INTEGRATION, the VLSI journal 87 (2022) 194–204

Contents lists available at ScienceDirect

Integration

journal homepage: www.elsevier.com/locate/vlsi

Compact MAX and MIN Stochastic Computing architectures

Paul P. Sotiriadis, Nikos Temenos *

Department of Electrical and Computer Engineering, National Technical University of Athens, 15780 Athens, Greece

ARTICLE INFO

Keywords:
Stochastic Computing
Stochastic MAX
Stochastic MIN
Unconventional computing

ABSTRACT

In this work we present Stochastic Computing MAX and MIN architectures. Their operation relies on an accumulator to store the signed-bit differences between their two stochastic input sequences without additional randomization. This counting process makes their operation deterministic, resulting in an improved latency-accuracy trade-off when compared to existing Stochastic Computing MAX and MIN architectures. Modeling the architectures as Markov Chains allows for an in-depth analysis of their stochastic operation, derivation of their statistical properties and proof of their correct operation. An overflow/underflow Markov Chain model allows for the analytic calculation of the register size they use, providing guidelines for its selection based on accuracy requirements. The performance of the proposed architectures is compared to those of existing ones in the Stochastic Computing literature in computational accuracy and hardware resources using MATLAB and Synopsys Tools. Their effectiveness is demonstrated in two standard Digital Image Processing tasks; image denoising with a 3×3 median filter and dimensionality reduction of an image with a 2×2 max pooling kernel.

1. Introduction

Efficient realization of digital systems in Integrated Circuits (ICs) and Field Programmable Gate Arrays (FPGAs) is of primary interest due to the accelerated growth of emerging applications [1–4]. Standard Digital Signal Processing (DSP) cores are often stressed by the hardware-demanding binary computing methods, especially when parallelization is necessary [3,5–7]. To this end, research has shifted towards unconventional computing paradigms to overcome the binary computing's design constraints.

Stochastic Computing (SC) falls within the category of unconventional computing techniques and is proven to be an effective approach [3,6,8,9]. Deviating from the standard binary arithmetic representations and processing, SC encodes numbers and signals probabilistically in the form of stochastic sequences [10]. Therefore, its single-bit processing allows for the realization of fundamental arithmetic operations as well as highly-complex functions using only a few standard logic cells [6,9,11,12]. Moreover, SC is inherently tolerant to soft-errors [6,10,13].

One of SC's main challenges is the latency to accuracy trade-off [8,14]. Given its bit-processing, increasing the computational cycles results in increased accuracy of the processed sequences, at the cost, however, of energy consumption [12]. Therefore, the increased computational accuracy combined with low-latency is of primary design concern in SC.

SC's advantages favor applications with massive parallelism needs. Neural Networks (NNs) [3,4,8,15–19] and Digital Image Processing [20–22] are two of the many fields in which SC is successful, others being soft-filtering & polynomial solving [23–28], error-correcting codes [9,29], etc. With emphasis in NNs and Image Processing tasks, necessary part of their processing cores includes non-linear functions, which can be realized effectively in SC as stochastic Finite State Machines (FSMs). They are known to realize widely used functions such as the tanh, the exponential, the linear-gain, the MAX & MIN and others [16,20,21,30]. Among them, the MAX & MIN is the most popular one [3], due to its importance in MAX pooling operations and in image filtering kernels, such as the median.

Several MAX & MIN architectures have been investigated within the context of SC [21,30–32]. The core of the approach in [21] is multiplexers (MUXs) and the stochastic tanh function, implemented as a FSM. The selection of the FSM's number of states is of critical importance as it is one of the two factors determining the output accuracy, the other being the sequence length. In [21], the FSM's number of states resulting in the highest computational accuracy are derived with numerical simulations. Furthermore, one MUX requires a binary-to-stochastic converter to generate its select signal, which is a hardware taxing block, increasing the total area, power & energy consumption.

To reduce the hardware overhead in [21], the approach in [31], replaces the binary-to-stochastic converter with an XOR gate, preserving

* Corresponding author.
E-mail address: ntemenos@gmail.com (N. Temenos).

https://doi.org/10.1016/j.vlsi.2022.06.006
Received 22 November 2021; Received in revised form 17 May 2022; Accepted 17 June 2022
Available online 4 July 2022

the rest of its processing elements including the stochastic tanh FSM.

Another recently introduced method for the MAX & MIN is presented in [30]. In this architecture, the FSM-based stochastic tanh is replaced by a shift-register that stores bits from one of its inputs, producing a logic 1 if it has saturated up to its least significant bit. The accuracy of the architecture's output is determined by the size of the shift-register, which is derived with numerical simulations based on the input sequence length.

The operation of the architecture presented in [32] deviates from the previous ones; it correlates first its input sequences using a three state FSM and then uses a single gate to produce either the max or the min. However, the FSM's fixed number of states limits the output sequence's accuracy, as it allows for only a few uncorrelated bits to be stored.

The common factor that the above architectures share, is the derivation with numerical simulations of the FSM's number of states, according each time to the stochastic input sequence length used. If not carefully selected, the register realizing the FSM will overflow and may cause bit-errors in the output sequence, reducing therefore its computational accuracy. As such, the register's size plays an important factor in the design of MAX & MIN architectures, considering that their computational accuracy may affect other processing blocks.

Motivated by the needs for the FSM's analytic design as well as the necessity to improve the latency-accuracy trade-off in SC, in this work we explore a different approach for the MAX & MIN; we use an accumulator to store the signed bit-differences between the input sequences, while the architecture is not affected by additional randomizing sources. The architectures' properties are demonstrated by modeling them as Markov Chains (MCs), allowing for: (1) the detailed analysis of their operating principles, (2) the derivation of the output's first order statistics and their proof of proper operation, (3) the analytic calculation of the probability of overflows and underflows and, (4) the selection of the register's size based on the stochastic input sequence length.

The remainder of this work is organized as follows. In Section 2, we provide with a background on the mathematical properties of stochastic numbers. In Section 3, we introduce the proposed stochastic MAX architecture and its detailed analysis using MCs. Based on the MAX architecture, in the same section we also present the proposed MIN architecture. In Section 4, we present an in-depth analysis of the register's stochastic behavior as well as provide with design guidelines to select its size based on the output sequence length. In Section 5, we compare the proposed architectures with existing ones from the SC literature, in computational accuracy and hardware requirements and discuss the results. In Section 6, we demonstrate the effectiveness of the proposed architectures in two image processing tasks; (i) the realization of a 3×3 median filter and its use in image de-noising and (ii) the down sampling of an image using a 2×2 max pooling kernel. Finally, in Section 7, we conclude our work.

2. Stochastic number representation

The Stochastic Number Generator (SNG) shown in Fig. 1, is the standard block used to encode a binary number into a stochastic sequence of logic 1s and 0s [6,10]. Its operation is based upon the comparison (on each clock cycle) of a k-bit binary word $B \in [0,1]$ with the value of a k-bit Linear-Feedback Shift Register (LFSR). It is important to note here that by definition, a LFSR can cycle through $\mathscr{R} = \{1, \ldots, 2^k - 1\}$ only once without repetitive values that may introduce correlations. Here, we consider $N = 2^k$-bit stochastic sequences and we assume that a repetition of the LFSR's initial value, does not degrade the result of the computations. Finally, to convert the stochastic number back to its binary form, an up-counter of k-bits is used.

Assuming that the LFSR's values are uniformly distributed in $\mathscr{R}$, the generated by the SNG N-bit output sequence, is independent and identically distributed (IID), i.e. $\{X_n\}, n = 1, 2, \ldots, N$, with n being the time index (or clock cycle). The stochastic number's value is a non-negative number in $[0, 1]$, known as *unipolar format* in SC, has probability defined as $X \triangleq P_r(X_n = 1)$ and time-average value

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

To represent negative numbers, known as *bipolar format* in SC, one can use the transformation $X \leftarrow 2X - 1$, which expands the stochastic number's value to range $[-1,1]$. As expected, in both formats the length of the stochastic sequence N plays an important role in the accuracy of the stochastic number, which is increased at the cost of additional clock cycles. For the remainder of this work, we use the above stochastic number properties to explain the principle operation of the proposed architectures.

3. Stochastic computing MAX architecture

Fig. 2 shows the proposed stochastic MAX architecture where $\{X_n\}$ and $\{Y_n\}$ are the stochastic input sequences, assumed to be generated by SNGs, and $\{Z_n\}$ is the output. Its operation is based on increasing the m-bit register's current value T_n by 1 if $X_n > Y_n$ and decreasing it by 1 if $X_n < Y_n$, within the set $\mathscr{T}_R \triangleq \{0,1,2,\ldots,M-1\}$, starting from $T_0 = M/2$, where $M = 2^m$ is the number values. Essentially, the register counts the number of *signed* bit-wise differences between its two inputs, X_n and Y_n. We can express the update of the register's value as

$$T_n = max\{min\{T_{n-1} + X_n - Y_n, M-1\}, 0\}, \tag{2}$$

where the *min* and *max* functions imply the natural *saturating* behavior of the counter since values 0 and $M-1$ cannot be exceeded.

To derive the output Z_n, we define first the result of the comparison between the register's current value T_n and the reference value $M/2$ as

$$J_n = \begin{cases} 0, & \text{if } T_n < M/2 \\ 1, & \text{if } T_n \geq M/2, \end{cases} \tag{3}$$

which, following Fig. 2, implies that

$$Z_n = J_n X_n + \overline{J}_n Y_n, \tag{4}$$

where $\overline{J}_n = 1 - J_n$ (considering 0 and 1 as Real numbers). Note that $J_n = 1$ means that input sequence X_n has had more 1s than Y_n had, within the storing range of the register. In this case, the output is $Z_n = X_n$ as expected, whereas if $J_n = 0$ it is $Z_n = Y_n$.

Although the input sequences are stochastic, the architecture's operation is deterministic, modeled by Eqs. (2)–(4), and the output Z_n is a function of T_n, X_n and Y_n. These imply that the accuracy of $\{Z_n\}$ depends on: (1) the size, m, of the register, and (2) the length, N, of the input sequences.

3.1. Markov Chain modeling

To investigate the stochastic behavior of the proposed MAX architecture we model it as the Markov Chain (MC) shown in Fig. 3. The MC has the M states in the set given by Eq. (5)

$$\mathscr{S} \triangleq \{0, 1, \ldots, M-2, M-1\}, \tag{5}$$

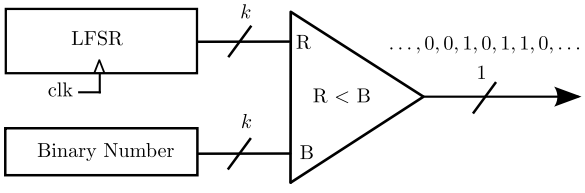

Fig. 1. Stochastic Number Generator (SNG) circuit [6].

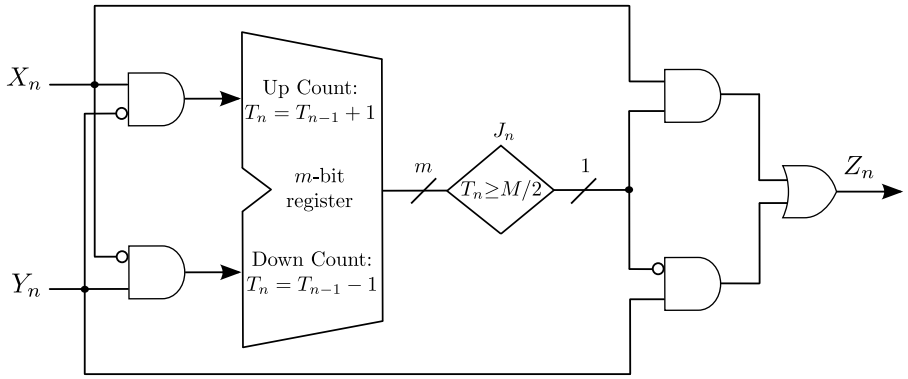

Fig. 2. Proposed Stochastic MAX architecture where $M = 2^m$. T_n is the register's current value, updated according to Eq. (2).

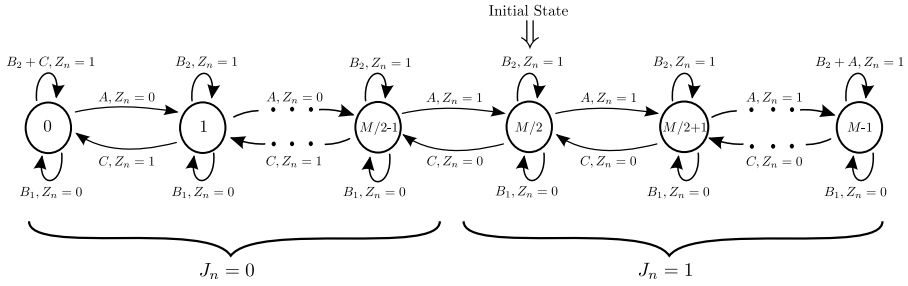

Fig. 3. Markov Chain model of the proposed stochastic MAX architecture. Transition probabilities are given by Eq. (6). J_n denotes the result of the comparison between the register's current value with the initial one $M/2$.

and its current state is S_n, corresponding to the current value T_n of the register. The initial state is $S_0 = M/2$.

The transition from state S_{n-1} to state S_n is determined by S_{n-1}, X_n and Y_n. Using the probability distributions of inputs X_n and Y_n, the assumption that their sequences are IID, and the operation of the MAX architecture in Fig. 2, we derive the transition probabilities shown in the MC model in Fig. 3 as

$$A \triangleq P_r(X_n = 1)P_r(Y_n = 0) = X(1 - Y)$$

$$B_1 \triangleq P_r(X_n = 0)P_r(Y_n = 0) = (1 - X)(1 - Y)$$

$$B_2 \triangleq P_r(X_n = 1)P_r(Y_n = 1) = XY$$

$$B \triangleq B_1 + B_2$$

$$C \triangleq P_r(X_n = 0)P_r(Y_n = 1) = (1 - X)Y, \tag{6}$$

where we have set $X = P_r(X_n = 1)$ and $Y = P_r(Y_n = 1)$.

Assuming the state ordering $(0, 1, \ldots, M-1)$ in $\mathscr{S}$ and using Eq. (6), the $M \times M$ transition probability matrix $H = [P_r(S_{n+1} = s_j | S_n = s_i)]_{s_i, s_j \in \mathscr{S}}$ is written as

$$H = \begin{bmatrix} 1-A & A & 0 & \ldots & \ldots & 0 \\ C & B & A & 0 & \ldots & 0 \\ 0 & C & B & A & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & C & B & A \\ 0 & \ldots & \ldots & 0 & C & 1-C \end{bmatrix}. \tag{7}$$

The probability distribution vector of state S_n, is defined as

$$p_n^T \triangleq \begin{bmatrix} P_r(S_n = 0) \\ P_r(S_n = 1) \\ P_r(S_n = 2) \\ \vdots \\ P_r(S_n = M-1) \end{bmatrix} \in [0, 1]^M, \tag{8}$$

and it is expressed as [33],

$$p_n = p_0 H^n \in [0, 1]^M. \tag{9}$$

Here, p_0 is the initial distribution vector representing the starting state of the register, $S_0 = M/2$. It is

$$p_0 = e_{M/2+1}, \tag{10}$$

where $e_i = [0, \ldots, 0, 1, 0, \ldots, 0] \in \mathbb{R}^M$ is the ith standard vector, i.e., with all zeros except the ith entry being one.

3.2. First order statistics

We use the MC model in Fig. 3 and Eqs. (6)–(10) to derive the output's first order statistics. The expected value of the output Z_n is expressed as

$$\begin{aligned} \mathbb{E}[Z_n] &= P_r(Z_n = 1) \\ &= \sum_{\substack{s \in \mathscr{S} \\ x, y \in \{0,1\}}} P_r(Z_n = 1, S_{n-1} = s, X_n = x, Y_n = y) \\ &= \sum_{\substack{s \in \mathscr{S} \\ x, y \in \{0,1\}}} P_r(Z_n = 1 | S_{n-1} = s, X_n = x, Y_n = y) \\ &\qquad \times P_r(S_{n-1} = s, X_n = x, Y_n = y). \end{aligned} \tag{11}$$

Regarding the conditional probability $P_r(Z_n = 1 | S_{n-1} = s, X_n = x, Y_n = y)$ we note that Z_n is a (deterministic) function of S_{n-1}, X_n and Y_n, as can be seen in the MC model in Fig. 3. Using the MC model and Eq. (4) we can distinguish between three possible cases, i.e.:

(1) When $S_{n-1} \leq M/2 - 2$, then $Z_n = 1$ if and only $Y_n = 1$,
(2) When $S_{n-1} = M/2 - 1$, then $Z_n = 1$ if and only at least one of X_n, Y_n is 1,

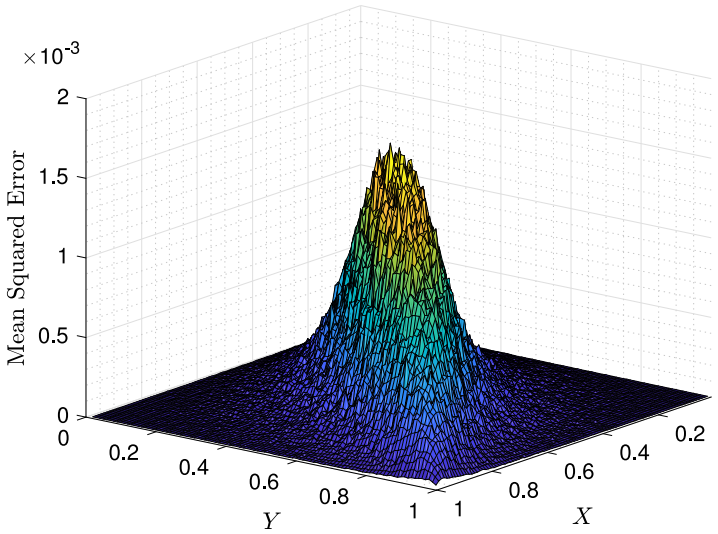

Fig. 4. Numerical simulation of the Mean Squared Error, Eq. (22). The stochastic sequence length is $N = 64$ and the register size is m = 4-bits. 10^3 runs are used for every pair (X, Y).

(3) When $S_{n-1} \geq M/2$, then $Z_n = 1$ if and only $X_n = 1$.

Therefore we can decompose the summation in Eq. (11) as

$$\begin{aligned}\mathbb{E}[Z_n] &= \sum_{s=0}^{M/2-2} P_r(S_{n-1} = s, Y_n = 1) \\ &+ \sum_{(x,y)\neq(0,0)} P_r(S_{n-1} = M/2-1, X_n = x, Y_n = y) \\ &+ \sum_{s=M/2}^{M-1} P_r(S_{n-1} = s, X_n = 1).\end{aligned} \tag{12}$$

Since X_n, Y_n and S_{n-1} are independent random variables, it is

$$P_r(S_{n-1} = s, X_n = x, Y_n = y) = P_r(S_{n-1} = s)P_r(X_n = x)P_r(Y_n = y)$$

simplifying (12) to

$$\begin{aligned}\mathbb{E}[Z_n] &= Y \sum_{s=0}^{M/2-2} P_r(S_{n-1} = s) \\ &+ (X + Y - XY)P_r(S_{n-1} = M/2-1) \\ &+ X \sum_{s=M/2}^{M-1} P_r(S_{n-1} = s) \\ &= p_{n-1}\left(Ye_L^T + (X + Y - XY)e_{M/2}^T + Xe_U^T\right),\end{aligned} \tag{13}$$

where $e_L = \sum_{i=1}^{M/2-1} e_i$ and $e_U = \sum_{i=M/2+1}^{M} e_i$.

Then, the N-bit output sequence time-average,

$$\widetilde{Z}_N = \frac{1}{N}(Z_1 + Z_2 + \cdots + Z_N), \tag{14}$$

has the expected value below based on Eq. (13),

$$\begin{aligned}\mathbb{E}[\widetilde{Z}_N] &= \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}[Z_n] \\ &= \frac{1}{N}p_0\left(\sum_{n=0}^{N-1} H^n\right)\left(Ye_L^T + (X + Y - XY)e_{M/2}^T + Xe_U^T\right).\end{aligned} \tag{15}$$

In the following subsection we use Eq. (15) to confirm the operation of the MAX architecture for large N and M values.

3.3. Proof of the MAX operation at the limit

To verify the operation of the proposed MAX architecture we assume that $0 < X, Y < 1$ and $X \neq Y$. Then, from Eq. (6) it is $0 < A, B, C < 1$, as well as $\rho \neq 1$, where we have defined

$$\rho \triangleq \frac{A}{C} = \frac{X(1-Y)}{Y(1-X)}. \tag{16}$$

Moreover, note that $\rho > 1$ if and only if $X > Y$.

Observing the MC model in Fig. 3 one can conclude that the MC is irreducible, as each state s_j is accessible, with positive probability, from every other state s_i, implying irreducibility for the matrix H as well. Therefore, from Theorem 8.6.1 in [34] we have that

$$\lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1} H^n = \underline{1}^T v, \tag{17}$$

where the row vector $v \in \mathbb{R}^M$ is the unique left eigenvector of H, $vH = v$, corresponding to eigenvalue 1 and being normalized, i.e. $v\underline{1}^T = 1$, and, $\underline{1} = [1, 1, \ldots, 1] \in \mathbb{R}^M$ is the all ones vector. It can be verified directly that $v = \lambda_M[1, \rho, \rho^2, \ldots, \rho^{M-1}]$, where

$$\lambda_M \triangleq \frac{1-\rho}{1-\rho^M}. \tag{18}$$

Combining Eqs. (15) and (17) and noting that $p_0\underline{1}^T = 1$ we get

$$\begin{aligned}\lim_{N\to\infty} \mathbb{E}[\widetilde{Z}_N] &= v\left(Ye_L^T + (X + Y - XY)e_{M/2}^T + Xe_U^T\right) \\ &= Yve_L^T + (X + Y - XY)ve_{M/2}^T + Xve_U^T.\end{aligned} \tag{19}$$

Using the expressions of v, e_L and e_U we get

$$\begin{aligned}ve_L^T &= \lambda_M\left(1 + \rho + \cdots + \rho^{M/2-2}\right) = \frac{1-\rho^{M/2-1}}{1-\rho^M} \\ ve_{M/2}^T &= \lambda_M\rho^{M/2-1} = \frac{\rho^{M/2-1} - \rho^{M/2}}{1-\rho^M} \\ ve_U^T &= \lambda_M\left(\rho^{M/2} + \rho^{M/2+1} + \cdots + \rho^{M-1}\right) = \frac{\rho^{M/2} - \rho^M}{1-\rho^M}\end{aligned} \tag{20}$$

directly implying from (20) that $lim_{M\to\infty}(lim_{N\to\infty}\mathbb{E}[\widetilde{Z}_N]) = Y$ if $\rho < 1$ and $lim_{M\to\infty}(lim_{N\to\infty}\mathbb{E}[\widetilde{Z}_N]) = X$ if $\rho > 1$, and so

$$\lim_{M\to\infty}\left(\lim_{N\to\infty}\mathbb{E}[\widetilde{Z}_N]\right) = \begin{cases} X, & X > Y \\ Y, & Y > X \end{cases}, \tag{21}$$

which proves that the proposed architecture provides the correct expected result in the limiting case.

3.4. Error calculation

To investigate the distribution of the average output's error for different probabilities $X, Y \in [0,1]$ of the inputs, we use the Mean Squared Error (MSE) metric, defined as

$$Z_{error} = \mathbb{E}\left[(\widetilde{Z}_N - max\{X,Y\})^2\right]. \tag{22}$$

The numerical calculation of the MSE is done for different X, Y values in $[0,1]$ while the simulation is performed for 10^3 runs for each pair. The MSE results are shown in Fig. 4, for stochastic sequence length $N = 64$ and for register size of $m = 4$-bits. It is observed that the MSE peaks at the center $X = Y = 0.5$ and gradually decreases when moving away of it.

3.5. The MIN architecture as a variation of the MAX one

The MIN architecture can be obtained as a variation of the MAX one, as shown in Fig. 5. The counting of logic 1s is identical to that of the MAX architecture. The difference between the two architectures is the swap of the NOT gate between the two AND gates that along with the OR gate, determine the output. Therefore, the MIN architecture's analysis is similar to that of the MAX one's and follows that in Sections 3.2–4.

4. Selection of the register's size

From the architecture in Fig. 2 and the register's update equation, Eq. (2), it is seen that an overflow or an underflow of the register may appear when N is relatively large. Consider for example the case when $N \gg M$ and it happens that $\{X_n\}$ has a large segment of 1s and at the same time $\{Y_n\}$ has a large segment of 0s. Therefore, to provide guidelines for the selection of the register's size, it is important to investigate how the number of states, M, is related to the probability of overflow or an underflow and when this leads to erroneous bits in the output sequence $\{Z_n\}$.

4.1. Markov chain overflow/underflow model

To investigate overflows/underflows in the MC in Fig. 3, we consider two possible cases for the transitions of its current state S_n. Assuming that $N \to \infty$ and the initial state is $S_0 = M/2$, if $X > Y$ the MC's state S_n will transition from $M/2$ right-wise up to state $M-1$, whereas, if $X < Y$, then S_n will transition from $M/2$ left-wise up to state 0, both with probability one. Exceeding $M-1$ right-wise or 0 left-wise is not possible and an overflow/underflow is observed which is not captured by the MC model in Fig. 3.

To model the overflow/underflow occurrence we modify the MC model in Fig. 3 to that in Fig. 6 having two additional states M_a, M_b, which are *absorbing*. Therefore, $S_{n-1} = M-1$, $X_n = 1$ and $Y_n = 0$ will result in $S_k = M$ for all $k \geq n$, capturing the state and indicating that an overflow has occurred. Similarly, $S_{n-1} = 0$, $X_n = 0$ and $Y_n = 1$ will result in $S_k = 0$ for all $k \geq n$. Both extra states M_a, M_b do not imply an increase in the register size and are only used for modeling purposes.

To calculate the probability of overflow/underflow, as a function of M and N, we define the new set of states $\widehat{\mathscr{S}} \triangleq \{0,1,\ldots,M-1,M_a,M_b\}$ such that $|\widehat{\mathscr{S}}| = M+2$. If we assume a state ordering $(0,1,\ldots,M-1,M_a,M_b)$, then the $(M+2)\times(M+2)$ transition probability matrix $\widehat{H}$ of the new MC is

$$\widehat{H} = \begin{bmatrix} B & A & 0 & \ldots & \ldots & 0 & C & 0 \\ C & B & A & 0 & \ldots & 0 & 0 & 0 \\ 0 & C & B & A & \ldots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & C & B & A & 0 & 0 \\ 0 & \ldots & \ldots & 0 & C & B & 0 & A \\ 0 & \ldots & \ldots & 0 & 0 & 0 & 1 & 0 \\ 0 & \ldots & \ldots & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{23}$$

The probability distribution vector of the current state $\widehat{S}_n$ of the MC model in Fig. 6 is

$$\widehat{p}_n^T \triangleq \begin{bmatrix} P_r(\widehat{S}_n = 0) \\ P_r(\widehat{S}_n = 1) \\ \vdots \\ P_r(\widehat{S}_n = M-1) \\ P_r(\widehat{S}_n = M_a) \\ P_r(\widehat{S}_n = M_b) \end{bmatrix} \in [0,1]^{M+2} \tag{24}$$

and it can be expressed as

$$\widehat{p}_n = \widehat{p}_0 \widehat{H}^n, \tag{25}$$

where the initial distribution vector $\widehat{p}_0$ is

$$\widehat{p}_0 = e_{M/2+1} \in [0,1]^{M+2} \tag{26}$$

and $e_i = [0,\ldots,0,1,0,\ldots,0] \in \mathbb{R}^{M+2}$ is the corresponding standard vector.

The register can underflow in M_a or overflow in M_b. The probability it has done so at least once, at time index n, is $P_r(\widehat{S}_n = M_a)$ and $P_r(\widehat{S}_n = M_b)$, respectively, calculated as

$$[P_r(\widehat{S}_n = M_a), P_r(\widehat{S}_n = M_b)] = \widehat{p}_0 \widehat{H}^n \left[e_{M+1}^T, e_{M+2}^T\right] \in \mathbb{R}^{1\times 2}. \tag{27}$$

In Fig. 7, we illustrate an example of the probability to underflow in M_a or overflow in M_b calculated using Eq. (27). We have selected inputs as $X = Y = 0.5$ and parameterized with $N = 64$ for number of states $M = 4$,

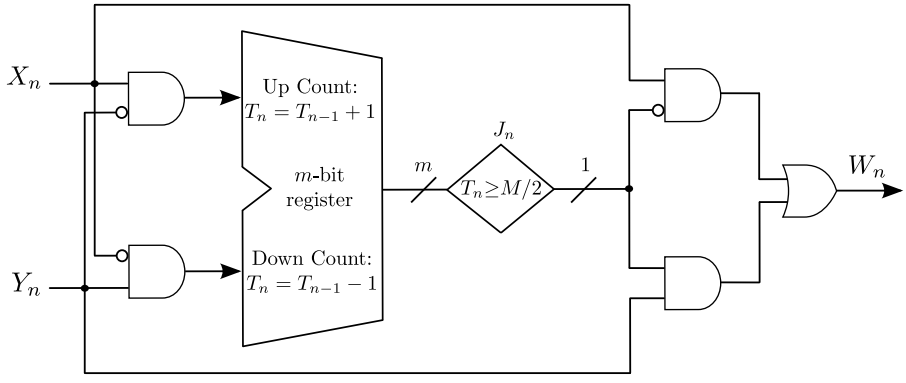

Fig. 5. Proposed Stochastic MIN architecture. T_n denotes the $M = 2^m$ register's current value and is updated according to Eq. (2).

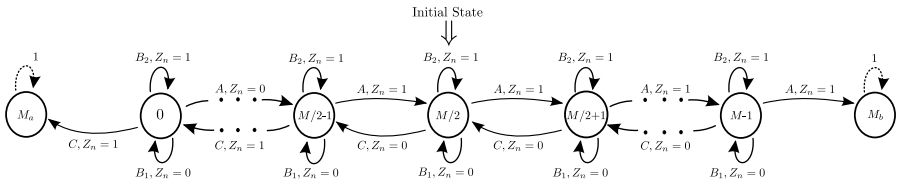

Fig. 6. Markov Chain overflow/underflow model of the proposed stochastic MAX architecture. Transition probabilities are given by (6). Absorbing states M_a and M_b represent underflow and overflow respectively.

8,…,32. As one can observe, the probability to underflow is smaller than the one to overflow since the initial state $M/2$ is closer to M_b.

4.2. Average number of steps to overflow or underflow

It is reasonable to further investigate the probability of overflows/underflows as Eq. (27) provides with an estimate of their occurrence. To this end, one can calculate the *expected* number of transitions for the MC's state to be absorbed in either M_a or M_b. To this end we decompose the transition probability matrix $\widehat{H}$ (canonical form [33,35]) as

$$\hat{H} = \left[\begin{array}{c|c} \tilde{H} & R \\ \hline 0_{2,M} & I_2 \end{array}\right], \tag{28}$$

where $\widetilde{H} \in [0,1]^{M\times M}$, $R \in [0,1]^{M\times 2}$, $I_2 \in [0,1]^{2\times 2}$ and $0_{2,M}$ is the $2\times M$ zero matrix. Using $\widetilde{H}$ from Eq. (28), we define the fundamental matrix of the absorbing MC in Fig. 6 as

$$F = (I_M - \widetilde{H})^{-1} \in \mathbb{R}^{M\times M}, \tag{29}$$

and thus, starting from $S_0 = M/2$ in our case, the expected number of transitions *before* the absorption is calculated as

$$N^* = p_0 F \underline{1}, \tag{30}$$

where $\underline{1}$ is the column vector of M ones and p_0 is given by Eq. (10). N^* is used to provide guidelines for the register's size selection in the following subsection.

4.3. Error due to overflow/underflow and register size selection

Overflows/underflows in the architecture in Fig. 2 do not necessarily imply that $\{Z_n\}$ will contain bit-errors. For instance, in the case of $X = 1$ and $Y = 0$, it is expected that T_n is increased linearly and overflow in a few transitions. Yet, the output's mean value is calculated correctly.

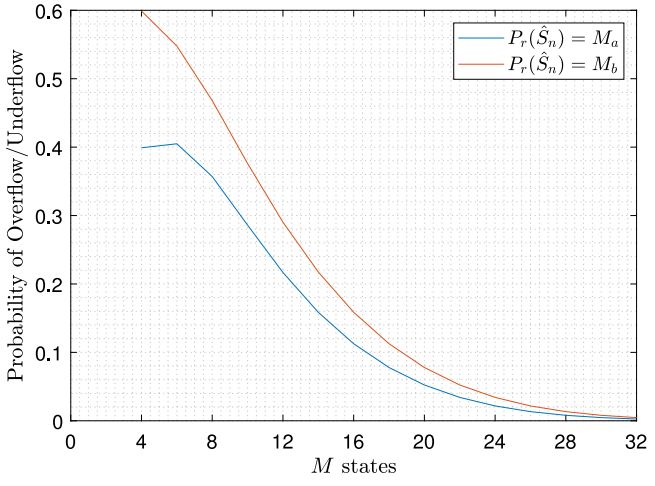

Fig. 7. Probability of overflow/underflow calculated using Eq. (27) for a number of register's states $M = 4,8,\ldots,32$, input sequence length $N = 64$, $X = Y = 0.5$ and starting state $S_0 = M/2$.

To further analyze when errors occur due to overflows/underflows we can consider the following scenario, which is an edge case for the MC of Fig. 3. Starting from $S_0 = M/2$, a monotonic transition up to state $M-1$ happens in $M/2-1$ transitions without overflowing. Now, if A happens, the next state is $M-1$ and the first overflow occurs in a total of $M/2$ transitions (or clock cycles). The overflow appears as an error in the output when the initial state $M/2$ is revisited in a minimum of $M/2-1$ transitions. Therefore M is the *minimum* number of transitions possibly leading to an erroneous bit at the output. OR stating it reversely, as long as $N \leq M-1$ there are no errors due to overflow/underflow. The above are captured in the example of Fig. 8 for $M = 8$ states.

In some cases it is difficult to satisfy $N \leq M-1$ as N may take large values when accurate calculations are required. Allowing some overflows/underflows, possibly implying output errors, may significantly reduce $M = 2^m$. To this end we use N^* in Eq. (30) as a guideline to select M, noting that $N^* = N^*(X,Y,M)$ is a function of X, Y and the numbers of states M. Then, the register's size can be selected as

$$\widehat{m} = min\left\{ m \in \mathbb{N} \,\middle|\, \min_{(x,y)\in[0,1]^2} N^*(x,y,M) \geq \delta N \right\}, \tag{31}$$

where we choose $\delta = 1$, but it can take any positive real. In Table 1 the values of $\widehat{m}$ are presented for typical stochastic sequence lengths N.

5. Comparison of the proposed architectures with existing ones in the Stochastic Computing literature

In this section we compare the proposed stochastic MAX and MIN architectures with existing ones selected from the SC literature, in both computational accuracy and hardware resources. With respect to the computational accuracy we use the MSE as our performance metric. It is calculated numerically in a grid of pairs (X,Y), where for each pair we performed 10^3 runs with IID sequences. Then, we averaged the MSE values of all points in each architecture and the procedure was repeated for typical stochastic sequence lengths $N = 2^k$, for $k = 4,\ldots,10$. We note the following: (1) for all comparisons we have assumed that the stochastic numbers are in unipolar format; (2) for all architectures we selected for each N the register size m that results in the highest computational accuracy possible and we provide it in Table 2; and (3) we assumed that the up & down counting in all MAX/MIN architectures is done using binary (ripple) counters able to count up to $M = 2^m$ states, where m is the register's size. The MSE results are graphically presented in Fig. 9.

The proposed as well as the rest architectures considered, were described in Verilog HDL using Xilinx's Vivado Design suite, so as to verify their proper operation. Then, the designs were synthesized using the Synopsys Design Compiler with the FreePDK CMOS library at 45 nm [36]. In Table 3 we provide the following estimates: (1) the total area in μm^2; (2) the average power consumption for the max operating frequency in mW; (3) the delay in ns; and (4) the energy = average power × delay in pJ. The energy consumption for N cycles is demonstrated in Fig. 10.

The max architectures [21,30–32] including the proposed one can

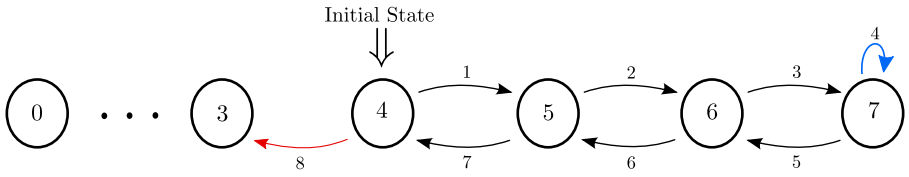

Fig. 8. Example of an error due to overflow for $M = 8$ states. Blue arrow indicates the overflow and red the erroneous bit.

Table 1
Register size $\widehat{m}$-bit satisfying $N^* \geq N$.

Sequence length N-bits	16	32	64	128	256	512	1024
Register size $\widehat{m}$-bits	2	3	3	4	5	5	6

output the MIN without additional hardware resources (only with modification). As such, the presented accuracy results in Fig. 9 and the hardware resources in Table 3 for the MAX architectures apply to the MIN ones as well.

5.1. Comparison with [21]

The inputs $\{X_n\}, \{Y_n\}$ in this architecture, are fed to a MUX, which uses a SNG to generate its select signal. The MUX's output is the input to a stochastic tanh function implemented as a saturating FSM of 2^m states. The architecture's output is determined by a second MUX, which outputs either X_n or Y_n, according to the tanh FSM's current output value.

According to Fig. 9, the proposed MAX results in better computational accuracy and also occupies less resources according to Table 3, which is due to the SNG used in [21]. Considering the register's size, in the proposed MAX architecture it is derived according to the analysis shown in Section 4, whereas in the architecture in [21] numerical simulations are required.

5.2. Comparison with [31]

To improve on the hardware resources from [21] and avoid the SNG, this architecture uses an XOR gate between its two inputs $\{X_n\}, \{Y_n\}$, which operates as an enable signal to count X_n using a tanh-based FSM. Similar to [21], the counting is based on a saturating tanh FSM of 2^m states, while the FSM's output is used as a select signal in a MUX that determines if X_n or Y_n is the current output.

Compared to [31], the proposed MAX results in better computational performance according to Fig. 9. In terms of hardware resources, the proposed MAX occupies slightly more area when the same register size is used, but, has less energy and power consumption; although it is expected that higher area will result in higher power and energy consumption, in fact, the synthesis tool optimizes further the design's mapping using high area, power and mapping effort. Therefore, even if one counts the gates used between the two architectures, the theoretical result will not reflect the one obtained from the synthesis tool. Moreover, the advantage of the proposed MAX architecture over the one in [31], is the design guidelines for the register's size selection according to N, which eliminates the simulation time completely.

5.3. Comparison with [30]

Inspired by [31], the architecture in [30] uses a linear FSM instead of a binary one, i.e. a shift register of m-bits. Preserving the enable operation from the XOR gate, the shift register performs a right shift of the most significant bit (MSB) when $X_n = 1$, and a left shift otherwise. The FSM's output is the register's least significant bit (LSB), which is 1 if it has saturated up to the LSB. The architecture's output is determined by a MUX along with additional logic gates and selects either the FSM's output or Y_n.

From Fig. 9, the proposed architecture results in better accuracy, but, the architecture in [30] is more hardware-efficient according to Table 3, which is due to the shift-register used over the binary one. However, it is expected that if the shift-register's size is not chosen carefully based on the sequence length N, it will directly affect the output's accuracy; when the number of the FSM's states are reduced, it will result in reduced computational accuracy and this is shown in [30]. We note that the shift register size values shown in Table 2 are taken from [30].

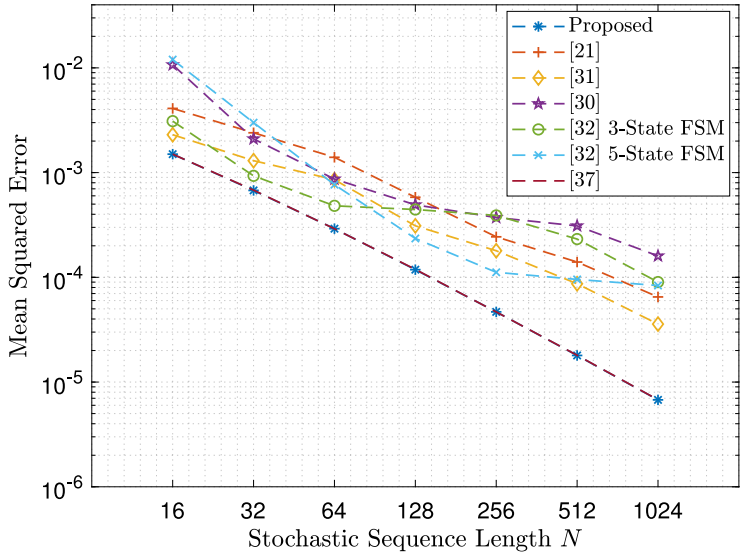

Fig. 9. Accuracy comparison in MSE of stochastic MAX architectures for typical sequence lengths N. For each N, their register sizes are selected to result in the highest MSE and are cited in Table 2.

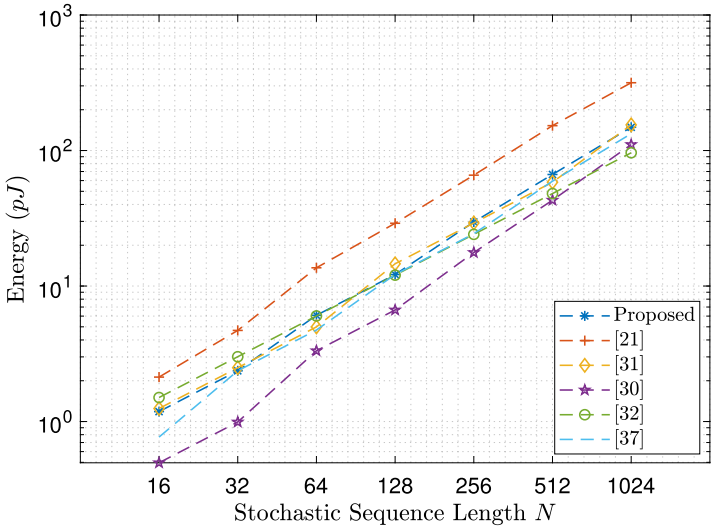

Fig. 10. Energy comparison in pJ of stochastic MAX architectures for typical sequence lengths N.

Table 2
Register sizes resulting in the highest MSE based on N.

$N = 2^k$	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Proposed	2	3		4	5		6
[0]	2		3			4	5
[0]	2			3		4	5
[0]	2			3	4	5	6
[0]	2			3			
[0]	1	2		3		4	5

5.4. Comparison with [32]

The operation of this architecture is based upon the correlation of its two input sequences $\{X_n\}, \{Y_n\}$ using a 3-state FSM that forces the overlap of their logic 1s. The FSM is then followed by an OR gate (or AND) to output the MAX (or the MIN). From Fig. 9, it can be seen that the proposed MAX results in better accuracy, regardless of the stochastic sequence length N, when a 3-state FSM is used. To further investigate the impact of the FSM's number of states in accuracy, we increased their total number from 3 to 5. One can observe that the accuracy is increased for sequences with $N \geq 128$-bit length when compared to the 3-state FSM, but, it is lower than that of the proposed MAX architecture. In terms of hardware resources, the proposed MAX achieves similar performance with register sizes $m = 2,3$-bits, while for more than 4-bits, the MAX in [32] is slightly better. However, one should not neglect the fact that to achieve the same accuracy as the proposed one, the MAX in [32] requires more computational cycles N, which reflects on the total energy consumed.

5.5. Comparison with [37]

In the architecture in [37], the procedure to store the differences between the inputs follows that of the proposed MAX. However, in the proposed work the current value of the register's state is compared with $M/2$ instead of 0 in [37]. As such, the overflow/underflow modeling is improved and hence the understanding of the errors due to overflows. According to Fig. 9 it can be seen that the MSE has the same order of magnitude. However, with respect to the hardware resources, the proposed MAX requires slightly less area when the same register size is used.

Table 3
Comparison of hardware resources in area (μm^2), critical path (ns), power (mW) and energy (pJ) consumption.

	Register m-(bit)	Area (μm^2)	Power (mW)	Critical path (ns)	Energy (pJ)
Proposed	$m = 2$	48.33	0.044	1.5	0.066
	$m = 3$	73.69	0.063		0.094
	$m = 4$	92.31	0.074		0.111
	$m = 5$	106.55	0.081		0.121
	$m = 6$	117.44	0.093		0.139
[21] 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
[31]	$m = 2$	48.01	0.052	1.5	0.078
	$m = 3$	61.47	0.076		0.114
	$m = 4$	104.97	0.101		0.151
[30] 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
[32] 3-State FSM	$m = 2$	91.49	0.062	1.5	0.094
[37]	$m = 1$	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	0.088		0.130

6. Application of the proposed MAX and MIN architectures in image processing

In this section we demonstrate the efficacy of the proposed stochastic MAX and MIN architectures in standard image processing tasks. To proceed with the experiments, we note the following; (1) The accuracy of computations is derived with simulations using MATLAB; and (2) All

Fig. 11. Denoising using a 3×3 median filter. From left to right: (I) MATLAB's 8-bit noisy image with salt & pepper noise density 0.05, (II) MATLAB's median filtered image, (III) Proposed stochastic median filter with sequence length $N = 256$ and register size $m = 3$-bits.

designs were described using Verilog HDL in Xilinx's Vivado Design Suite and then synthesized using the Synopsys Design Compiler with the Free PDK CMOS library at 45 nm [36].

6.1. Noise reduction

The first image processing task we examine is that of the image denoising. To execute this task, here we consider a 3×3 median filter realized using the proposed stochastic MAX and MIN architectures and the filter's structure is based upon the sorting network presented in [21].

We first select a gray-scale image using 8-bit representation and then inject salt & pepper noise, with noise density of 0.05. The pixel values are afterwards normalized from range $[0, 255]$ to range $[0, 1]$ in order to be processed in the SC domain. To investigate the computational accuracy we consider typical stochastic sequence lengths $N = 2^k$, with $k = 5, \ldots, 10$ and calculate the Peak Signal-to-Noise Ratio (PSNR) in dB and the Structural Similarity Index Measure (SSIM). The first metric, the PSNR, measures the absolute accuracy of computations, whereas the second one, the SSIM, measures the perceived quality of an image with values $[0, 1]$ (higher value means better quality).

In Table 4 the calculated PSNR and SSIM results for typical values of N are shown. Moreover, in Fig. 11 for $N = 2^8$-bit sequences and a register size of $m = 3$-bits, the denoising with the 3×3 median filter using the proposed stochastic MAX and MIN is shown, compared to the computation using MATLAB. As one can observe, both the PSNR and the SSIM with values 31.87 and 0.90 respectively, demonstrate the increased computational efficiency of the proposed MAX & MIN architectures.

In Table 5, we present the hardware resources required to realize the 3×3 median filter using the proposed MAX & MIN architectures and the standard binary method, where we cite two different implementations. In the first one, we have not included the hardware resources for the generation of the input sequences as we want for our implementation to be flexible based on the designer's choice of inputs (pseudorandom, random etc.). In the second one, we have included the hardware resources of an optimized SNG based on the sharing scheme in [26]. We note that we relaxed the register size requirements to $m = 3$-bits in the proposed MAX/MIN architectures for this specific task as our experiments showed that the accuracy of computations is not degraded.

From the results shown in Table 5, the advantage of the proposed method is the occupied area, which is reduced by approximately 40%/28% with/without SNGs when compared to the binary one. Moreover, with respect to the energy efficiency, it is expected that the stochastic sequence length N affects it directly; for example for $N = 64$-bit sequences the total energy consumed is 85.76/75.52 pJ with/without SNGs resulting in moderate values compared to the binary method.

6.2. Down sampling

The second image processing task we examine is that of the down sampling. It is a standard process used in NNs as it reduces the dimensionality of the input image based on a max pooling kernel, allowing for the most important features to be preserved. Here, we consider a 2×2 max pooling kernel, realized using the proposed stochastic MAX architecture.

Similar to the denoising task, we first select a grayscale image and normalize its pixel values to range $[0, 1]$. Then we select stochastic sequence lengths $N = 2^k$ with $k = 5, \ldots, 10$ and investigate the kernel's performance considering the PSNR and SSIM metrics.

In Tables 6 and 7 the accuracy on computations and the required hardware resources to realize the 2×2 max pooling kernel are respectively cited. It is shown that using more than $N = 2^7$-bit sequence lengths, the downsampling of an image can be achieved accurately. This is also demonstrated in Fig. 12 for sequence length $N = 2^8$-bits and register size of $m = 4$-bits, where the max pooling is compared to the MATLAB's calculation.

For the reported hardware resources in Table 6, it is shown that the realization of the 2×2 kernel using the proposed stochastic MAX, occupies smaller area when compared to the binary one, approximately 40% less. This can benefit NN-based designs since (1) multiple copies of the kernel are required and (2) they have to operate in parallel.

7. Conclusion

In this work Stochastic Computing MAX & MIN architectures were presented. Their stochastic operation was modeled analytically using Markov Chains, which allowed for an in-depth description of their statistical properties and the verification of their proper operation. The Markov Chain overflow/underflow analysis allowed to model the

Table 4
Computational accuracy in PSNR and SSIM of the realized 3×3 median filter using the proposed MAX and MIN architectures.

$N = 2^k$	2^5	2^6	2^7	2^8	2^9	2^{10}
PSNR (dB)	24.85	27.33	29.72	31.87	33.66	34.91
SSIM	0.59	0.73	0.83	0.90	0.94	0.96

Table 5
Hardware resources for the implementation of a 3×3 median filter using the proposed MAX & MIN architectures in area (μm^2), critical path (ns), power (mW) and energy (pJ).

	Area (μm^2)	Power (mW)	Critical Path (ns)	Energy (pJ)
Proposed	1,539	0.59	2.0	1.18
Proposed w/ SNG	1,812	0.67	2.0	1.34
Binary 8-bit	2,520	2.295	2.2	5.05

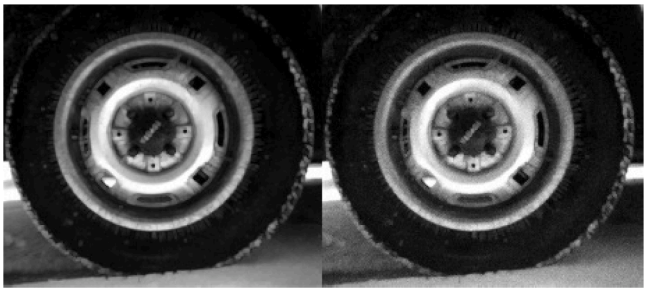

Fig. 12. Down sampling using a 2 × 2 max-pooling kernel. Left: MATLAB's max pooling computation for 8-bit pixel representation, Right: max pooling kernel realized using the proposed stochastic MAX with sequence length $N = 2^8$ and register size $m = 4$-bits.

Table 6
Computational accuracy in PSNR and SSIM of the realized 2 × 2 Max pooling kernel using the proposed MAX architecture.

$N = 2^k$	2^5	2^6	2^7	2^8	2^9	2^{10}
PSNR (dB)	20.09	23.14	26.31	29.58	32.94	36.52
SSIM	0.58	0.72	0.82	0.90	0.94	0.97

Table 7
Hardware resources for the implementation of a 2 × 2 Max pooling kernel using the proposed MAX architecture in area (μm^2), critical path (ns), power (mW) and energy (pJ).

	Area (μm^2)	Power (mW)	Critical Path (ns)	Energy (pJ)
Proposed	250.61	0.084	2.0	0.17
Binary 8-bit	432.71	0.058	2.2	0.12

probability of overflows/underflows and potential errors caused by them, providing guidelines for the register's size as well. Comparisons of the proposed architectures with the Stochastic Computing literature demonstrated the improvement on the latency-accuracy trade-off. Finally, the two image processing tasks demonstrated their efficiency in area occupation and computational accuracy.

CRediT authorship contribution statement

Paul P. Sotiriadis: Conceptualization, Methodology, Validation, Formal Analysis, Supervision, Writing – review & editing. **Nikos Temenos:** Methodology, Software, Validation, Formal Analysis, Visualization, Investigation, Writing – original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

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

References

[1] T. Araujo, M.B. Cardoso, E.G. Nepomuceno, C.H. Llanos, J. Arias-Garcia, A new floating-point adder FPGA-based implementation using RN-coding of numbers, Comput. Electr. Eng. 90 (2021), 106947.
[2] Y. Mounica, K.N. Kumar, S. Veeramachaneni, N. Mahammad, Energy efficient signed and unsigned radix 16 booth multiplier design, Comput. Electr. Eng. 90 (2021), 106892.
[3] Y. Liu, S. Liu, Y. Wang, F. Lombardi, J. Han, A survey of stochastic computing neural networks for machine learning applications, IEEE Trans. Neural Netw. Learn. Syst. 32 (7) (2021) 2809–2824.
[4] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, W.J. Gross, VLSI implementation of deep neural network using integral stochastic computing, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 25 (10) (2017) 2688–2699.
[5] P. Zakian, R.N. Asli, An efficient design of low-power and high-speed approximate compressor in FinFET technology, Comput. Electr. Eng. 86 (2020), 106651.
[6] A. Alaghi, W. Qian, J.P. Hayes, The promise and challenge of stochastic computing, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (8) (2018) 1515–1531.
[7] N. Temenos, P.P. Sotiriadis, Nonscaling adders and subtracters for stochastic computing using Markov chains, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 29 (9) (2021) 1612–1623.
[8] Y. Liu, L. Liu, F. Lombardi, J. Han, An energy-efficient and noise-tolerant recurrent neural network using stochastic computing, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 27 (9) (2019) 2213–2221.
[9] W.J. Gross, V.C. Gaudet, Stochastic Computing: Techniques and Applications, Springer, International Publishing, Springer Nature Switzerland AG, 2019.
[10] B.R. Gaines, Stochastic Computing Systems, Springer, Boston, MA, 1967.
[11] W. Qian, M.D. Riedel, H. Zhou, J. Bruck, Transforming probabilities with combinational logic, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 30 (9) (2011) 1279–1292.
[12] M.H. Najafi, D. Jenson, D.J. Lilja, M.D. Riedel, Performing stochastic computation deterministically, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 27 (12) (2019) 2925–2938.
[13] Z. Lin, G. Xie, W. Xu, J. Han, Y. Zhang, Accelerating Stochastic Computing Using Deterministic Halton Sequences, IEEE Trans. Circuits Syst. II 68 (10) 3351–3355.
[14] M. Yang, B. Li, D.J. Lilja, B. Yuan, W. Qian, Towards Theoretical Cost Limit of Stochastic Number Generators for Stochastic Computing, in: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Hong Kong, China, 2018.
[15] A. Morro, V. Canals, A. Oliver, M.L. Alomar, F. Galán-Prado, P.J. Ballester, J. L. Rosselló, A stochastic spiking neural network for virtual screening, IEEE Trans. Neural Netw. Learn. Syst. 29 (4) (2018) 1371–1375.
[16] B.D. Brown, H.C. Card, Stochastic neural computation I: Computational elements, IEEE Trans. Comput. 50 (9) (2002) 891–905.
[17] B.D. Brown, H.C. Card, Stochastic neural computation II: Soft competitive learning, IEEE Trans. Comput. 50 (9) (2002) 906–920.
[18] S. Liu, H. Jiang, L. Liu, J. Han, Gradient descent using stochastic circuits for efficient training of learning machines, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (11) (2018) 2530–2541.
[19] V.T. Lee, A. Alaghi, J.P. Hayes, V. Sathe, 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, 2017.
[20] P. Li, D.J. Lilja, Using Stochastic Computing to Implement Digital Image Processing Algorithms, in: IEEE 29th International Conference on Computer Design (ICCD), Amherst, MA, USA, 2011.
[21] P. Li, D.J. Lilja, W. Qian, K. Bazargan, M.D. Riedel, Computation on stochastic bit streams digital image processing case studies, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 2 (3) (2014) 449–462.
[22] A. Alaghi, C. Li, J.P. Hayes, Stochastic circuits for real-time image-processing applications, in: IEEE 50th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 2013.
[23] Y. Liu, K.K. Parhi, Architectures for recursive digital filters using stochastic computing, IEEE Trans. Signal Process. 64 (14) (2016) 3705–3718.
[24] N. Saraf, K. Bazargan, D.J. Lilja, M.D. Riedel, IIR filters using stochastic arithmetic, in: IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2014.
[25] A. Vlachos, N. Temenos, P.P. Sotiriadis, Exploring the Effectiveness of Sigma-Delta Modulators in Stochastic Computing-Based FIR Filtering, in: IEEE 10th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 2021.
[26] H. Ichihara, T. Sugino, S. Ishii, T. Iwagaki, T. Inoue, Compact and accurate digital filters based on stochastic computing, IEEE Trans. Emerg. Top. Comput. 7 (1) (2019) 31–43.
[27] A. Alaghi, J.P. Hayes, STRAUSS: Spectral transform use in stochastic circuit synthesis, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 35 (11) (2015).
[28] S. Liu, W.J. Gross, J. Han, Introduction to dynamic stochastic computing, IEEE Circuits Syst. Mag. 20 (3) (2020) 19–33.

[29] S.S. Tehrani, W.J. Gross, S. Mannor, Stochastic decoding of LDPC codes, IEEE Commun. Lett. 10 (10) (2006) 716–718.
[30] M. Lunglmayr, D. Wiesinger, W. Haselmayr, Design and analysis of efficient maximum/minimum circuits for stochastic computing, IEEE Trans. Comput. 69 (3) (2020).
[31] J. Yu, K. Kim, J. Lee, K. Choi, Accurate and Efficient Stochastic Computing Hardware for Convolutional Neural Networks, in: IEEE 35th International Conference on Computer Design, Boston, MA, USA, 2017.
[32] V.T. Lee, A. Alaghi, L. Ceze, Correlation manipulating circuits for stochastic computing, in: IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2018.
[33] C.M. Grinstead, J.L. Snell, Introduction to Probability, second ed., American Mathematical Society, 1997.
[34] R.A. Horn, C.R. Johnson, Matrix Analysis, first ed., Cambridge University Press, 1990.
[35] C.D. Meyer, Matrix Analysis and Applied Linear Algebra, first ed., Society for Industrial and Applied Mathematics, 2000.
[36] J.E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W.R. Davis, P.D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, R. Jenkal, FreePDK: An Open-Source Variation-Aware Design Kit, in: IEEE International Conference on Microelectronic Systems Education, San Diego, CA, USA, 2007.
[37] N. Temenos, P.P. Sotiriadis, Stochastic computing max & min architectures using Markov chains: Design, analysis, and implementation, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 29 (11) (2021) 1813–1823.