots & & & 0 \\
D & 0 & A & B & 0 & C & 0 & \ldots & & & 0 \\
D & 0 & A & B & 0 & C & 0 & \ldots & & & 0 \\
0 & 0 & D & 0 & A & B & 0 & C & 0 & \ldots & 0 \\
0 & 0 & D & 0 & A & B & 0 & C & 0 & \ldots & 0 \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & & & \ldots & & 0 & D & 0 & A & B & 0 & C \\
0 & & & \ldots & & 0 & D & 0 & A & B & 0 & C \\
0 & & & \ldots & & & & 0 & D & 0 & A & U \\
0 & & & \ldots & & & & 0 & D & 0 & A & U
\end{bmatrix}. \tag{29}$$

C. First-Order Statistics, Proof of Operation, and Error Profile

The expected value of the instantaneous output K_n is

$$\mathbb{E}[K_n] = P_r(K_n = 1) = P_r\big(\tilde{S}_n \in \mathcal{S}_a\big) = \tilde{p}_0 \tilde{H}^n q_o^T \tag{30}$$

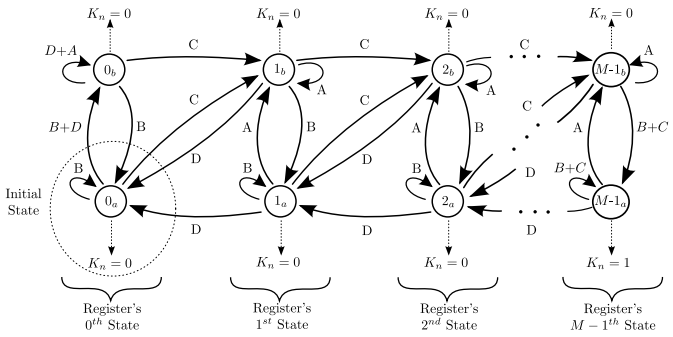

Fig. 9. Extended MC model of the proposed stochastic min with transition probabilities given by (4). Each register state is represented by two states: upper one outputs $K_n = 0$ and lower one outputs $K_n = 1$. Subscripts a and b denote in which set $\tilde{S}_n$ is currently into.

where we used (13) and (29), and q_o is defined as

$$q_o \triangleq [1, 0, 1, 0, \ldots, 1, 0] \in [0, 1]^{2M}. \tag{31}$$

The average of the output N-bit sequence is

$$\widetilde{K}_N = \frac{1}{N}\Big(K_1 + K_2 + \cdots + K_N\Big) \tag{32}$$

and its expected value, using (30), is given by

$$\mathbb{E}[\widetilde{K}_N] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[K_n] = \frac{1}{N}\tilde{p}_0\left(\sum_{n=1}^{N}\tilde{H}^n\right)q_o^T. \tag{33}$$

Finally, the proof of operation of the min architecture is similar to that of the max one given in the Appendix. Also, it can be verified that the min's MAE distribution follows the same behavior as the max's one in Fig. 4.

D. Register's Size and Overflow Markov Chain Model

The up- & down counting in the m-bit register follows the same principles as in the max case. Similarly, one can add an absorbing state in the MC model of Fig. 8 and follow the procedure described in Subsection III-E to calculate the probability of overflow and to derive the register's size based on accuracy requirements. The results are in agreement with Table I.

V. Comparison of the Proposed Architectures to Stochastic Computing Literature

In this section, we compare the proposed max and min architectures with existing ones focusing on SC-based implementations widely used in the SC literature [3], [7], [21], [23]. To compare the accuracy of the proposed and the rest architectures, we use the MAE as the performance metric. We assume that inputs X, Y are i.i.d. and perform 10^3 random runs on each point (X, Y) that we examine. Then, we calculate the average value over all points for stochastic sequence lengths $N = 2^k$ with $k = 4, 5, \ldots, 10$. We note that, except for the proposed architectures where the register size is derived analytically according to Section III-E, for the rest, we have selected the register size that yields the highest accuracy possible based on simulations according to N. The results are demonstrated in Fig. 10 and are cited in Table II.

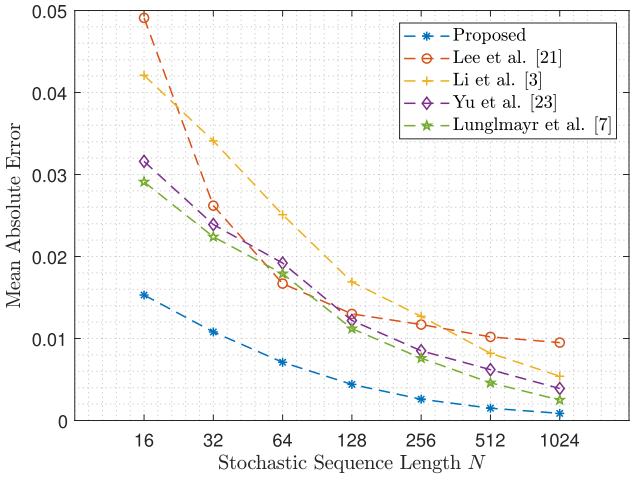

Fig. 10. Accuracy comparison in MAE of stochastic max architectures for typical sequence lengths N. For each N, the architectures' number of states is selected to result in the highest computational accuracy.

TABLE II
Comparison of Accuracy in MAE

Mean Absolute Error (MAE) $\times 10^{-2}$

$N = 2^k$	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Proposed	1.53	1.08	0.71	0.44	0.26	0.15	0.08
Register m-bit	1	2		3		4	5
Lee et al. [21]	4.91	2.62	1.67	1.30	1.11	1.02	0.95
Register m-bit	2						
Li et al. [3]	4.21	3.41	2.51	1.69	1.27	0.82	0.54
Register m-bit	2		3			4	5
Yu et al. [23]	3.16	2.39	1.92	1.22	0.85	0.62	0.39
Register m-bit	2			3		4	5
Lunlgmayr et al. [7]	2.91	2.24	1.79	1.11	0.76	0.46	0.25
Shift Register m-bit	2			3	4	5	6

For the hardware requirements of each architecture, we first synthesized all designs using Verilog HDL in Xilinx's Vivado Design Suite targeting the Kintex-7 KC705 Field-Programmable Gate Array (FPGA) kit, so as to verify their proper operation. Then, we used the Synopsys Design Compiler with the FreePDK CMOS Library at 45 nm [27] to extract the area, critical path, power, and energy consumption metrics. For each comparison, we provide estimates of: 1) the total area in squared micrometer; 2) the average power consumption in mW for the max operating frequency; 3) the critical path in ns; and 4) the energy (average power $\times$ critical path) per operation in pJ. The detailed results per operation are cited in Table III, while Fig. 11 presents comparisons for the energy and the power $\times$ delay^2 product for N clock cycles.

At this point, we note that the max architectures in [3], [7], [21], and [23], including the proposed one, are able to output the min as well without affecting the total hardware resources, i.e., introducing additional logic units or registers. Therefore, the presented accuracy and hardware resource metrics for the max architectures apply to the min architectures as well.

It is worth mentioning for the proposed architectures that their critical path is the minimum time required for the state T_n to be updated, which is described by (1). In our case,

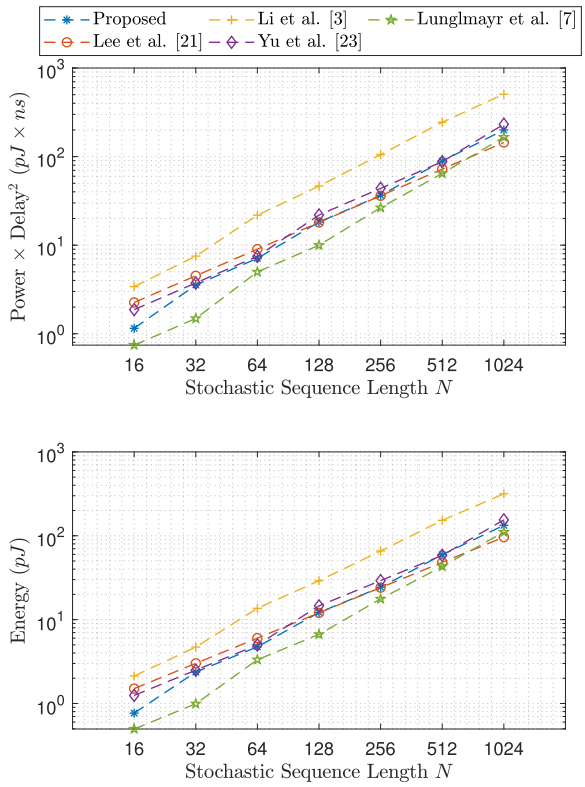

Fig. 11. Comparison of Power × Delay^2 (pJ × ns) (top) and Energy pJ (bottom) consumption. For each N, the architectures' number of states is selected to result in the highest computational accuracy.

TABLE III

COMPARISON OF HARDWARE RESOURCES IN AREA (μm^2), CRITICAL PATH (ns), POWER (mW), AND ENERGY (pJ) CONSUMPTION PER OPERATION

	Register m-(bit)	Area (μm^2)	Power (mW)	Critical path (ns)	Energy (pJ)
Proposed	$m=1$	37.54	0.032	1.5	0.048
	$m=2$	60.86	0.049		0.074
	$m=3$	88.69	0.063		0.095
	$m=4$	106.24	0.077		0.116
	$m=5$	119.21	0.086		0.130
Lee et al. [21]	$m=2$	91.49	0.062	1.5	0.094
Li et al. [3] MUX LFSR size k	$m=2, k=4$	109.81	0.083	1.6	0.133
	$m=2, k=5$	131.87	0.092		0.147
	$m=3, k=6$	176.92	0.133		0.213
	$m=3, k=7$	199.45	0.142		0.227
	$m=3, k=8$	236.32	0.161		0.257
	$m=4, k=9$	291.90	0.184		0.295
	$m=5, k=10$	311.61	0.193		0.309
Yu et al. [23]	$m=2$	48.01	0.052	1.5	0.078
	$m=3$	61.47	0.076		0.114
	$m=4$	104.97	0.101		0.151
Lunglmayr et al. [7] Shift Register	$m=2$	46.46	0.021	1.5	0.031
	$m=3$	57.25	0.034		0.052
	$m=4$	73.21	0.046		0.069
	$m=5$	89.16	0.057		0.084
	$m=6$	105.12	0.068		0.103

this value dominates the FFs' propagation delay for register sizes up to $m = 5$ bits, as shown in Table III, justifying the constant value as well.

1) Comparison With Lee et al. Method in [21]: The core of the architecture is a three-state FSM that forces the overlap of logic ones between its two input i.i.d. sequences $\{X_n\}$ and $\{Y_n\}$, to produce two correlated outputs. These are used as inputs to an OR gate to produce the final output. If the OR gate is replaced by an AND in the architecture, then the min can be realized.

According to Fig. 10, the proposed architecture has better accuracy regardless of the sequence length N used, and this is intensified especially for smaller values of N. Hardware-wise, the proposed architecture occupies less area and consumes less power and energy for register sizes $m = 1, 2$, similar for $m = 3$, while, for $m = 4, 5$ Lee *et al.*'s method [21] is slightly better. However, the fact that Lee *et al.*'s approach in [21] requires more clock cycles to achieve the same accuracy as the proposed architecture should not be neglected; the increased latency implies a further increase in dissipated energy, exceeding the proposed one's.

2) Comparison With Li et al.'s Method in [3]: The inputs of this architecture are fed to an MUX that uses an SNG as its select signal. The MUX's stochastic output is the input of the stochastic tanh function, implemented as an FSM of 2^m states (m-bits), while the FSM's output is determined by the current state; starting from the zero states, the first $2^m/2 - 1$ outputs 0, while the rest outputs 1. The FSM's output is also used as a select signal in an MUX that determines whether X_n or Y_n to be the architecture's current output.

From Fig. 10, the proposed architecture results in better computational performance in terms of accuracy. The increased performance of the proposed architecture also applies to the hardware utilization, as shown in Table III, which is due to the additional SNG used, contributing negatively to the total area, power, and energy consumption. Moreover, an important design aspect is the register's size. As stated in [3], increasing its size and, hence, the number of its states, the computational accuracy increases as well. Yet, the selection of its size that yields the highest accuracy is estimated with numerical simulations. On the other hand, the guidelines to select the register's size in the proposed architecture eliminate the parametric simulation time completely.

3) Comparison With Yu et al.'s Method in [23]: To avoid the power and area hungry SNG from Li *et al.*'s method in [3], the architecture by Yu *et al.* [23] uses an XOR between the two inputs instead, which acts as an enable signal to up-count logic 1s coming from its input X_n. The counting is based on the stochastic tanh FSM, implemented in the same way shown by Li *et al.* in [3]. Consequently, the tanh's output is used as a select signal in an MUX that determines if X_n or Y_n is the output.

Accuracy-wise, the proposed architecture results in better computational results, as shown in Fig. 10. In terms of hardware resources, the proposed architecture occupies a larger area but has reduced power and energy consumption when the same register size is used according to Table III. From a designer's perspective, the register size that maximizes the accuracy of Yu *et al.*'s architecture in [23] is derived with simulations. If not chosen carefully based on the sequence length N, it directly affects the output's accuracy; by reducing the number of its states, it will increase the output's error.

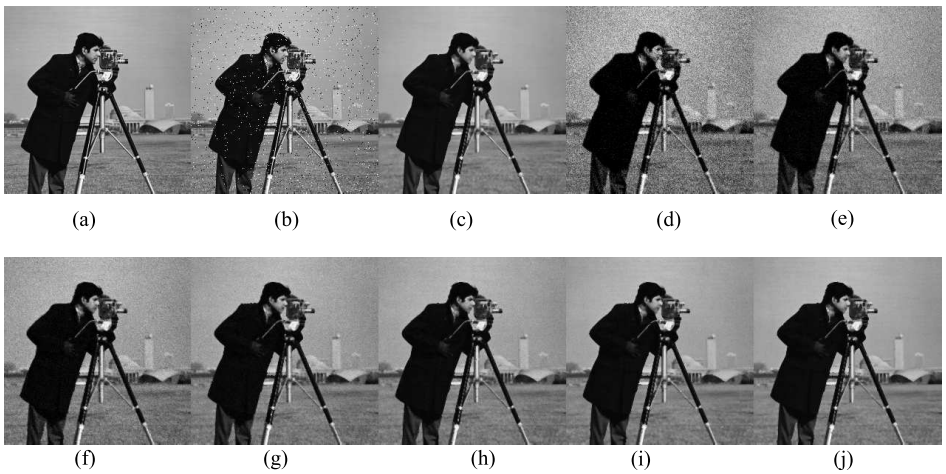

Fig. 12. Median filtering with a 3×3 kernel for various sequence lengths N. From upper left to lower right: (a) MATLAB's original image, (b) MATLAB's noisy image with salt & pepper noise density 0.02, (c) MATLAB's filtered image, (d) $N = 16$, (e) $N = 32$, (f) $N = 64$, (g) $N = 128$, (h) $N = 256$, (i) $N = 512$, and (j) $N = 1024$. Register size used is $m = 2$ corresponding to $M = 4$ states.

On the contrary, the analytic derivation of the proposed max's register size provides insight on its design.

4) Comparison With Lunglmayr et al.'s Method in [7]: In this architecture, motivated by Yu *et al.*'s method in [23], an XOR between the inputs is used as an enable signal in a linear FSM, implemented as a shift register of m-bits (can also be implemented as a binary counter). The FSM performs a right shift of the most significant bit (MSB) if $X_n = 1$, whereas a left shift is if $X_n = 0$. The FSM's output is determined by the LSB of the register and produces 1 if it has saturated up to the LSB. Finally, an MUX selects either the FSM's output or Y_n along with additional logic gates.

From the comparison with the proposed architecture shown in Fig. 10, the approach by Lunglmayr *et al.* [7] results in lower computational accuracy. However, the power and energy consumption per clock cycle is its strong point, which is due to the advantage of the shift register over the binary one, as shown in Table III. Yet, the architecture's output accuracy depends on the saturation of the shift register up to its LSB. If its size is not chosen accurately, for instance, if it is less or more than a specific value, the output's accuracy can be greatly reduced, and this is also shown in [7]. We note that the register size that results in the highest accuracy possible is used in the simulations and is also cited in Table II, taken from [7].

VI. Image Processing Application: Median Filter

In this section, we demonstrate the effectiveness of the proposed max and min architectures in a standard digital image processing application. We use them as building blocks to implement a 3×3 median filter, which is typically used to reduce noise from images [28]. The kernel's structure is based on the sorting network presented in [3].

We first select a gray-scale image with 8-bit representation for each pixel and inject salt & pepper noise with a noise density of 0.02. Afterwards, we normalize the pixel values to range $[0, 1]$ so as to be processed in the SC domain. Given the fact that the accuracy of the architecture is based on the stochastic sequence length $N = 2^k$, we use typical values of $k = 4, 5, \ldots, 10$ and investigate their effect in computational accuracy with simulations using MATLAB.

TABLE IV
Accuracy in PSNR of the Realized 3×3 Median Filter Using the Proposed Max and Min Architectures

Peak-Signal-to-Noise Ratio (PSNR) dB							
$N = 2^k$	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Proposed	20.64	23.16	25.55	27.75	29.64	31.05	32.11

A graphical illustration of the computations using the proposed architectures to implement the median filter is shown in Fig. 12, while their respective accuracy results evaluated with the peak signal-to-noise ratio (PSNR) in dB are cited in Table IV. From sequence lengths $N = 64$ and forth, the proposed approach provides with a sufficient approximation of the median filter's computation, supported by the PSNR of 25.55 dB as well.

To proceed with the hardware resources, we note that the selected register size for each max and min that we utilize is $m = 2$ as it does not degrade the computational accuracy and holds for all the lengths N used in this specific application. In Table V, the comparison between the traditional binary method using 8 bits is shown. Both designs are synthesized first using Verilog HDL in Xilinx's Vivado Design Suite using the Kintex-7 KC705 platform to verify their operation and then their hardware requirements are extracted using the Synopsys Design Compiler with the FreePDK CMOS library at 45 nm [27]. It is important to note that, since the SNG's LFSR size determines N, it also affects the overall hardware utilization. Moreover, given the fact that N is a parameter selected according to the accuracy requirements, the SNG's hardware resources are, therefore, not included in Table V. Yet, their design can be optimized according to [13].

According to the results, the proposed approach occupies almost half the binary method's area, which is its strong point. Depending on the required accuracy from the sequence length

TABLE V
HARDWARE RESOURCES FOR THE IMPLEMENTATION OF A 3×3 MEDIAN FILTER USING THE PROPOSED MAX AND MIN ARCHITECTURES 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	1,139	0.534	2.0	1.068
Binary 8-bit	2,470	2.295	2.2	5.049

N, the hardware efficiency follows accordingly. For instance, for $N = 64$, which provides acceptable results, the total energy consumption has moderate values compared to the binary ones corresponding to 68.35 pJ.

VII. CONCLUSION

This work presented a max & min architecture for SC. Analytic modeling with MCs allowed describing their stochastic operation in detail and deriving guidelines to achieve overall design optimization. Compared with state-of-the-art architectures selected from the SC literature, it was shown that the proposed ones improve the latency–accuracy tradeoff by combining fast-converging and highly accurate computations. Finally, the realization of a 3×3 median filter using the proposed stochastic max and min architectures and its successful application in an image denoising task demonstrated its effectiveness.

APPENDIX

To prove (18), we assume that $0 < X, Y < 1$ and $X \neq Y$, which implies that $0<A, B, C, D<1$ and $\rho \neq 1$, where

$$\rho \triangleq \frac{C}{D} = \frac{(1-X)Y}{(1-Y)X}. \tag{34}$$

By inspecting the MC model of Fig. 3, one can observe that the chain is *irreducible* since every state is accessible from every other one, and so the transition matrix $\tilde{H}$ is also irreducible.

Let $v^T = [v_1, v_2, \ldots, v_{2M}]^T \in \mathbb{R}^{2M}$ be the left eigenvector of $\tilde{H}$, i.e., $v^T \tilde{H} = v^T$, corresponding to eigenvalue 1, and be normalized such that $v^T \underline{1} = 1$, where $\underline{1} = [1, 1, \ldots, 1]^T \in \mathbb{R}^{2M}$ is a column vector of ones. Then, it can be verified that

$$\begin{aligned} v_1 &= A w_1 + D w_2 \\ v_2 &= F w_1 \\ v_{2k-1} &= A w_k + D w_{k+1} \\ v_{2k} &= C w_{k-1} + B w_k \\ v_{2M-1} &= A w_M \\ v_{2M} &= C w_{M-1} + U w_M \end{aligned} \tag{35}$$

where $k = 2, 3, \ldots, M-1$ and w_k is given by

$$w_k = \lambda \rho^{k-1}, \qquad k = 1, 2, \ldots, M \tag{36}$$

with

$$\lambda \triangleq \frac{\rho - 1}{\rho^M - 1}. \tag{37}$$

Since the transition matrix H is irreducible, from [29, Th. 8.6.1], it is $\lim_{N\to\infty}(1/N)\sum_{n=1}^{N} \tilde{H}^n = \underline{1} v^T$. Combining it with (17), we get

$$\lim_{N\to\infty} \mathbb{E}[\tilde{Z}_N] = \tilde{p}_0 \underline{1} v^T q_e^T = v^T q_e^T = \sum_{k=1}^{M} v_{2k}. \tag{38}$$

From (35) and (36), we have

$$\begin{aligned} v_2 &= F\lambda \\ v_{2k} &= \lambda(C + B\rho)\rho^{k-2} \\ v_{2M} &= \lambda(C + U\rho)\rho^{M-2} \end{aligned} \tag{39}$$

resulting in

$$\sum_{k=1}^{M} v_{2k} = \lambda \left\{ F + (C + B\rho)\frac{\rho^{M-2} - 1}{\rho - 1} + (C + U\rho)\rho^{M-2} \right\}. \tag{40}$$

Combining the above and taking the limit when $N, M \to \infty$, we get

$$\lim_{M\to\infty}\left(\lim_{N\to\infty} \mathbb{E}[\tilde{Z}_N]\right) = \lim_{M\to\infty}\left(\sum_{k=1}^{M} v_{2k}\right) = \begin{cases} X, & X > Y \\ Y, & Y > X. \end{cases} \tag{41}$$

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Z. Wang, D. Larso, M. Barker, S. Mohajer, and K. Bazargan, "Deterministic shuffling networks to implement stochastic circuits in parallel," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 8, pp. 1821–1832, Aug. 2020.

[6] 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.

[7] M. Lunglmayr, D. Wiesinger, and W. Haselmayr, "Design and analysis of efficient maximum/minimum circuits for stochastic computing," *IEEE Trans. Comput.*, vol. 69, no. 3, pp. 402–409, Mar. 2020.

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

[9] B. R. Gaines, *Stochastic Computing Systems*. Boston, MA, USA: Springer, 1967.

[10] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embedded Comput. Syst.*, vol. 12, no. 2, pp. 1–19, May 2013.

[11] W. Qian, M. D. Riedel, H. Zhou, and J. Bruck, "Transforming probabilities with combinational logic," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 9, pp. 1279–1292, Sep. 2011.

[12] 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.

[13] M. Yang, B. Li, D. J. Lilja, B. Yuan, and W. Qian, "Towards theoretical cost limit of stochastic number generators for stochastic computing," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Hong Kong, Jul. 2018, pp. 154–159.

[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] V. T. Lee, A. Alaghi, J. P. Hayes, V. Sathe, and L. Ceze, "Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Laussane, Switzerland, Mar. 2017, pp. 13–18.
[19] P. Li and D. J. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *Proc. IEEE 29th Int. Conf. Comput. Design (ICCD)*, Amherst, MA, USA, Oct. 2011, pp. 154–161.
[20] A. Alaghi, C. Li, and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *Proc. IEEE 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, May 2013, pp. 1–6.
[21] V. T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2018, pp. 1417–1422.
[22] 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.
[23] J. Yu, K. Kim, J. Lee, and K. Choi, "Accurate and efficient stochastic computing hardware for convolutional neural networks," in *Proc. IEEE 35th Int. Conf. Comput. Design*, Boston, MA, USA, Nov. 2017, pp. 105–112.
[24] A. Alaghi and J. P. Hayes, "Fast and accurate computation using stochastic circuits," in *Proc. IEEE Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2014, pp. 105–112.
[25] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed. Providence, RI, USA: American Mathematical Society, 1997.
[26] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, 1st ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.
[27] J. E. Stine *et al.*, "Freepdk: An open-source variation-aware design kit," in *Proc. IEEE Int. Conf. Microelectron. Syst. Educ.*, San Diego, CA, USA, Jun. 2007, pp. 173–174.
[28] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*, 2nd ed. Knoxville, TN, USA: Gatesmark Publishing, 2009.
[29] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 1990.

Nikos Temenos (Student Member, IEEE) received the B.Sc. degree in computer and systems engineering from 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 and co-authored several IEEE conferences. His main research area includes digital VLSI design, computer arithmetic, and 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 Electrical and Computer Engineering Department, 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 Assistant Professor of electrical and computer engineering. In 2012, he joined the Faculty of the Electrical and Computer Engineering Department, National Technical University of Athens. He is currently a Faculty Member of the Electrical and Computer Engineering Department, National Technical University of Athens, the Director of the Electronics Laboratory of the University, and a member of the Governing Board of the Hellenic Space Center, National Space Center of Greece, 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 sensors systems. 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 received several awards, including the 2012 Guillemin-Cauer Award from the IEEE Circuits and Systems Society, the Best Paper Award in the IEEE International Symposium on Circuits and Systems 2007, the Best Paper Award in the IEEE International Frequency Control Symposium 2012, the Best Paper Award in the IEEE International Conference on Modern Circuits and Systems Technologies 2019, and the Best Paper Award in the International Conference on Microelectronics 2020. He has served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS from 2016 to 2020 and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS from 2005 to 2010. He is also an Associate Editor of IEEE SENSORS JOURNAL. He regularly reviews for many IEEE transactions and conferences and serves on proposal review panels.