Deterministic Finite State Machines for Stochastic Division in Unipolar Format

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*—Stochastic computing has been successfully applied in a plethora of applications, including machine learning, computer vision and soft coding/decoding, due to its low complexity, chip area, and power consumption advantages, as well as its tolerance to soft errors. Among the four fundamental numerical operations, addition, subtraction and multiplication are simple to realize stochastically. Division however is significantly more challenging and complex. This work introduces a new architecture for stochastic division in unipolar format using a deterministic finite state machine. In contrast to the existing architectures, the proposed divider does not require any internal stochastic number generator, which makes it more versatile, compact and easy to implement. The divider's accuracy is defined based on mean absolute error metrics and it is estimated using MATLAB simulation. Applications of the proposed divider in image processing are presented demonstrating its accuracy and efficiency in realistic systems.**

***Index Terms*—Stochastic Computing, Stochastic Divider, Stochastic Circuits**

I. Introduction

Unconventional computing methods have gained increased attention in the design of modern Digital Signal Processing (DSP) and arithmetic units [1]–[3]. Among them, the encoding of information in the form of stochastic sequences and the processing of them, known as Stochastic Computing (SC) [4], is a promising one [1]–[3], [5]–[7].

Considering the stochastic nature of the processed signals, SC is inherently error tolerant; soft-errors, such as bit flips, do not significantly affect the result of calculations. Furthermore, the implementation of basic arithmetic operations is performed by standard logic cells, for example an AND gate in the case of multiplication [2]–[4].

In order to be processed by SC elements, a binary number must be first converted into the stochastic domain. This procedure is executed by a Stochastic Number Generator (SNG), as shown in Fig. 1. The Linear-Feedback Shift Register (LFSR) operates as a pseudo-random number generator and produces on each clock cycle a k-bit random word. Afterwards, it compares with the selected binary number of the same length (k-bit) and creates the stochastic sequence. The length of the generated sequence, $N = 2^k$, is also referred to as *stochastic precision* of the sequence [2].

The generated stochastic sequence is finite (word) and represents a real number in range $[0, 1]$, i.e. $\{C_n, n = 1, 2, ..., N\}$. Here, $\{C_n\}$ are considered to be independent and identically distributed (i.i.d.) Bernoulli random variables, whose expected

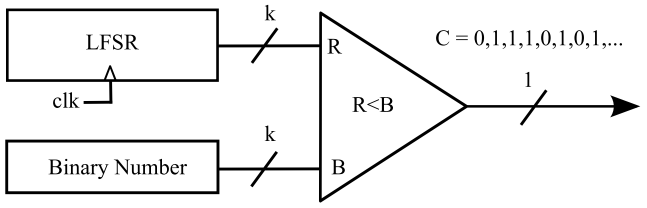

Fig. 1. Stochastic Number Generator (SNG) circuit

value is $\mathbb{E}[C_n] = p$. The value of the word is defined as the average of ones in it, namely,

$$\overline{C} = \frac{1}{N}\left(C_1 + C_2 + \cdots + C_N\right).$$

Undoubtedly, the stochastic precision N is directly associated with the accuracy of the word, which is increased at the cost of additional clock cycles.

Signed number representation is well supported in SC for positive and negative numbers. The former is represented in range $[0, 1]$, named unipolar format, whereas the latter belongs in range $[-1, 1]$, called bipolar format. Mapping from unipolar to bipolar, is achieved by $P_r(C_b = 1) = 2 \cdot P_r(C_n = 1) - 1$. In addition, according to the format used, each SC element implements a different function. Multiplication in bipolar format is performed by XNOR, opposing the AND gate in the unipolar format. SC elements, fall under the principle of combinational logic and thus it is expected that realization of non-linear functions, like division, can be extremely challenging [8]–[10].

Current work proposes a new stochastic divider architecture in unipolar format. Its operation is based on a Finite State Machine (FSM), which relies on input signals and current state to produce a directly stochastic output, without requiring an additional SNG, by exploiting a binary register.

In the following section, the algorithm of the proposed stochastic divider architecture is provided accompanied by a brief comparison with the original one. In section III, simulation results regarding performance in terms of computational accuracy are shown. A proof-of-concept with two applications in digital image processing, is demonstrated in section IV. Finally, in section V, our work is concluded.

II. Stochastic Division

The proposed architecture is inspired by the divider architecture in Fig. 2, introduced in [4]. Inputs $\{X_n\}$ and $\{Y_n\}$

are generated by SNGs while $\{Z_n\}$ is the output, under the assumption that $\mathbb{E}[X_n] \leq \mathbb{E}[Y_n]$ with $\mathbb{E}[Y_n] \in (0,1]$. The main concept here is that the feedback loop of the architecture enforces the equality $X_n = Z_{n-1} \cdot Y_n$ in a statistical sense and essentially converting the operation of division into multiplication.

Central part of the divider is the k-bit up/down counter. At the beginning of the operation, the register of the counter is preset to a fixed initial value, c, independent of the input sequences and their mean values. The selection of c should be done according to the range of $\mathbb{E}[X_n]/\mathbb{E}[Y_n]$ in each application, as it impacts the convergence rate of the loop.

The combination of $X_n = 1$ and $U_n \triangleq Z_{n-1} \cdot Y_n = 0$ increases the counter's value by 1-bit, whereas the case $X_n = 0$ and $U_n = 1$, decreases it by 1-bit. For the remaining input combinations the counter holds its current value. Finally, the SNG converts the binary number into a stochastic $0, 1$ output.

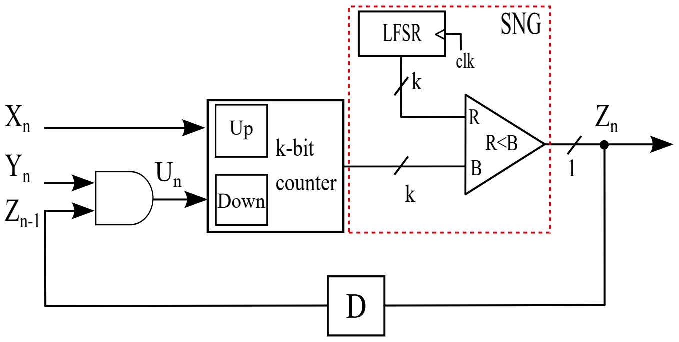

Fig. 2. The original Divider introduced in [4]

Although the divider operates properly, the time needed for it to converge, can be extremely long [2], [8]. As one would expect, this is the case when the initial preset value, with respect to the number of states, i.e $c/2^k$, is far from $\mathbb{E}[X_n]/\mathbb{E}[Y_n]$.

A. Proposed Stochastic Divider Architecture

To address the convergence rate issue of the original divider, we propose the new architecture in Fig. 3. The operation principle of the original divider, i.e. the feedback loop and the logic of up and down counting of the bits in the sequences $\{X_n\}$ and $\{U_n\}$, respectively, is preserved. The SNG however is replaced by a comparison of the counter's value with 0.

As with the original architecture, here the stochastic division $\{X_n\}$ by $\{Y_n\}$ is transformed to the multiplication of $\{Y_n\}$ with $\{Z_n\}$ and stochastic equation of the result with $\{X_n\}$. The loop forces this stochastic equation and the counter, counts the 1s "owned" to the output $\{Z_n\}$.

Again, the value of the counter, S_n, transitions within the $M = 2^m$ states $\{0, 1, ..., M-1\}$ according to the inputs, $\{X_n\}$ and $\{U_n\}$. It counts $+1$ if the inputs are $X_n = 1$ and $U_n = 0$, and, -1 if the inputs are $X_n = 0$ and $U_n = 1$; it maintains its value otherwise. Since $U_n = Y_n \cdot Z_{n-1}$, the iteration will always result in non-negative values of the counter. To account for the possible overflow of the counter the complete expression of the iterative value of S_n is

$$S_n = \min\Big(S_{n-1} + X_n - Y_n \cdot Z_{n-1}\ ,\ M-1\Big).$$

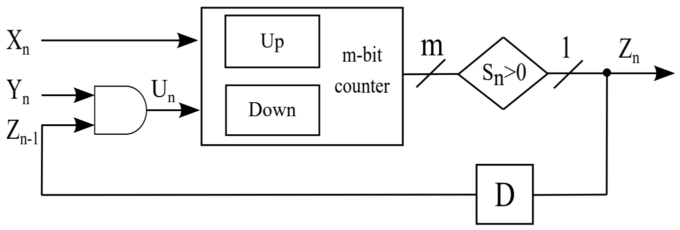

Fig. 3. Proposed Stochastic Divider Architecture

B. Finite State Machine Model of the Proposed Divider

The operation of the proposed stochastic divider is modeled by the Finite State Machine (FSM) shown in Fig. 4.

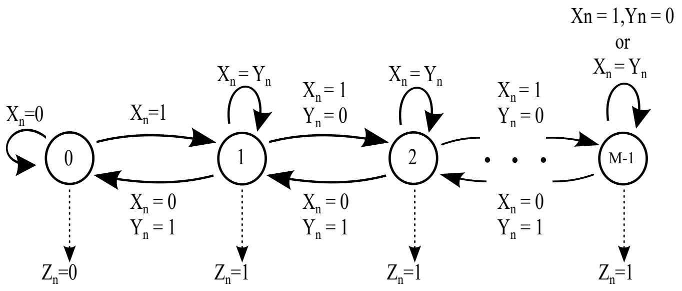

Fig. 4. Finite State Machine of the proposed Stochastic Divider

To explain the derivation of the FSM based on the architecture, we consider the counter's value S_n in $\{0, 1, ..., M-1\}$, $M = 2^m$ at time n. Let

(a) $S_{n-1} = 0$. Then, we see in both the architecture and the FSM that $Z_{n-1} = 0$. In the architecture this implies $U_n = 0$ and therefore $S_n = X_n$. This is captured in the FSM by the transition to $S_n = 1$ if and only if $X_n = 1$.

(b) $S_{n-1} \in \{1, 2, ..., M-2\}$. Then, in both the architecture and the FSM it is $Z_{n-1} = 1$. In the architecture this implies $U_n = Y_n$ and therefore $S_n = S_{n-1} + X_n - Y_n$, in Real Algebra arithmetic. This is captured in the FSM by the transition from S_{n-1} to $S_n = S_{n-1} + 1$ if $X_n = 1$ and $Y_n = 0$, or from S_{n-1} to $S_n = S_{n-1} - 1$ if $X_n = 0$ and $Y_n = 1$, or, no transition, i.e. $S_n = S_{n-1}$ if $X_n = Y_n$.

(c) $S_{n-1} = M - 1$. Then, again in both the architecture and the FSM it is $Z_{n-1} = 1$. In the architecture this implies that $U_n = Y_n$ but in this case it is $S_n = \min(S_{n-1} + X_n - Y_n, M)$, in Real Algebra arithmetic, due to the register's overflow in case of $X_n - Y_n = 1$. This is captured in the FSM by maintaining $S_n = M - 1$ if $X_n \geq Y_n$ and $S_n = M - 2$ if $X_n = 0$ and $Y_n = 1$.

C. Performance of the Proposed Divider

The output accuracy of the proposed stochastic divider is justified using Monte Carlo simulation. For mean value of X

ranging from 0.1 to 0.9 with step of 0.1 and for mean value of Y ranging from X to 0.9 with step of 0.1, we generate $K = 10,000$ pairs of random $0, 1$ sequences $\{X_n\}$ and $\{Y_n\}$ such that $\mathbb{E}[X_n] = X$ and $\mathbb{E}[Y_n] = Y$. For every pair $i = 1, 2, \ldots, K$ of sequences $\{X_n\}$ and $\{Y_n\}$, we calculate the time averages $\bar{X}_i$, $\bar{Y}_i$ and $\bar{Z}_i$ of the input and output sequences respectively. Then, we derive the Mean Absolute Error (MAE) between the output of the proposed divider and the numerical division, $\bar{X}_i/\bar{Y}_i$, namely

$$MAE = \frac{1}{K} \sum_{i=1}^{K} |\bar{Z}_i - \bar{X}_i/\bar{Y}_i|. \qquad (1)$$

The results of the procedure for $N = 128$ and 256 bit stochastic precision using an $m = 5$ bit register are shown in Fig. 5. The MAE dos not exhibit a uniform behavior. It is relatively high for small values of $\mathbb{E}[X_n]$ and $\mathbb{E}[Y_n]$ and it gradually decreases when the expected values of the inputs increase.

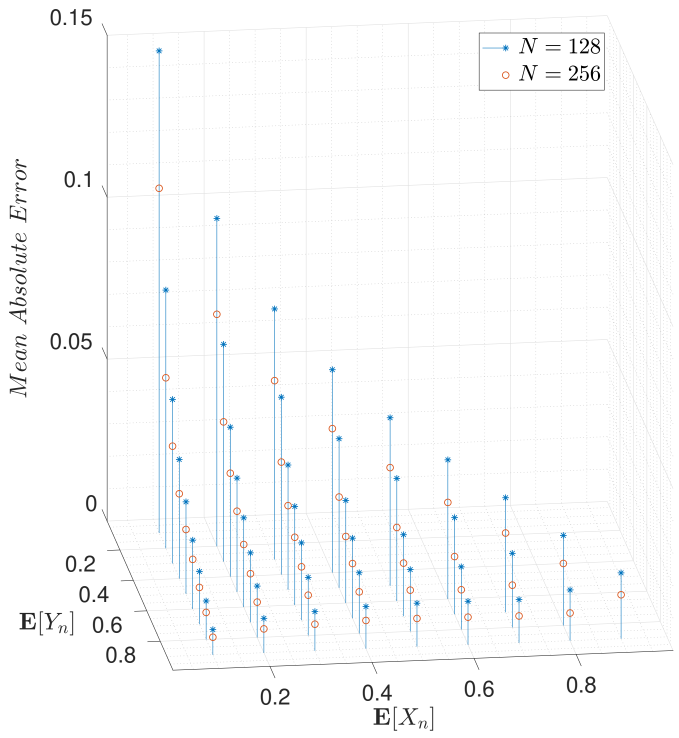

Fig. 5. Mean Absolute Error of the proposed stochastic divider for two selected N bit sequence lengths

D. Comparison of the Convergence Rate

A comparison of the converging rate of the division processes of the two architectures is shown in Fig. 6. We generated $1,000$ pairs of random sequences $\{X_n\}$, $\{Y_n\}$ with expected values equal to $\mathbb{E}[X_n] = 0.25$ and $\mathbb{E}[Y_n] = 0.85$ respectively, resulting in $\mathbb{E}[Z_n] \approx 0.2941$. In both architectures, the register size is $k, m = 8$ bits.

The register of the proposed architecture has, by default, initial value $S_0 = 0$ independently of the mean values of the input sequences. In the original architecture however, the initial value c of the register must be preset according to $\mathbb{E}[Z_n]$. Therefore, we considered three cases to investigate its converging rate: 1) we intentionally set $c = 20$, i.e. $c/2^k = 0.0781$, a value far away of $\mathbb{E}[Z_n]$, 2) we set $c = 65$, yielding $c/2^k = 0.2539$ which is very close to $\mathbb{E}[Z_n]$ and 3) we set the almost equal to $\mathbb{E}[Z_n]$ value of $c = 75$, resulting in $c/2^k = 0.2930$.

As shown in Fig. 6, especially when $c/2^k$ is far from $\mathbb{E}[X_n]/\mathbb{E}[Y_n]$, the original divider approximates the desired output after a considerable number of clock cycles. Nevertheless, if the offset is minimal, it does not experience severe converging issues. Eventually the converging rate is minimized in the case when $c/2^k$ is almost equal to the desired output. On the other hand, the proposed stochastic divider calculates the result relatively fast without requiring an excessive amount of states.

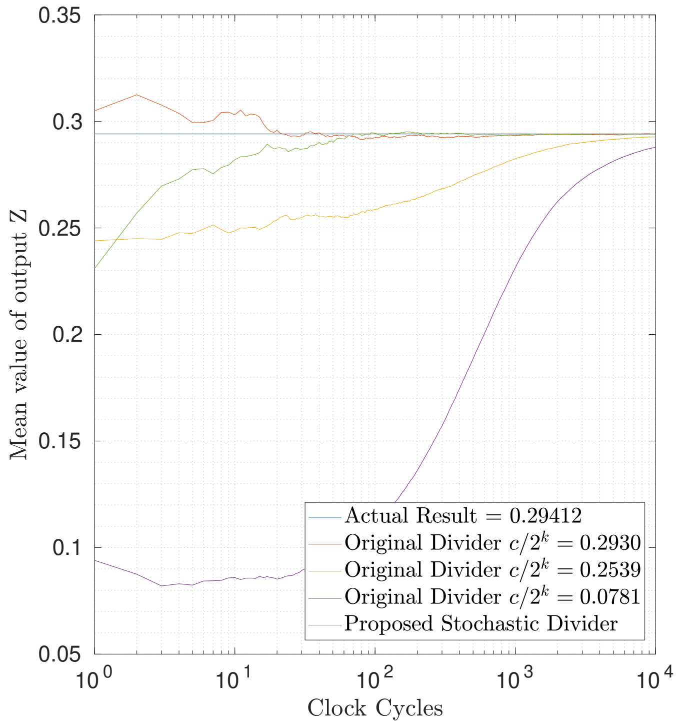

Fig. 6. Comparison of convergence in clock cycles between the original [4] and the proposed stochastic divider architecture. Output averaged over $1,000$ random input sequences pairs. The original divider starts with 3 different initial conditions

III. Applications of the Proposed Divider in Digital Image Processing

Applications in the field of digital image processing can benefit well from SC. The main advantage comes from massive parallelism; processing can be applied pixel-wise throughout an entire image [2], [11]–[13]. Furthermore, the cost in computational cycles is typically low, usually $N = 256$ when using $k = 8$ bit numbers, but varies according to the design requirements.

We demonstrate the efficacy of the proposed stochastic divider for digital image processing using two applications in Matlab. The first one, corresponds to pixel division, which reduces the dynamical scale of an image [14], while the second to the data normalization algorithm [14].

Application 1: We select a Matlab built-in grayscale image with high contrast, i.e. the intensities span across the entire range of $[0, 255]$. After mapping them to range $[0, 1]$, we apply our algorithm on each pixel $P_{m,n}$ across the entire image of size $m \times n$ and scale each one down by a constant factor of 0.85, as a test case.

The stochastic precision bits selected are $N = 512$ with a register of $m = 5$ bits. As a figure-of-merit for the comparison between the stochastic divider and the simulated division, we calculated the peak signal-to-noise ratio (PSNR). For the results in Fig. 7 it corresponds to PSNR $= 30.96dB$.

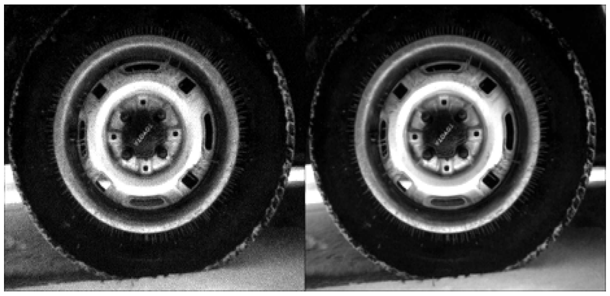

Fig. 7. Left: pixel division using the proposed stochastic divider of precision $N = 512$ bits Right: Matlab simulated division

Application 2: Image (or data) normalization, is a technique used to expand the pixel intensity values of a low-contrast image into its full range [14]. The linear normalization for each pixel $P_{m,n}$ is performed according to

$$Pnew_{m,n} = (P_{m,n} - a) \cdot \frac{b_{new} - a_{new}}{b - a} + a_{new}, \qquad (2)$$

where a and b denote the minimum and maximum pixel intensity values respectively.

In our example, a default low-contrast image in grayscale was selected and simulated as proof-of-concept. First, we identified the reduced range of the grayscale image $[a, b]$ and mapped it to the corresponding range $[0, 1]$.

Each pixel value, was then normalized to the new desired range $[a_{new}, b_{new}]$, using the proposed stochastic divider on each pixel $P_{m,n}$, after subtracting the former with the image's minimum intensity a. Here, the new intensities a_{new} and b_{new}, were selected to be the bottom 1% and top 99% of $[0, 1]$.

Note that, in our case the quantity $b_{new} - a_{new}$ was very close to 1 and thus the multiplication from eq. (2) was omitted. The same applied to the addition with a_{new}, since it was a negligible value and would not significantly affect the result of the calculation. Otherwise, if $b_{new} - a_{new}$ was not close to 1, an AND gate would be required to perform the multiplication as well as the addition of the division's result with a_{new}.

Each pixel was represented with a sequence length of $N = 1024$ bits and the register's size was equal to $m = 6$ bits. The output image was then compared with the corresponding calculation performed by Matlab's imadjust function [14], while the results of the procedure, yield a $PSNR = 32.78dB$. In Fig. 8 the image normalization performed by the two methods is shown accompanied by the original image for comparison.

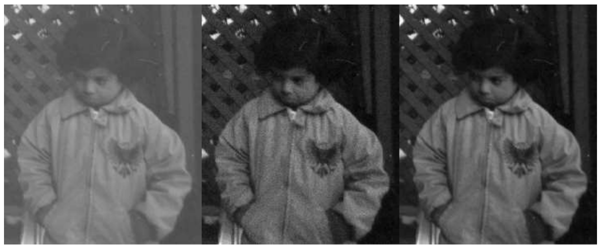

Fig. 8. Left: Original low-contrast image Middle: image normalization using the proposed stochastic divider with $N = 1024$ bits Right: Matlab's normalization using imadjust function [14]

IV. Conclusion

A new architecture for stochastic division in unipolar format using FSMs was presented. It has the advantage that it does not need the hardware costly stochastic number generator which was necessary in previous architectures. Furthermore, it achieves satisfactory mean absolute output error, even with relatively small number of stochastic precision bits as well as fast convergence rate. Two applications of the proposed stochastic divider architecture in digital image processing demonstrated its effectiveness.

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 traveling and conference attendance expenses.

References

[1] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2, May 2013.

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

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

[6] 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.

[7] N. Saraf and K. Bazargan, “Polynomial arithmetic using sequential stochastic logic,” in *IEEE International Great Lakes Symposium on VLSI (GLSVLSI)*, Boston, MA, USA, May 2016.

[8] T. Chen and J. P. Hayes, “Design of division circuits for stochastic computing,” in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Pittsburgh, PA, USA, Jul. 2016.

[9] N. Temenos and P. P. Sotiriadis, “A new technique for stochastic division in unipolar format,” in *IEEE International Conference on Modern Circuits and System Technologies (MOCAST)*, Thessaloniki, Greece, May 2019.

[10] D. Wu and J. S. Miguel, “In-stream stochastic division and square root via correlation,” in *Proceedings of the 56th Annual Design Automation Conference 2019 (DAC)*, Las Vegas, NV, USA, Jun. 2019.

[11] P. Li and D. J. Lilja, “Using stochastic computing to implement digital image processing algorithms,” in *IEEE 29th International Conference on Computer Design (ICCD)*, Amherst, MA, USA, Oct. 2011.

[12] 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 (VLS)*, vol. 2, no. 3, pp. 449–462, Apr. 2014.

[13] A. Alaghi, C. Li, and J. P. Hayes, “Stochastic circuits for real-time image-processing applications,” in *IEEE 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Austin, TX, USA, May 2013.

[14] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using Matlab*, 2nd ed. Gatesmark Publishing, 2009.