A New Technique 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 (SC) is an alternative designing technique where signals are processed non-deterministically. Apart from low-area occupation and greatly reduced power dissipation, SC is inherently fault tolerant due to its probabilistic nature. Considering the aforementioned, SC has regained attention in system design where traditional Digital Signal Processing (DSP) cores require demanding number of resources to perform operations or are sensitive to soft-errors. However, specific operations are considered challenging for implementation due to the fact that processing elements are incapable of constructing non-linear functions such as the division. In this work we propose a new architecture that performs stochastic division in unipolar format that produces direct stochastic output and lowers the hardware requirements. Simulation results of Normalised Mean Root Squared Errors (NRMSE) are provided in order to demonstrate the accuracy of the proposed architecture as well as the simplicity for implementation.**

***Index Terms*—Stochastic Computing, Stochastic Divider, Stochastic Circuits, Fault Tolerance, Low-Power Design**

I. Introduction

Stochastic Computing (SC) was proposed in [1] as a system designing technique where Digital Signal Processing (DSP) elements process signals non-deterministically. Although at a glance, the idea of performing operations stochastically may not seem effective, SC has regained attention due to numerous advantages it offers. First and foremost, SC is naturally fault tolerant; soft-errors such as bit-flips do not affect the result of a calculation. Furthermore, the implementation of basic operations such as multiplication can be performed by a single-gate, i.e. an AND gate. Finally, as a result of the low-area occupation of processing elements, SC is also characterized by low-power consumption.

Regarding the application level, SC is suitable in systems where parallelization and high demand in hardware resources are necessary [2]. In image processing, the algorithm performing edge detection can be applied in parallel blocks across an entire image with low stochastic bit precision requirements for signal representation [2], [3]. Furthermore, in the field of Artificial Neural Networks (ANN), the implementation cost is extremely challenging due to the large number of parallel multipliers used in binary format [4]–[6]. Moreover, ANNs do not require high precision results, rather than emphasize on behavior, where SC is capable of following effectively [4]–[8]. Another successful application for SC is decoding Low-Density Parity-Check (LDPC) and other several modern error-correcting codes, where the sum and product units are present [2], [3].

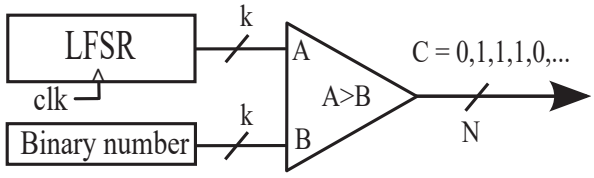

Fig. 1. Circuit for the Stochastic Number Generator (SNG)

In order to represent data into the stochastic domain, a Stochastic Number Generator (SNG) circuit to convert the binary number is required, as shown in Fig. 1. More specifically, a pseudo-random number generator, which is implemented as a Linear Feedback Shift Register (LFSR), is fed with an initial k-bit binary number that produces a k-bit word which is repeated every $2^k - 1$ clock cycles. Afterwards, the k-bit word is compared with a selected binary number of the same length in order to create the stochastic sequence of $N = 2^k$ bits for further process [1], [9], [10].

The converted stochastic sequence, can be either a positive or negative signed number. In SC, positive numbers are represented in $[0, 1]$ range, called unipolar format, whereas the negative ones in $[-1, 1]$ range, called bipolar format. Advancing to the computational elements, i.e. AND, OR, NOT, XOR and MUX, their operation is based on fundamental boolean algebra [1]. Moreover, according to the format used, each gate implements a different function and thus the selection for the number format can be intriguing. Therefore, it is expected that specific arithmetic operations, such as division, can be complicated due to the fact that they cannot be represented effortlessly via set theory.

The theoretical model for division in SC was firstly presented in [1]. However, the aforementioned model suffers from long computational cycles and demands large number of resources to be implemented [3], [8], [11]. In this work we propose a stochastic divider architecture in unipolar format that uses a binary accumulator, which operates stochastically in order to eliminate the former issues. In the section follows, the probabilistic properties of conventional logic gates for positive signed numbers are briefly described. In section III, the proposed architecture that performs division is explained and simulations to prove its operation are shown. Finally,

978-1-7281-1184-1/19/$31.00 ©2019 IEEE

section IV, provides the conclusion.

II. Probabilistic Properties of Logic Gates in Unipolar Format

In this section, we provide a brief description of the stochastic operation of basic logic gates in unipolar format, in order to get a better understanding of how SC elements process data.

We assume that the finite stochastic sequence (word) representation of a real number in $[0, 1]$ is done by the output of the SNG circuit, i.e. $\{X_n, n = 1, 2, ..., N\}$. Here $\{X_n\}$ are considered to be independent and identically distributed (iid) Bernoulli random variables, N is the length of the stochastic word and n is the time index. The Probability Mass Function (PMF) of each random variable is given by:

$$p_{X_n}(x) = \begin{cases} p, & \text{x = 1} \\ q = 1 - p, & \text{x = 0} \end{cases}. \tag{1}$$

The value of a realization of word $\{X_n\}$ is defined to be the average of the ones in it, i.e.,

$$\frac{1}{N}(X_1 + X_2 + \cdots + X_N)$$

whose expected values is $\mathbb{E}[X_i] = p$.Therefore, by increasing the length of the word, N, also referred as *stochastic precision bits*, the accuracy increases [1].

To describe the probabilistic behaviour of the following logic gates, we denote their two inputs as X and Y and their output as Z, where X and Y are random independent variables.

A. NOT Gate

The output of the NOT gate, i.e. $Z = 1 - X$, has expected value

$$P_r(Z = 1) = P_r(X = 0) = 1 - P_r(X = 1) \tag{2}$$

and achieves the probability complement.

B. AND Gate

The AND gate $Z = X \cdot Y$ can be seen as a multiplication and has expected value

$$P_r(Z = 1) = P_r(X = 1, Y = 1) = P_r(X = 1) \cdot P_r(Y = 1). \tag{3}$$

C. OR Gate

The OR gate can be expressed as $Z = X + Y - X \cdot Y$ and results in output expected value

$$\begin{aligned} P_r(Z = 1) &= P_r(X = 1) + P_r(Y = 1) - P_r(X = 1, Y = 1) \\ &= P_r(X = 1) + P_r(Y = 1) \\ &\quad - P_r(X = 1) \cdot P_r(Y = 1) \end{aligned} \tag{4}$$

D. XOR Gate

The XOR gate is written as $Z = X + Y - 2 \cdot X \cdot Y$ and results in output expected value

$$\begin{aligned} P_r(Z = 1) &= P_r(X = 1) + P_r(Y = 1) \\ &\quad - 2 \cdot P_r(X = 1, Y = 1) \\ &= P_r(X = 1) + P_r(Y = 1) \\ &\quad - 2 \cdot P_r(X = 1) \cdot P_r(Y = 1) \end{aligned} \tag{5}$$

E. MUX Gate

In the MUX gate, the select signal S is also assumed to be a Bernoulli random variable independent of the inputs. Then the output $Z = S \cdot X + (1 - S) \cdot Y$ has expected value

$$\begin{aligned} P_r(Z = 1) &= P_r(S = 1) \cdot P_r(X = 1) + \\ &\quad (1 - P_r(S = 1)) \cdot P_r(Y = 1) \end{aligned} \tag{6}$$

Therefore, the MUX operates as a scaling adder, with the special case of $P_r(S = 1) = 1/2$ giving

$$P_r(Z = 1) = \frac{P_r(X = 1) + P_r(Y = 1)}{2} \tag{7}$$

III. Stochastic Division

In this section we discuss the principle of stochastic division as it was introduced and we proceed with our proposed architecture along with simulation results.

A. Proposed Stochastic Divider

The stochastic divider was first proposed by Gaines in [1] as shown in Fig. 2, where X_n and Y_n are the two inputs for division and Z_n is the output with $Z_n = X_n/Y_n$. The main concept is to create a feedback loop from the output Z_n and multiply with Y_n in order to remove the division operation, resulting in $X_n = Z_n \cdot Y_n = U_n$.

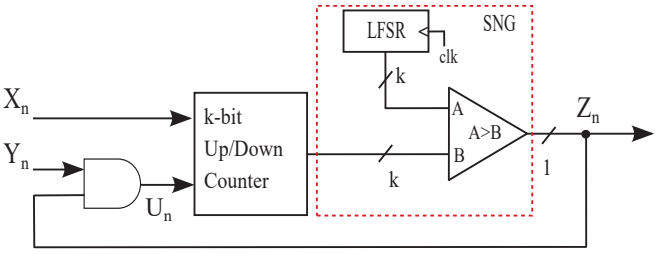

Fig. 2. Original Divider introduced by Gaines

The processing is based on a k-bit up/down counter, where the simultaneous appearance of $X_n = 1$ and $U_n = 0$, increases the accumulator by 1 bit, whereas the reverse case $X_n = 0$ and $U_n = 1$ decreases by 1 bit. On any other combination of inputs, the counter value remains unchanged. Then, a SNG is used to convert the binary value of the counter into a stochastic one and the procedure is restarted until the system achieves equilibrium.

However, the model is impractical due to the following reasons; the additional required SNG increases the total area for implementation significantly [9], [10] and the converge

time to the approximate result can be long [3], [12]. In order to avoid the aforementioned issues, we propose the stochastic divider shown in Fig. 3. In our configuration, we consider an accumulator which transitions between M states given by state space $S_n \in \{0, 1, ..., M-1\}$, where each state S_n represents the position of a one-hot encoded M-bit register.

Once again, the simultaneous appearance of $X_n = 1$ and $U_n = 0$, increases the accumulator by 1-bit while $X_n = 0$ and $U_n = 1$ decreases by 1-bit. Moreover, the uniqueness of the proposed scheme is that the increments and decrements in the accumulator, are implemented as left and right shifts in the M-bit register and thus the traditional binary operation is eliminated completely. Furthermore, we use an additional delay register in order to avoid correlation of Z_n [1]. As a result, the proposed architecture converges to the approximate result after N clock cycles and achieves a decrease in overall hardware resources required.

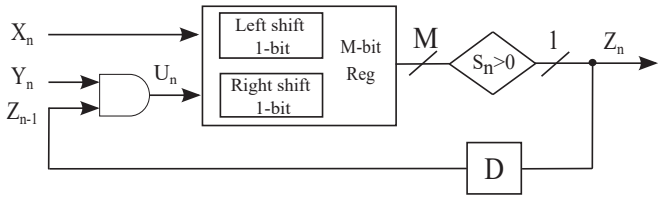

Fig. 3. Proposed Stochastic Divider

B. Simulation Results

In this subsection, we prove the functionality of the proposed stochastic divider architecture with simulations using Matlab. In order to capture all possible values for division in the range of $[0, 1]$, we generate two signals under the assumption of $\mathbb{E}[Y] > 0$ and $0 < \mathbb{E}[X]/\mathbb{E}[Y] <= 1$. To achieve the aforementioned, we consider a sweep where the inputs of the expected values increase simultaneously, with $\mathbb{E}[X] = 0.1 : 0.1 : \mathbb{E}[Y]$ and $\mathbb{E}[Y] = 0.1 : 0.1 : 0.9$. As a result, we have ensured that the expected result will not reach infinity due to possible appearance of $\mathbb{E}[Y] = 0$.

Since the input signals are stochastic, it is reasonable to perform a sufficient number of calculations in order to obtain the error behavior for each different division pair tested. For this reason, we consider the Normalised Root Mean Squared Error (NRMSE) between the stochastic and the deterministic division, namely:

$$NRMSE = \frac{\sqrt{\sum_{i=1}^{K} \frac{(D_i - SC_i)^2}{K}}}{D} \tag{8}$$

where $K = 10,000$ and denotes the number of iterations performed for each generated division pair of $\mathbb{E}[X]$ and $\mathbb{E}[Y]$.

In Fig. 4, the results for the aforementioned procedure are shown for $N = 128$-bit stochastic precision. Notice that it is more difficult to perform the operation when the expected values are relatively close to each other and close to zero as well, for example $\mathbb{E}[X] = \mathbb{E}[Y] = 0.1$ with corresponding NRMSE of 0.3. On the contrary, for expected values with higher frequency of logic $'1's$, the proposed architecture performs division without affecting the calculation result, with typical NRMSE values of < 0.1. Moreover, we provide with the corresponding NRMSE results for $N = 256$-bit precision, as shown in Fig. 5. Notice that the highest error values are reduced. Therefore, it is evident that when the stochastic precision increases, the NRMSE decreases.

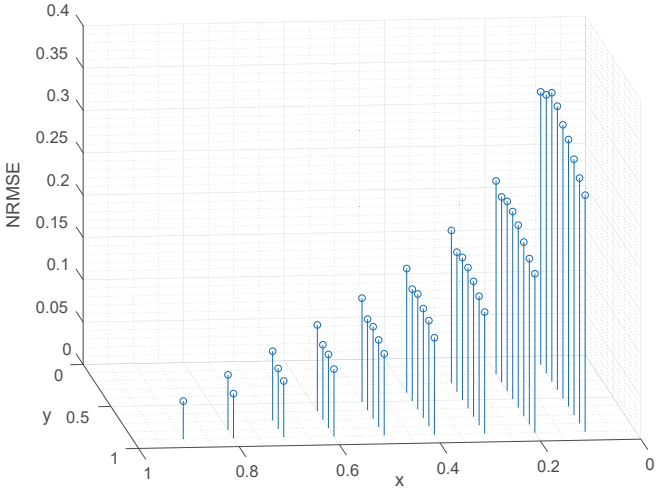

Fig. 4. NRMSE for $N = 128$ stochastic precision

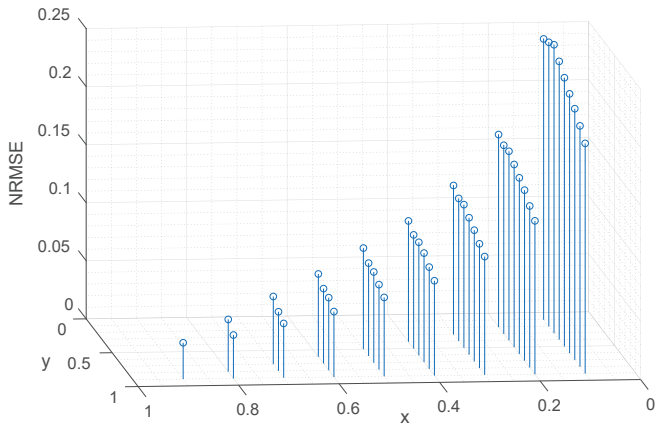

Fig. 5. NRMSE for $N = 256$ stochastic precision

We also performed simulations in order to investigate the minimum allowable size of the accumulators' M-bit register in order to avoid overflows. The procedure explained in previous paragraph was repeated and the stochastic precision was set in powers of two, namely $N = 64, 128, 256, 512, 1024$. The results for selected precision bits are cited in Table I.

TABLE I
ACCUMULATOR M-BIT REGISTER SIZE

Bit Precision N	64	128	256	512	1024
Register Size	12	14	19	21	23

C. Accumulator Bit-Level Example

In the final subsection, we examine the stack of the accumulator with a bit-level procedure, under two case scenarios for $N = 10$-bit precision, as presented in table II. In the first case, we assume that the spreading of logic $'1's$ is ideal whereas in the second one, bit-overflow occurs due to insufficient M-bit register size. In the first case, the expected values for division are $\mathbb{E}[X] = 0.7$ and $\mathbb{E}[Y] = 0.8$, with result of $\mathbb{E}[Z] = 0.8$. Although the expected value is close to the desired one $\mathbb{E}[Z] = 0.875$, the inadequate number of precision bits as well as the non-linear nature of division, contribute in the deviation of the result. In the second case, the two input signals were set intentionally with such spreading of $'1's$ in order to introduce overflow. The expected values are respectively $\mathbb{E}[X] = 0.7$ and $\mathbb{E}[Y] = 0.9$. Notice that the final pair of logic $'1's$ is responsible for overflow with the result being $\mathbb{E}[Z] = 0.7$ instead of $\mathbb{E}[Z] = 0.7777 \simeq 0.8$, which is closer to the desired.

TABLE II
STOCHASTIC DIVISION EXAMPLE

Case: 1											
Outputs N	No. Bits										Overflow
Bit	1	2	3	4	5	6	7	8	9	10	
x	1	1	1	1	1	1	1	0	0	0	
y	0	1	0	1	1	1	1	1	1	1	
z	1	1	1	1	1	1	1	1	0	0	
Accumulator	1	1	2	2	2	2	2	1	0	0	**0**
Case: 2											
Outputs N	No. Bits										Overflow
Bit	1	2	3	4	5	6	7	8	9	10	
x	0	0	1	1	1	1	0	1	0	1	
y	1	1	1	1	1	1	1	1	0	1	
z	0	0	1	1	1	1	1	1	0	1	
Accumulator	0	0	1	1	1	1	0	1	0	1	**1**

IV. CONCLUSION

In this work an optimization technique in order to implement the operation of division in Stochastic Computing was presented. The primary objective was to replace the SNG circuit as well as the k-bit up/down counter logic with a costly-efficient one in terms of hardware implementation requirements. Simulation results demonstrate that the proposed architecture can achieve satisfactory results in terms of NRMSE, for the majority of the calculations tested at realizable precision. Moreover, simulations also provided the sufficient register M-bit size for various stochastic precision levels. In conclusion, our configuration can be considered in SC applications that approach division, such as Neural Networks, where low-area specifications are essential.

REFERENCES

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

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

[3] 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.

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

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

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

[7] 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.

[8] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in *IEEE 31st International Conference on Computer Design (ICCD)*, Asheville, NC, USA, Oct. 2013.

[9] M. Yang, B. Li, D. J. Lilja, B. Yuan, and 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, Jul. 2018.

[10] H. Ichihara, T. Sugino, S. Ishii, T. Iwagaki, and T. Inoue, "Compact and accurate digital filters based on stochastic computing," *IEEE Transactions on Emerging Topics in Computing*, to be published.

[11] A. Alaghi and J. P. Hayes, "Fast and accurate computation using stochastic circuits," in *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, Mar. 2014.

[12] 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.