This article has been accepted for publication in IEEE Transactions on Circuits and Systems--II: Express Briefs. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2022.3161991

Modeling a Stochastic Computing Non-Scaling Adder and its Application in Image Sharpening

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

***Abstract*—Non-scaling Stochastic Computing adder and subtracter architectures are introduced. They are modeled using Markov Chains to obtain important statistical properties enabling their design optimization. To demonstrate their efficacy, they are used to realize a stochastic computing-based image sharpening filter which is simulated in MATLAB and Synopsys. The filter's computational efficiency is showcased with standard image processing metrics while its hardware resources are compared to those of the standard binary filter, highlighting the advantages of the proposed approach.**

***Index Terms*—Stochastic Computing, Stochastic Non-Scaling Adder, Stochastic Image Sharpening Filter**

I. Introduction

Stochastic Computing (SC) is an unconventional computing technique where binary-value numbers are encoded in the form of finite-length stochastic sequences of 0s and 1s, $\{X_n\}_{n=1}^N$ [1]–[3]. The encoding is typically done using a Stochastic Number Generator (SNG); on each clock cycle, the k-bit binary number is compared to the output of a k-bit random number source for a total of $N = 2^k$ clock cycles. The N-bit output sequence $\{X_n\}_{n=1}^N$ is assumed to be formed of independent and identically distributed (i.i.d.) random variables and the expected value of the time-average $\widetilde{X}_N = (\sum_{n=1}^N X_n)/N$ to be equal to the original binary-value number. It belongs to $[0, 1]$ (unipolar format) and can be extended to $[-1, 1]$ (bipolar format) using the mapping $X \mapsto 2X - 1$. For convenience we also define $X = P_r(X_n = 1)$.

SC's bit-serial processing allows for the realization of the fundamental arithmetic operations, as well as complex functions using a few standard and minimal logic cells. Moreover, its probabilistic nature makes it resilient to soft-errors. These properties are attractive for applications with massive parallelism needs which are (small)-error tolerant such as Neural Networks [4]–[7] and Image Processing [8], [9].

A standard operation required in the SC-based DSP cores is that of the multiply-and-add and although SC multiplication is a trivial process using a single logic gate, the SC addition can be challenging [10]. The standard way to realize an adder in SC is with a MUX along with a random number source which operates as its select signal. This scales the result of the stochastic addition to the SC's number representation range, according to the MUX's number of inputs. In this direction, scaling adders have been proposed to avoid the additional hardware taxing due to the random number source and to increase the accuracy of the computations [11], [12].

When multiple cascaded computations are required, scaled adders do not favor them, especially when other operations follow. Non-scaling adders have also been explored [13], [14], but, existing approaches operate with different SC number representation formats than the standard unipolar and bipolar ones, thereby imposing design constraints to the other connected computational blocks.

Motivated by the former design challenges, in this work we present a SC non-scaling adder architecture. Our approach deviates from existing ones in the sense that it does not scale the result of addition, operates with random input sequences and is compatible with the standard SC number representation formats. As such, cascaded computations can be realized efficiently, with flexibility in the SC design space.

In the following section, the proposed non-scaling adder architecture is presented, along with its modeling using Markov Chains and the derivation of the subtracter's architecture based on the adder. In Section III, important design guidelines to select the register's size are shown. In Section IV, the performance of the proposed non-scaling adder & subtracter in computational accuracy and hardware resources is shown and is compared to several approaches from the SC literature. Section V demonstrates the proposed adder's & subtracter's cascaded computations with the realization of an image sharpening filter. Finally, Section VI concludes our work.

II. Non-Scaling Stochastic Adder

A. High-Level Architecture

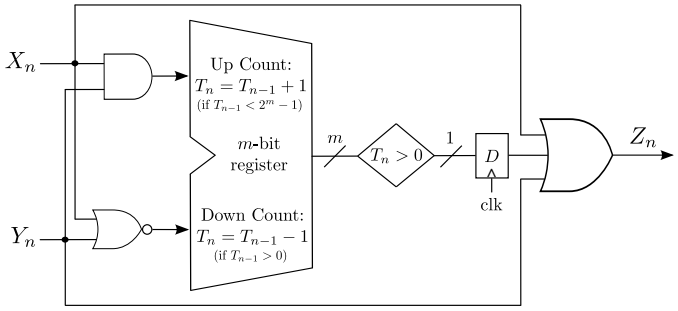

Fig. 1. Proposed stochastic non-scaling adder architecture. T_n is the m-bit register's state, updated according to (1).

Fig. 1 shows the proposed non-scaling adder architecture, where its input sequences $\{X_n\}, \{Y_n\}$ are assumed to be i.i.d.

and $\{Z_n\}$ is its output; $n = 1, 2, \ldots, N$ is the time index. Its operation is based on the saturating up & down counting of a m-bit register within $\{0, 1, 2, ..., M-1\}$ states, where $M = 2^m$. The saturating behavior implies that states 0 and $M-1$ cannot be exceeded. The adder implements the following iteration: A) if $X_n = Y_n = 1$, then $T_n = T_{n-1} + 1$ if $T_{n-1} < M-1$ and $T_n = T_{n-1}$ otherwise, B) if $X_n = Y_n = 0$, then $T_n = T_{n-1} - 1$ if $T_{n-1} > 0$ and $T_n = T_{n-1}$ otherwise, and C) $T_n = T_{n-1}$ if $X_n \neq Y_n$. The iteration is expressed as

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

considering $X_n, Y_n \in \{0, 1\}$ as real numbers. The output is given by

$$Z_n = \text{OR}(X_n, Y_n, T_{n-1} > 0). \quad (2)$$

Finally the register's initial state is assumed to be $T_0 = 0$.

B. Markov Chain Modeling and First-Order Statistics

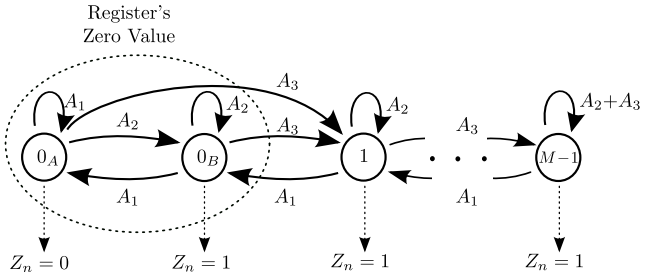

Fig. 2. Markov Chain model of the proposed non-scaling adder, in the form of a stochastic Moore FSM. Transition probabilities A_1, A_2 and A_3 are given by (3).

The operation of the proposed stochastic adder is described by the Markov Chain (MC) model of Fig. 2. The register's zero value, is represented by two different states in the model, namely 0_A and 0_B, whereas states $1, 2, \ldots, M-1$ represent the corresponding values of the register. This allows for its behavior to be expressed as a stochastic Moore FSM, relating the output Z_n to the current state only.

The MC's current state S_n can take values within the set $\mathcal{S} \triangleq \{0_A, 0_B, 1, 2, ..., M-1\}$ of size $M+1$. Assuming that the MC's state is S_{n-1} at time index $n-1$, then the transition to the next state S_n is governed by the transition probabilities

$$\begin{aligned} A_1 &= P_r(X_n = 0) P_r(Y_n = 0) \\ A_2 &= P_r(X_n = 1) + P_r(Y_n = 1) - 2P_r(X_n = 1) P_r(Y_n = 1) \\ A_3 &= P_r(X_n = 1) P_r(Y_n = 1). \end{aligned} \quad (3)$$

To analyze the adder's stochastic behavior we consider the $(M+1) \times (M+1)$ transition probability matrix of its MC model with state ordering $(0_A, 0_B, 1, 2, ..., M-1)$. It is defined as $W = \Big[P_r(S_{n+1} = s_b | S_n = s_a)\Big]$ and expressed as

$$W = \begin{bmatrix} A_1 & A_2 & A_3 & \ldots & \ldots & 0 \\ A_1 & A_2 & A_3 & \ldots & \ldots & 0 \\ 0 & A_1 & A_2 & A_3 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & A_1 & A_2 & A_3 \\ 0 & \ldots & \ldots & 0 & A_1 & A_2 + A_3 \end{bmatrix}. \quad (4)$$

The state's S_n probability distribution vector, is calculated as

$$\pi_n = \pi_0 W^n \in [0, 1]^{M+1}, \quad (5)$$

where $\pi_0 = \big[1, 0, 0, ...0\big] \in [0, 1]^{M+1}$, is the initial distribution vector, representing the register's starting state $S_0 = 0_A$.

Proceeding to the first-order statistics, we leverage the fact that Z_n depends only on the state S_n, which is zero if and only if $S_n = 0_A$. Therefore Z_n's expected value is

$$\mathbb{E}[Z_n] = P_r(Z_n = 1) = 1 - P_r(Z_n = 0) = 1 - \pi_n e_1^T, \quad (6)$$

where we used (5) and $e_i = \big[0, ...0, 1, 0, .., 0\big] \in \mathbb{R}^{M+1}$ is the i-th standard vector. The average value of the output N-bit sequence $\widetilde{Z}_N = \Big(\sum_{n=1}^{N} Z_n\Big)/N$, has expected value

$$\mathbb{E}[\widetilde{Z}_N] = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[Z_n] = 1 - \frac{1}{N} \pi_0 \left(\sum_{n=1}^{N} W^n \right) e_1^T. \quad (7)$$

C. Non-scaling Stochastic Subtracter

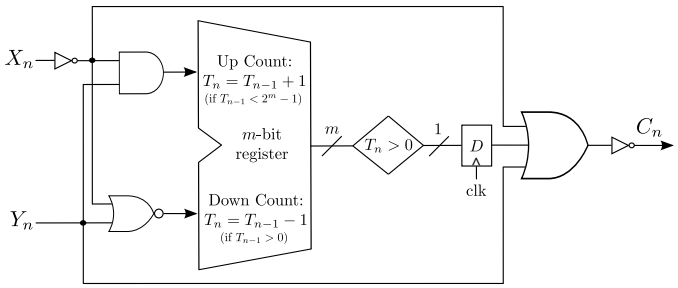

Fig. 3. Proposed stochastic non-scaling subtracter architecture. It consists of the proposed adder in Fig. 1 with two NOT gates.

A stochastic subtracter can also be obtained from the proposed adder by only adding two NOT gates inverting one of its inputs and its output. It's architecture is shown in Fig. 3. Since the adder's operation implies $\widetilde{Z}_N \approx 1 - X + Y$, then we have $\widetilde{C}_N = 1 - \widetilde{Z}_N \approx X - Y$. Note that the proposed subtracter operates in the range $[0, 1]$ implying that $X \geq Y$. The MC model of the subtracter and the corresponding analysis follow closely those of the adder's.

III. Register's Size Design Guidelines

The proposed adder's counting process is constrained by the m-bit register's finite size. A small m may result in significant percentage of ignored ones in the input sequences, due to overflow, impacting the adder's output accuracy. On the other

This article has been accepted for publication in IEEE Transactions on Circuits and Systems--II: Express Briefs. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2022.3161991

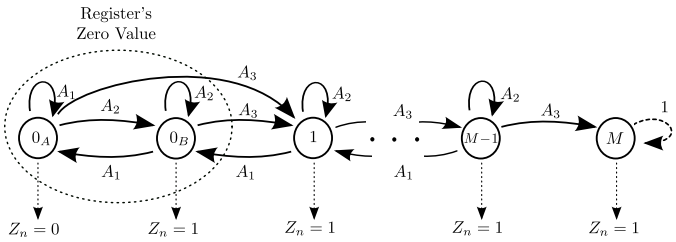

Fig. 4. Markov Chain overflow model of the proposed non-scaling adder with absorbing state M. Transition probabilities A_1, A_2 and A_3 are given by (3).

hand, an unnecessarily large m will tax power and area. It is therefore important to derive guidelines for the register's size selection in order to achieve overall design optimization.

Consider the MC model of Fig. 2 and suppose that its current state is $S_{n-1} = M-1$. If $X_n = Y_n = 1$, then the next state is also $S_n = M-1$. A logic one is outputted, but, the second 1 is ignored since $M-1$ is the last available state of the counter; implying an overflow.

The MC model in Fig. 2 does not model overflow. To do so the extended MC model in Fig. 4 can be used. Its sole difference is an extra *absorbing* state M, indicating that an overflow has occurred.

Assuming the state ordering $(0_A, 0_B, , 1 \ldots, M-1, M)$ containing the extra state M, the transition probability matrix $\hat{W} \in [0,1]^{(M+2)\times(M+2)}$ is

$$\hat{W} = \begin{bmatrix} A_1 & A_2 & A_3 & \ldots & \ldots & \ldots & 0 \\ A_1 & A_2 & A_3 & \ldots & \ldots & \ldots & 0 \\ 0 & A_1 & A_2 & A_3 & \ldots & \ldots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ldots & 0 \\ \vdots & \ldots & 0 & A_1 & A_2 & A_3 & 0 \\ \vdots & \ldots & \ldots & 0 & A_1 & A_2 & A_3 \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 1 \end{bmatrix}. \quad (8)$$

Decomposing the matrix $\hat{W}$ into the form

$$\hat{W} = \left[\begin{array}{c|c} \tilde{W} & R \\ \hline 0 & 1 \end{array}\right], \quad (9)$$

then the fundamental matrix F of an absorbing MC, [15], is

$$F = (I - \tilde{W})^{-1} \in [0,1]^{(M+1)\times(M+1)}. \quad (10)$$

Using matrix F one can calculate the expected number of transitions *before* reaching the (only) absorbing state, M, as

$$\hat{N} = \pi_0 F \underline{1}, \quad (11)$$

where π_0 is the initial distribution vector and $\underline{1}$ is the column vector of $M+1$ ones.

The expected number of transitions $\hat{N}$ before the first overflow in used as a guideline to select the total number of states M and hence the register's size m. Note that $\hat{N}$ is a function of X, Y and $M = 2^m$, i.e., defining the set $\mathcal{XY} = \{(x,y) \in [0,1]^2 \mid x+y \leq 1\}$, it is $\hat{N} : \mathcal{XY} \times \mathbb{N} \to \mathbb{N}$, where $\mathbb{N} = \{1, 2, 3, ...\}$. The register's size should be selected such that $\hat{N}$ is always larger or equal to a multiple of the input sequences' length N, i.e.

$$\hat{m} = \min\left\{m \in \mathbb{N} \mid \min_{(x,y)\in\mathcal{XY}} \hat{N}(x,y,2^m) \geq \rho N\right\}, \quad (12)$$

where here we choose $\rho = 1$. We know that $\hat{m}$ always exists since $\hat{N} \to \infty$ as $m \to \infty$. The register's size $\hat{m}$ for different values of N are cited in Table I.

TABLE I
REGISTER SIZE $\hat{m}$-BIT SATISFYING $\hat{N} \geq N$

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

IV. NON-SCALING ADDER'S & SUBTRACTER'S PERFORMANCE

A. Computational Accuracy Comparison

The computational accuracy of the proposed adder is measured using the Mean Absolute Error (MAE). It is defined as $z_{error} = \mathbb{E}|(X+Y) - \tilde{Z}_N|$, where $(X,Y) \in \mathcal{XY}$ and similarly for the subtracter considering that $X - Y \geq 0$. The MAE is estimated numerically using MATLAB: First, a finite subset of $\mathcal{XY}$ is created by randomly selecting pairs (X,Y) in $\mathcal{XY}$, according to the uniform distribution; then for every such pair the simulation is run 1000 times, each with a new pair of input i.i.d. sequences $\{X_n\}, \{Y_n\}$. The MAE values are calculated and averaged. The experiment is repeated for stochastic sequence lengths of $N = 2^k$, where $k = 4, 5, \ldots, 10$. Note that for the MAE computation, we have considered Sobol low-discrepancy (LD) sequences [16].

The comparison of the proposed adder's and subtracter's computational efficiency to those of existing architectures in the SC literature [11]–[14], [17] is illustrated in Fig. 5. According to the results, the proposed adder and the non-scaling adder in [13] result in the highest performance for sequences with length up to $N = 64$-bit, but, the proposed one is slightly better beyond $N = 64$. Moreover, the proposed non-scaling adder operates with the standard SC representation format in contrast to one in [13] which assumes a two-line encoding (i.e. two sequences representing the magnitude and the sign of the stochastic number). Compared to the other architectures in [11]–[14] and the MUX approach, the proposed adder is more suitable for cascaded computations due to its non-scaling nature, besides its accuracy. With respect to the subtracters, the proposed one achieves the highest computational accuracy while allowing for cascaded computations.

B. Hardware Resources Comparison

All compared adders and subtracters were described using Verilog HDL and then synthesized using the Synopsys Design Compiler with the Free PDK CMOS library at 45 nm [18]. The estimated area, average power consumption for the maximum operating frequency, delay and energy consumption (average power $\times$ delay product) are cited in Table II.

This article has been accepted for publication in IEEE Transactions on Circuits and Systems--II: Express Briefs. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2022.3161991

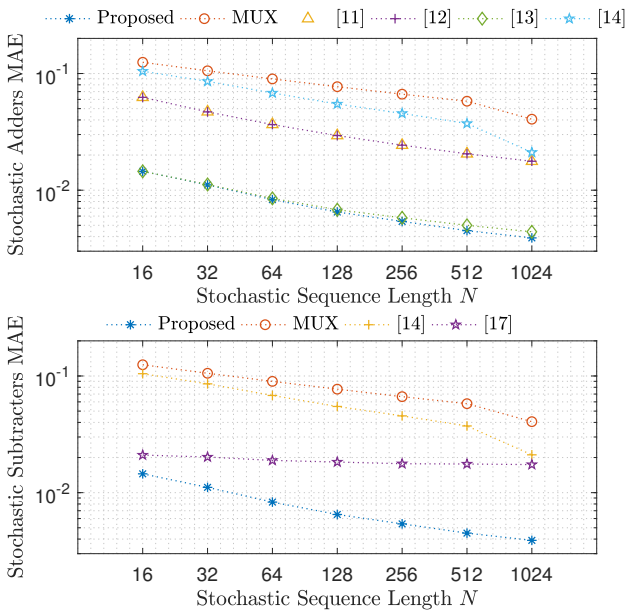

Fig. 5. Mean Absolute Error comparison for stochastic adders and subtracters with different sequence lengths N. Sobol sequences are used.

TABLE II
HARDWARE COMPARISON BETWEEN THE PROPOSED ARCHITECTURES AND STATE-OF-THE-ART

	Stochastic Adders & Subtracters				
	Register (bit)	Area (μm^2)	Power (mW)	Delay (ns)	Energy (pJ)
Proposed* Adder/Subtracter	$m=2$	59.60	0.053	1.4	0.074
	$m=3$	83.49	0.077		0.108
	$m=4$	98.30	0.084		0.117
	$m=5$	112.61	0.098		0.137
[11]		22.41	0.021	0.8	0.016
[12]		54.39	0.040	1.2	0.048
[13]		92.49	0.071	1.2	0.057
MUX* Adder/Subtracter Sobol seq. generator size k	$k=4$	109.34	0.241	0.8	0.192
	$k=5$	117.32	0.272		0.216
	$k=6$	125.03	0.325		0.261
	$k=7$	140.19	0.362		0.289
	$k=8$	164.83	0.378		0.302
	$k=9$	188.54	0.440		0.352
	$k=10$	201.61	0.461		0.368
[14]* Adder/Subtracter Sobol seq. generator size k	$k=4$	113.48	0.251	0.8	0.200
	$k=5$	119.69	0.282		0.225
	$k=6$	136.24	0.330		0.264
	$k=7$	144.54	0.372		0.297
	$k=8$	164.61	0.380		0.304
	$k=9$	188.64	0.445		0.356
	$k=10$	204.75	0.466		0.372
[17]		41.76	0.063	0.8	0.050

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

According to Table II, the proposed adder requires less resources compared to the MUX one and the approach in [14] which requires a random source for the select signal. Compared to [13] the proposed architecture requires slightly more resources when register sizes of $m = 5$-bit are used. The adders in [11], [12] are intended for area & power constrained designs, however, this does not apply to the energy, since they require larger sequence lengths to achieve the same computational accuracy with that of the proposed adder.

Regarding the subtracters, the proposed one requires fewer resources when compared to the MUX and [14]. Although the subtracter in [17] is better hardware-wise, its accuracy improvement is small when compared to the proposed subtracter and is increased when additional hardware is used [17].

V. STOCHASTIC IMAGE SHARPENING FILTER

We demonstrate the efficiency of the proposed architectures in cascaded computations with the realization of an image sharpening filter [19]. Its operation, is described as

$$g(k,l) = f(k,l) + c\big(f(k,l) - w * f(k,l)\big), \qquad (13)$$

where $f(k,l)$ and $g(k,l)$ are the input and output images of size $k \times l$ respectively, w is a weight mask and c is a constant.

To further explain the image enhancement properties of each operation in (13), we start first with its second term. Convolving the input image $f(k,l)$ with a weight kernel w, outputs a filtered version of $f(k,l)$, determined by the kernel's weight values. Then, subtracting $w*f(k,l)$ from $f(k,l)$ allows to extract the "details" of an image. The multiplication with the constant value c, results in image sharpening for $c = 1$ and high-boost filtering for values $c > 1$. Here, we consider $c = 1$. Finally, the sharpened image $g(k,l)$ is obtained by adding the extracted details to the input image. Note that in the convolution process, AND gates are used for multiplication.

Proceeding to the experimental setup, we selected a gray-scale image assuming an 8-bit number representation for each pixel and then we normalized their values to range $[0, 1]$. The normalization is based on stochastic numbers with sequence length $N = 2^k$, with $k = 6, \ldots, 10$. This also applies to the weight values of the 3×3 mask w, which are selected here to be 0.125. Note that all computations are conducted with simulations using MATLAB and Sobol sequences.

The computational accuracy of the proposed stochastic sharpening filter is evaluated with two metrics, the Peak Signal-to-Noise Ratio (PSNR) in dB and the Structural Similarity Index Measure (SSIM). In Table III, the results for typical values of N considered are shown, while in Fig. 6 a graphical illustration of the computations using the proposed architectures with stochastic sequence length $N = 256$ and a register of $m = 4$-bits is demonstrated.

TABLE III
COMPUTATIONAL ACCURACY & IMAGE QUALITY FOR THE IMAGE SHARPENING FILTER REALIZED USING THE PROPOSED ARCHITECTURES

$N = 2^k$	2^6	2^7	2^8	2^9	2^{10}
PSNR (dB)	31.98	33.23	34.63	34.93	34.98
SSIM	0.950	0.960	0.966	0.967	0.967

From a hardware perspective, the realization of the image sharpening filter requires the following computations: 1) a 3×3 convolution kernel, 2) a subtraction and 3) an addition. Among them, the convolution kernel is the largest computational block and its implementation using the proposed stochastic adder requires 8 adders and 9 AND gates for multiplication. For the 9 adders in total and the subtracter, a register size of $m = 4$-bits is used, as it does not degrade the accuracy of computations

This article has been accepted for publication in IEEE Transactions on Circuits and Systems--II: Express Briefs. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2022.3161991

Fig. 6. Image Sharpening Filter. From left to right: a) MATLAB's Original Image, b) MATLAB's Image Sharpening calculation, c) Image Sharpening Filter realized with the proposed SC architectures. Sequence length $N = 256$ and register size $m = 4$-bit.

for stochastic sequences with length N up to 1024.

To compare the hardware resources with the standard 8-bit binary approach, we synthesized both designs using Verilog HDL and Synopsys Design Compiler. According to the results shown in Table IV, the advantage of the proposed approach is that of the area occupation, which is approximately 11% of the standard binary one's. On the other hand, the total energy consumed by the proposed approach is determined by N, which is selected according to the accuracy requirements. For instance, for $N = 2^7$ the total energy dissipated has moderate values equal to $142pJ$. Note that since N is determined by the sobol input sequence generators, they are not included in Table IV, but, they can be designed efficiently according to [16].

TABLE IV
COMPARISON OF HARDWARE RESOURCES FOR THE IMPLEMENTATION OF THE IMAGE SHARPENING FILTER

	Area (μm^2)	Power (mW)	Critical Path (ns)	Energy (pJ)
Proposed	1,093	0.62	1.8	1.11
Binary 8-bit	9,284	2.8	7.6	21.28

VI. CONCLUSION

A non-scaling stochastic computing adder was presented. Its stochastic operation was modeled using Markov Chains, which allowed us to calculate its first order statistics and to provide guidelines for its register size. A non-scaling subtracter architecture was derived by adding two not gates to the proposed adder. Finally, the architectures' effectiveness in cascaded computations, were demonstrated with the implementation of a stochastic image sharpening filter, highlighting its compactness compared to the standard binary implementation.

ACKNOWLEDGEMENTS

The research work was supported by the Hellenic Foundation for Research and Innovation (HFRI) under the HFRI PhD Fellowship grant (Fellowship Number:1216). The authors acknowledge financial support from Weasic Microelectronics S.A. for covering conference attendance expenses.

REFERENCES

[1] B. R. Gaines, *Stochastic Computing Systems*. Springer, Boston, MA, 1967.
[2] A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1515 – 1531, Aug. 2018.
[3] W. J. Gross and V. C. Gaudet, *Stochastic Computing: Techniques and Applications*. Springer, International Publishing, 2019.
[4] A. Morro, V. Canals, A. Oliver, M. L. Alomar, F. Galán-Prado, P. J. Ballester, and J. L. Rosselló, "A stochastic spiking neural network for virtual screening," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 1371 – 1375, Apr. 2018.
[5] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "Vlsi implementation of deep neural network using integral stochastic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2688 – 2699, Oct. 2017.
[6] Y. Liu, L. Liu, F. Lombardi, and J. Han, "An energy-efficient and noise-tolerant recurrent neural network using stochastic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 9, pp. 2213 – 2221, Jun. 2019.
[7] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2809 – 2824, Jul. 2021.
[8] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 2, no. 3, pp. 449–462, Apr. 2014.
[9] N. Temenos and P. P. Sotiriadis, "Stochastic computing max & min architectures using markov chains: Design, analysis, and implementation," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 29, no. 11, pp. 1813 – 1823, 2021.
[10] ——, "Nonscaling adders and subtracters for stochastic computing using markov chains," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 29, no. 9, pp. 1612 – 1623, 2021.
[11] V. T. Lee, A. Alaghi, J. P. Hayes, V. Sathe, and L. Ceze, "Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing," in *ACM Proceedings of the Conference on Design, Automation & Test in Europe*, Laussane, Switzerland, Mar. 2017.
[12] P. Ting and J. P. Hayes, "Eliminating a hidden error source in stochastic circuits," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Cambridge, UK, Oct. 2017.
[13] A. Ren, Z. Li, C. Ding, Q. Qiu, Y. Wang, K. Li, X. Qian, and B. Yuan, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," in *ACM 22nd International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS)*, Xi'an, China, Apr. 2017.
[14] V. Canals, A. Morro, A. Oliver, M. L. Alomar, and J. L. Rosselo, "A new stochastic computing methodology for efficient neural network implementation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, Mar. 2016.
[15] C. M. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed. American Mathematical Society, 1997.
[16] S. Liu and J. Han, "Toward energy-efficient stochastic circuits using parallel sobol sequences," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 7, pp. 1326 – 1339, Jul. 2018.
[17] Y. Liu and K. K. Parhi, "Computing polynomials using unipolar stochastic logic," *ACM Journal of Emerging Technologies in Computing Systems*, vol. 13, no. 3, May 2017.
[18] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "Freepdk: An open-source variation-aware design kit," San Diego, CA, USA, Jun. 2007.
[19] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using Matlab*, 2nd ed. Gatesmark Publishing.