Stochastic Computing Circuits for Nonlinear Functions Based on Multiplexers

Yongqiang Zhang, *Member, IEEE*, Defang Meng, Jie Han, *Senior Member, IEEE*, Shaowei Wang, Guangjun Xie, *Senior Member, IEEE*, and Xin Cheng

***Abstract*—Stochastic computing (SC) is a renewing computing paradigm that differs from weighted binary ones. It operates on binary stochastic bitstreams, or called stochastic numbers (SNs), instead of binary numbers (BNs). This work proposes circuits based on multiplexers (MUXs) to address the issue of SC in computing nonlinear functions. These circuits significantly reduce the number of required stochastic number generators (SNGs) and shorten critical path delay (CPD), thereby lowering hardware costs. Simulation and experimental results demonstrate that the proposed designs outperform existing designs in both computing accuracy and hardware overhead. For example, the proposed design for sin(*x*) using a 3rd-order Maclaurin series achieves averaged reductions of 66.84% in mean square error (MSE) and 24.62% in power-delay product (PDP), respectively, compared with previous designs.**

***Index* Terms—Stochastic computing, multiplexer, nonlinear function.**

I. INTRODUCTION

STOCHASTIC computing (SC) has reemerged as a representative computing paradigm to address circuit complexity and reduce hardware costs. These kinds of circuits have been successfully applied in image processing and neural networks, particularly in computing nonlinear functions [1]. Unlike weighted binary circuits, circuits in SC operate using bitstreams where each bit carries equal weight. The numerical value that a bitstream carries is represented by the probability of a '1' appearing in the bitstream. Converting binary numbers (BNs) into corresponding stochastic bitstreams or stochastic numbers (SNs) requires a stochastic number generator (SNG), which comprises a random number source (RNS) and a comparator. The RNS can be designed using a linear feedback shift register (LFSR) or other pseudo-RNSs [2].

The applications of SC circuits to compute nonlinear functions have become a prominent research focus in recent years. Brown has first implemented exponential (Exp) function using a one-dimensional finite state machine (FSM) [3]. Subsequently, Li has extended this approach by providing theoretical derivations for implementing FSM-based Exp function [4]. A two-dimensional FSM and its corresponding stochastic architecture have been derived from the original one-dimensional FSM [5, 6, 7]. Further developments have led to Integral SC [8], which has been applied to FSMs to reduce the number of states [9]. However, these methods are limited to computing only a few specific nonlinear functions. A significant advancement is a reconfigurable stochastic circuit based on Bernstein polynomials, called RESC, which substantially expanded the range of functions that could be computed [10, 11, 12]. Later improvements have replaced the adder in the RESC with an FSM, resulting in a new reconfigurable architecture [7, 13]. Subsequent research has incorporated Taylor expansion [14, 15, 16, 17, 18] based on polynomials and Horner's method [14, 15] into SC implementations for nonlinear functions.

However, these methods require that all input coefficients remain mutually independent, necessitating the use of distinct SNGs for generating bitstreams. This consequently leads to increased hardware costs. To address this limitation, correlation [19] and weighted-adder [20] based methods have been developed. However, all existing correlation-based implementations employ cascaded chain structures, which suffer from significantly increased critical path delays when computing complex functions. These two works surpass previous works both in computing accuracy and hardware overhead because correlation-based designs generally have higher accuracy and simple circuit complexity.

Inspired by these considerations, this work proposes a circuit design method in SC for nonlinear functions based on MUXs. For this purpose, a given nonlinear function has to be expanded to an *n*th-order polynomial and then decomposed into the multiplication of first- and second-order factors according to the factorization theorem for polynomials with real coefficients. Different circuits are designed for these factors based on MUXs, respectively. Compared to state-of-the-art designs, the proposed circuits achieve improved computing accuracy along with reduced hardware overhead.

II. PROPOSED DESIGNS

To compute nonlinear functions in this work, it has to convert them into polynomials, such as trigonometric (Trig), hyperbolic tangent (tanh), and Exp functions, by using Maclaurin expansions. An *n*th-order polynomial in (1) is then

This work was supported in part by the National Natural Science Foundation of China under Grant 62404067, in part by the Fundamental Research Funds for the Central Universities under Grant JZ2025HGTB0231 and PA2025GDSK0034, and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Grant RES0048688. (Corresponding author: X. Cheng)

Y. Zhang, D. Meng, G. Xie, and X. Cheng are with the School of Microelectronics, Hefei University of Technology, Hefei 230009, China (email: ahzhangyq@hfut.edu.cn; 2024110952@mail.hfut.edu.cn; gjxie8005@hfut.edu.cn; xcheng@hfut.edu.cn).

J. Han is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (e-mail: jhan8@ualberta.ca)

S. Wang is with the School of Integrated Circuits, Anhui University, Hefei 230601, China (e-mail: swwang@ahu.edu.cn)

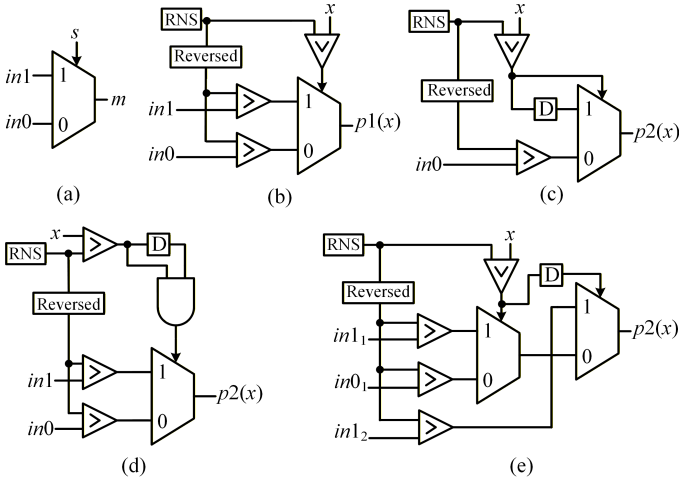

Fig. 1. The proposed circuits based on MUXs. (a) A MUX. (b) Circuit for computing $p_1(x)$ based on a MUX. (c) Circuit for computing $p_2(x)$ based on a MUX. (d) Circuit for computing $p_2(x)$ based on an AND gate and a MUX. (e) Circuit for computing $p_2(x)$ based on cascaded MUXs.

factorized into products of first- and second-order factors, with a scaling factor.

$$f(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + a_n x^n . \quad (1)$$

where a_0, a_1, …, a_n are coefficients, and n is the highest order of $f(x)$. As an example, consider a 3rd-order polynomial in (2).

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 . \quad (2)$$

To factorize (2), the coefficient of the highest order term is extracted as an initial scaling factor. (2) is then reformulated as a product of first- and second-order factors, as in (3).

$$f(x) = a_3 (x + b_1)\left(x^2 + c_1 x + c_2\right). \quad (3)$$

Each factor is then constrained to generate outputs within the range of [0, 1]. For functions with singular points, such as tan(x) near π/2, the proposed method operates within a bounded input range to avoid undefined regions. For factors exceeding this range, scaling should be applied. Subsequently, (3) is transformed into

$$f(x) = a_3 \cdot b \cdot c \cdot \frac{(x + b_1)}{b}\frac{\left(x^2 + c_1 x + c_2\right)}{c}. \quad (4)$$

The coefficients in (4) can be determined by solving relevant equations.

Intuitively, the first-order factor can be implemented using a MUX, while the second-order factor has to be realized by cascading two MUXs. The final scaled result is then obtained through an AND gate.

A MUX with four ports is shown in Fig. 1(a), where the data inputs are connected to ports *in*1 and *in*0, while the result is output through port *m*, and port *s* serves as a selection control terminal. The function of a MUX is expressed as

$$p(m) = p(s)\,p(in1) + \left(1 - p(s)\right)p(in0). \quad (5)$$

where $0<in1,\ in0,\ s,\ m<1$. The notation $p()$ denotes the probability of '1' occurring in a corresponding SN. For example, $p(s)$ represents the probability of '1' in SN *s*, which equals the BN *s*. Note that SNs and the corresponding BNs that they represent are written in the same symbols for simplicity in this paper. Its meaning can easily be understood in a specific context.

A. The First-Order Factor

The first-order factor in (4) is denoted as

$$p1(x) = \frac{(x + b_1)}{b}. \quad (6)$$

To design a stochastic computing circuit for it based on a MUX, three circuit structures are designed by connecting input variable *x* to ports *in*1, *in*0, and *s*, respectively. Mean square error (MSE) values for three cases are simulated. The structure with *x* connected to port *s* yields the lowest MSE value and is therefore selected for further usage. To access this structure, $p1(x)$ has to be reformulated as

$$p1(x) = x \cdot \frac{b_1 + 1}{b} + (1 - x)\frac{b_1}{b}. \quad (7)$$

According to (7), four ports of a MUX are respectively set as $in0=b_1/b$, $in1=(b_1+1)/b$, $s=x$, and $m=p1(x)$, as shown in Fig. 1(b). Two input ports are insensitive to correlation between them and can employ the same RNS [2]. Although the selection port is sensitive to correlation related to input ports, an RNS is shared by reversing the bit order of its outputs [2, 21]. Reconfigurable implementations using LFSRs defined over the Galois field (GF) with primitive polynomials are promising to support multiple nonlinear functions and flexible decorrelation, which will be explored in future work.

B. The Second-Order Factor

The second-order factor in (4) is denoted as

$$p2(x) = \frac{\left(x^2 + c_1 x + c_2\right)}{c}. \quad (8)$$

Since the presence of x^2, a MUX requires two input ports for *x*, leading to 3 structures, one of which is illustrated in Fig. 1(c). Additionally, one may consider structures combining an AND gate with a MUX, as well as cascaded MUXs. The AND gate and MUX result in 3 combined structures, one of which is shown in Fig. 1(d). Cascaded MUXs give rise to 13 structures, with an example shown in Fig. 1(e). For each of the 19 structures described above, the feasible coefficients for implementing second-order factors are analyzed, and the corresponding MSE values are measured. Among all structures capable of implementing the same coefficients, the one with the lowest MSE value is selected.

To implement (8) under the assumption $c1<0$, according to the MSE values, the cascaded-MUXs structure is selected. (8) can be rewritten as

$$\begin{aligned} p2(x) &= x \cdot in1_2 + (1 - x)m_1 \\ &= x \cdot in1_2 + (1 - x)\left(in1_1 \cdot x + in0_1 - x \cdot in0_1\right) \\ &= \left(in0_1 - in1_1\right)x^2 + \left(in1_2 + in1_1 - 2in0_1\right)x + in0_1 \end{aligned}, \quad (9)$$

where m_1 denotes the output of the first MUX, and $in0_1$ $in1_1$, and $in1_2$ are coefficients, which can be computed as

$$in1_1 = \frac{c_2 - 1}{c},\ in0_1 = \frac{c_2}{c},\ \text{and } in1_2 = \frac{c_1 + c_2 + 1}{c}. \quad (10)$$

In (9) and (10), the SNs for the independent variable *x* and coefficients are all generated from the same RNS. Decorrelation between *x* and coefficients is achieved using line-swapping method [2]. The input variable *x* supplied to

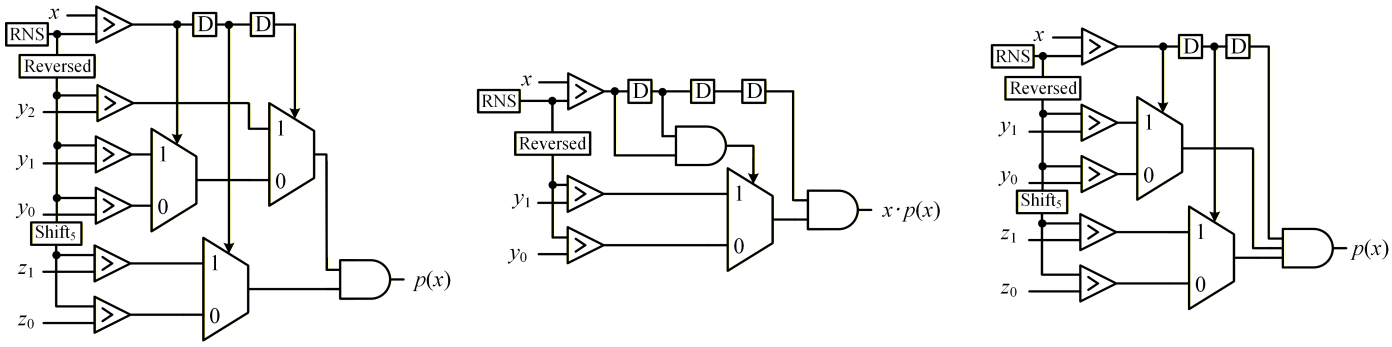

Fig. 2. MUX-based circuits for computing 3rd-order polynomials of (a) exp(-*x*), (b) tanh(*x*), and (c) sin(*x*).

TABLE I
COMPUTING ACCURACY OF EXAMPLES 1-3

Methods	*exp*(-*x*)		*tanh*(*x*)		*sin*(*x*)	
	MAE (10^{-3})	MSE (10^{-4})	MAE (10^{-3})	MSE (10^{-4})	MAE (10^{-3})	MSE (10^{-4})
NAND-AND	29.51	11.43	35.91	23.30	10.08	1.47
Weighted-adder	15.78	3.47	35.16	22.80	9.11	1.26
Proposed	15.21	3.39	32.49	20.12	5.05	0.45

TABLE II
HARDWARE COSTS OF EXAMPLES 1-3

Functions	Methods	Area (10^2)	CPD	Power (10^{-2})	ADP (10^2)	PDP (10^{-2})
exp(-*x*)	NAND-AND	3.80	1.23	3.23	4.67	3.97
	Weighted-adder	2.92	1.16	1.93	3.39	2.24
	Proposed	2.59	1.11	1.82	2.87	2.02
tanh(*x*)	NAND-AND	3.12	1.20	3.04	3.74	3.65
	Weighted-adder	2.21	1.26	1.79	2.78	2.26
	Proposed	1.97	1.07	1.76	2.11	1.88
sin(*x*)	NAND-AND	3.12	1.20	3.04	3.74	3.65
	Weighted-adder	2.23	1.09	1.76	2.43	1.92
	Proposed	2.23	1.09	1.74	2.43	1.90

Area: μm^2; CPD: ns; Power: mW; ADP: μm^2·ns; PDP: mW·ns.

different MUXs is maintained uncorrelated through D flip-flops (DFFs) [22], as in Fig. 1(e).

III. IMPLEMENTATIONS OF NONLINEAR FUNCTIONS

As representative nonlinear functions, SC circuits for Exp, tanh, and sin are implemented in detail to illustrate the effectiveness of the proposed design method. These functions are expanded using 3rd-order Maclaurin series, respectively.

A. Exponential Function

$$\begin{aligned} exp(-x) &= 1 - x + \frac{x^2}{2} - \frac{x^3}{6} \\ &= \left(-\frac{1}{6}\right)(x - 1.5961)\left(x^2 - 1.4039x + 3.7592\right), \\ &= \frac{(1.5961 \times 3.7592)}{6} p(x) \end{aligned} \quad (11)$$

where $p(x)$ is denoted as the multiplication of $p1(x)$ and $p2(x)$ and computed as

$$\begin{aligned} p(x) &= p1(x) \cdot p2(x), \\ p1(x) &= 1 - \frac{x}{1.5961},\ p2(x) = \frac{x^2}{3.7592} - \frac{1.4039x}{3.7592} + 1 \end{aligned}. \quad (12)$$

Its circuit structure is shown in Fig. 2(a), where input coefficients y_0, y_1, y_2, z_0, and z_1 are determined using the method described in Section II. The 'Shift$_5$' module is to circularly shift the outputs of RNS by 5 bits to maximally decorrelate for MUXs [23] because the adopted RNS is a 10-bit LFSR.

B. Hyperbolic Tangent Function

$$tanh(x) = x - \frac{x^3}{3} = x\left(\left(1 - x^2\right) + \frac{2}{3}x^2\right) = x \cdot p(x). \quad (13)$$

In (13), $p(x)$ represents the second-order factor without a linear term. Following the proposed design method, a MUX combined with an AND gate is selected for its implementation. The corresponding circuit structure in SC is shown in Fig. 2(b). The input coefficients y_0 and y_1 in Fig. 2(b) can be computed using the method described in Section II.

C. Sine Function

$$\begin{aligned} sin(x) &= x - \frac{x^3}{6} = \left(-\frac{1}{6}\right)x(x + 2.4495)(2.4495 - x) \\ &= \frac{(3.4495 \times 2.4495)}{6} p(x) \end{aligned}, \quad (14)$$

where the 3rd-order polynomial is also represented as the multiplication of two factors, as

$$\begin{aligned} p(x) &= x \cdot p1(x) \cdot p2(x), \\ p1(x) &= \frac{x}{3.4495} + \frac{2.4495}{3.4495},\ p2(x) = 1 - \frac{x}{2.4599} \end{aligned}. \quad (15)$$

The circuit structure for *sin*(*x*) is shown in Fig. 2(c), where coefficients are computed in the same way.

IV. EXPERIMENTAL RESULTS

The proposed circuits and two previous designs [19, 20] are programmed in Verilog hardware description language (HDL) because they are more efficient than other works in terms of computing accuracy and hardware overhead. Functional simulation is performed using Modelsim, and synthesis is carried out with Synopsys Design Compiler with 65nm process technology at an operating frequency of 100 MHz. A 10-bit LFSR is employed in an SNG to convert BNs into SNs, within the range of [0, 1]. The computing accuracy for nonlinear functions is evaluated through simulations in MATLAB. The sampling step is 0.1 within [0, 1]. Since the seeds in an LFSR are randomly generated, 500 simulation runs are performed, and the average of these results is taken for each input value of *x*.

A. Computing Accuracy

The mean absolute error (MAE) and MSE values are adopted for assessing the computing accuracy of these circuits, as listed in TABLE I. Clearly, compared with the correlation-based NAND-AND [19] and weighted adder structures [20], the proposed designs achieve significant reductions in both MAE and MSE values. For example, for $sin(x)$, the MAE value of the proposed design is lowered by 49.9% and 44.5% compared with the other two designs. Similarly, the MSE value is reduced by 69.3% and 64.2%, respectively.

B. Hardware Performance

TABLE II lists the hardware costs of three nonlinear functions, including area, critical path delay (CPD), power consumption, along with area-delay product (ADP) and power-delay product (PDP). Note that PDP is computed using area multiplied by CPD.

For three nonlinear functions, the area and delay are lowered, which is attributed to the circuit structure. The proposed MUX-based circuit employs a parallel processing structure for multiple factors and shared RNS, resulting in a shorter critical path and smaller area than cascaded gate structures, yielding lowered PDP and ADP values. For example, the power consumption is reduced by 48.00% and 1.24%, respectively, compared to the other two designs for $sin(x)$.

V. Conclusion

In this work, circuits in SC for computing nonlinear functions are proposed based on MUXs. For this end, a nonlinear function is decomposed into the multiplication of multiple first- and second-order factors. For implementing these factors, 22 circuit structures are designed, simulated, and compared in computing accuracy. Two structures, respectively for first- and second-order factors, are selected for further usage. Experimental results demonstrate that the proposed circuits outperform previous works in computing accuracy and hardware costs. In detail, for the 3rd-order Maclaurin series of the three considered nonlinear functions, the proposed designs achieve 38.62% and 28.82% reductions on average in MSE and PDP, respectively, compared to previous works. The PDP differences between the proposed designs and the weighted adder based ones are small for simple functions due to their low complexity and short critical paths, which limit performance gains. In contrast, the proposed structure with third-order Maclaurin expansion offers advantages in delay and error performance for more complex functions.

References

[1] Y. Zhu, Y. Zhang, K. Han, and J. Hu, "A nonlinear function logic computing architecture with low switching activity," *IEEE Trans. Circuits Syst., II, Exp. Briefs,* vol. 70, no. 9, pp. 3649-3653, May. 2023.

[2] S. Salehi, "Low-cost stochastic number generators for stochastic computing," *IEEE Trans. Very Large Scale Integr. VLSI Syst.,* vol. 28, no. 4, pp. 992-1001, Apr. 2020.

[3] B. Brown, and H. Card, "Stochastic neural computation I: Computational elements," *IEEE Trans. Comput.,* vol. 50, no. 9, pp. 891-905, Sep. 2001.

[4] P. Li, D. Lilja, W. Qian, M. Riedel, and K. Bazargan, "Logical computation on stochastic bit streams with linear finite-state machines," *IEEE Trans. Comput.,* vol. 63, no. 6, pp. 1473-1485, Jun. 2014.

[5] P. Li, D. Lilja, W. Qian, K. Bazargan, and M. Riedel, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," in 2012 IEEE/ACM International Conference on Computer-Aided Design, New York, 2012, pp. 480-487.

[6] X. Feng, K. Hu, and K. Han, "A high-efficiency general nonlinear function generator for stochastic computation," in 2021 IEEE 3rd International Conference on Circuits and Systems (ICCS), Chengdu, China, 2021, pp. 150-155.

[7] M. Najafi, P. Li, D. Lilja, W. Qian, K. Bazargan, and M. Riedel, "A reconfigurable architecture with sequential logic-based stochastic computing," *ACM J. Emerg. Technol. Comput. Syst.,* vol. 13, no. 4, pp. 1-28, Aug. 2017.

[8] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. Gross, "VLSI implementation of deep neural network using integral stochastic computing," *IEEE Trans. Very Large Scale Integr. VLSI Syst.,* vol. 25, no. 10, pp. 2688-2699, Oct. 2017.

[9] S. Chu, C. Hsieh, and Y. Huang, "Design of FSM-based function with reduced number of states in integral stochastic computing," *IEEE Trans. Very Large Scale Integr. VLSI Syst.,* vol. 27, no. 6, pp. 1475-1479, Jun. 2019.

[10] W. Qian, X. Li, M. Riedel, K. Bazargan, and D. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Trans. Comput.,* vol. 60, no. 1, pp. 93-105, Jan. 2011.

[11] X. Li, W. Qian, M. Riedel, K. Bazargan, and D. Lilja, "A reconfigurable stochastic architecture for highly reliable computing," in Proceedings of the 19th ACM Great Lakes Symposium on VLSI, Boston Area, MA, USA, 2009, pp. 315-320.

[12] W. Qian, and M. Riedel, "The synthesis of robust polynomial arithmetic with stochastic logic," in 45th ACM/IEEE Design Automation Conference, Anaheim, CA, USA, 2008, pp. 648-653.

[13] P. Li, W. Qian, and D. Lilja, "A stochastic reconfigurable architecture for fault-tolerant computation with sequential logic," in IEEE 30th International Conference on Computer Design (ICCD), Montreal, QC, Canada, 2012, pp. 303-308.

[14] K. Parhi, and Y. Liu, "Computing arithmetic functions using stochastic logic by series expansion," *IEEE Trans. Emerging Top. Comput.,* vol. 7, no. 1, pp. 44-59, Mar. 2019.

[15] K. Parhi, "Stochastic logic implementations of polynomials with all positive coefficients by expansion methods," *IEEE Trans. Circuits Syst., II, Exp. Briefs,* vol. 65, no. 11, pp. 1698-1702, Nov. 2018.

[16] Y. Liu, and K. Parhi, "Computing complex functions using factorization in unipolar stochastic logic," in International Great Lakes Symposium on VLSI, Boston, MA, USA, 2016, pp. 109-112.

[17] Y. Liu, and K. Parhi, "Computing hyperbolic tangent and sigmoid functions using stochastic logic," in 50th Asilomar Conference on Signals, Systems and Computers, Los Alamitos, USA, 2016, pp. 1580-1585.

[18] Y. Liu, and K. Parhi, "Computing polynomials using unipolar stochastic logic," *ACM J. Emerg. Technol. Comput. Syst.,* vol. 13, no. 3, pp. 1-30, May. 2017.

[19] S. Chu, C. Wu, T. Nguyen, and B. Liu, "Polynomial computation using unipolar stochastic logic and correlation technique," *IEEE Trans. Comput.,* vol. 71, no. 6, pp. 1358-1373, May. 2021.

[20] S. Wang, G. Xie, X. Cheng, and Y. Zhang, "Weighted-adder-based polynomial computation using correlated unipolar stochastic bitstreams," *IEEE Trans. Circuits Syst., II, Exp. Briefs,* vol. 69, no. 11, pp. 4528-4532, Nov. 2022.

[21] A. Alaghi, and J. Hayes, "Exploiting correlation in stochastic circuit design," in IEEE 31st International Conference on Computer Design (ICCD), Asheville, NC, USA, 2013, pp. 39-46.

[22] P. Ting, and J. Hayes, "Isolation-based decorrelation of stochastic circuits," in IEEE 34th International Conference on Computer Design (ICCD), Scottsdale, AZ, USA, 2016, pp. 88-95.

[23] H. Ichihara, S. Ishii, D. Sunamori, T. Iwagaki, and T. Inoue, "Compact and accurate stochastic circuits with shared random number sources," in IEEE 32nd International Conference on Computer Design (ICCD), Seoul, South Korea, 2014, pp. 361-366.