An SRAM-based Stochastic Number Generator for Stochastic Computing

Heng Shi*, Zhengkun Yu*, Tingting Zhang†, Jie Han†, Yumeng Yang*, Siting Liu*
*School of Information Science and Technology, ShanghaiTech University, Shanghai, China
†Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada

***Abstract*—Stochastic computing (SC) features a unique number representation, where real values are encoded by the probability of "1"s in a random binary bit stream or a stochastic sequence. It enables hardware-efficient arithmetic circuit designs with simple logic gates. However, stochastic number generators (SNGs) are required to produce stochastic sequences. The high hardware cost of an SNG offsets the advantage of SC. To reduce the hardware cost of an SNG, we propose an SRAM-based SNG using voltage under-scaling. It generates random bits by leveraging the access instability of selected SRAM cells, induced by a reduced supply voltage. It is suitable for energy-efficient SC. We implemented the SRAM-based SNG on a Xilinx ZC702 FPGA using block RAMs and evaluated its performance across multiple SC applications, including finite-state machine-based tanh function generation and an Ising machine that solves max-cut problems (MCPs). For the tanh function, our design achieves a comparable mean-squared error (MSE) (8.7×10^{-3}) compared to the use of traditional SNGs, such as Sobol- (4.67×10^{-2}) and linear feedback shift register (LFSR)-based (1.53×10^{-2}) ones. For MCPs, a maximum cut value comparable to that of a cutting-edge design is achieved. Compared with LFSR- and Sobol-based designs, the proposed design consumes 84.3% and 92.5% less energy, respectively.**

***Index Terms*—stochastic computing (SC), random number generator (RNG), voltage under-scaling, low-power design.**

I. Introduction

Stochastic computing (SC) systems have re-emerged as a low-power design alternative to conventional binary computing systems [1]. Unlike conventional computing, SC performs computation on stochastic bit streams or sequences. For example, "11001010" contains four 1's out of eight bits, so it represents the value of 4/8. A prominent feature of SC is its low hardware complexity in implementing arithmetic circuits. Fig. 1 shows that an AND gate implements multiplication in SC. The two input stochastic sequences encode 0.25 and 0.5, respectively. A bitwise AND of the two sequences yields "00000010". The probability of the resulting stochastic sequence then approximates the product of the probabilities encoded in the two input sequences when they are independent. Due to the low hardware cost, SC has been employed in low-power compute-intensive tasks such as image processing [2], neural networks [3], [4], [26], [30], [31], [33], [34], as well as complex computations such as function generation [14]–[16], [21], [22], [25]. However, a stochastic

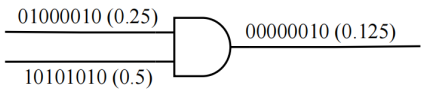


Fig. 1. An AND gate implements SC multiplication.

number generator (SNG) is necessary to convert the input data into stochastic sequences in an SC circuit. Fig. 2 illustrates an SNG, consisting of a random number generator (RNG) and a comparator. During each clock cycle, a random number (RN) is compared with the binary number. The SNG produces a 1 when the input value is lager than the RN, otherwise 0.

A pseudo-random number generator based on a linear feedback shift register (LFSR) is commonly used as the RNG in an SNG due to its simplicity and reliable random behavior [6]. However, to ensure a high accuracy, uncorrelated stochastic sequences, and thus multiple independent SNGs, are often required [7]. This results in SNGs consuming the majority of the circuit area [8]. Low-discrepancy (LD) or quasi-random

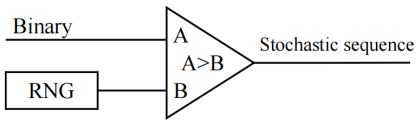


Fig. 2. A stochastic number generator (SNG).

sequences, such as Sobol sequence generators [5], have been shown to be effective for RNGs; they offer better accuracy and higher energy efficiency due to their regular generation pattern and even distribution. However, in order to implement a Sobol sequence generator, a large hardware cost is required, which in turn negates the advantages of SC. In recent years, SRAM-based true random number generators have become interesting to investigate [9], [10]. Compared to the former RNG designs, SRAM-based RNGs do not require dedicated circuits to generate RNs, and thus provide high throughput with low energy [13].

In this work, we propose an SRAM-based SNG for stochastic sequence generation. It is applied to synthesize functions and implement SC Ising machines. Experimental results on an FPGA show that the proposed SRAM-based SNG pro-

This work was financially supported in part by National Natural Science Foundation of China (No. 62204155), and in part by Shanghai "Sailing" Program (No. 22YF1428300). The work at the University of Alberta was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Numbers: RES0048688, RES0051374 and RES0054326) and Alberta Innovates (Project Number: RES0053965). Siting Liu and Yumeng Yang are the corresponding authors.

duces high accuracy in approximating a tanh function and implementing Ising machines. The proposed design consumes 84.3% and 92.5% less energy compared with LFSR- and Sobol-based designs, respectively. The main contributions of this paper are as follows.

- An SRAM-based SNG is devised by employing the access failure of SRAM bit cells when the voltage is under-scaled.
- Stochastic sequences are generated by the proposed design on an FPGA. They are used in tanh function synthesis and solving a max-cut problem with high accuracy and reduced energy consumption.

II. Background

A. SRAM-based RNGs

Previous SRAM-based RNGs have leveraged different entropy sources to generate random numbers, such as physical fingerprinting [11] and intrinsic bit instability [10]. Holcomb et al. [11] found that power-up of SRAM produces randomness. Chen et al. [13] proposed an SRAM-based method relying on access failure. Access failures occur when the supply voltage of the SRAM cell is decreased below the manufacturer's recommended value. While the data are not destroyed during voltage under-scaling reads, the sense amplifier cannot distinguish the stored data and can behave randomly. Ismail et al. have confirmed that access failures are a key factor in producing randomness [12].

B. Applications of stochastic computing

1) Function synthesis using SC-FSMs: Unlike conventional function units, SC can realize special functions using a finite-state machine (FSM). For example, the FSM in Fig. 3 implements a tanh function for neural networks [14]. The N-state

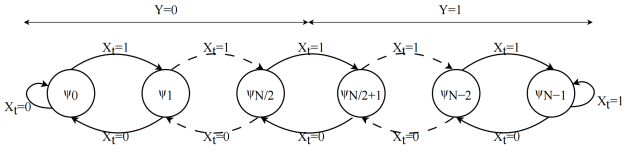


Fig. 3. An FSM with N states that generates the tanh function. X_t denotes the t^{th} input of stochastic sequence X [14].

FSM takes one bit in the stochastic sequence at each clock cycle. If the bit is 1, the FSM moves to the state on the right or stays if it is at the rightmost state; otherwise, the FSM moves to the state on the left or stays if it is at the leftmost state. The FSM produces an output bit 1 if it is at one of the states on the right, and produces a 0 if it is at one of the states on the left. As per the theory of Markov chain, the probability of the output stochastic sequence is obtained approximately as $\tanh(\frac{N}{2}x)$ [14], where x is the value encoded by the input stochastic sequence. To satisfy the working conditions of a Markov chain model, a true RNG can be used to generate the stochastic sequence encoding x. The FSM itself can be implemented by a saturated counter.

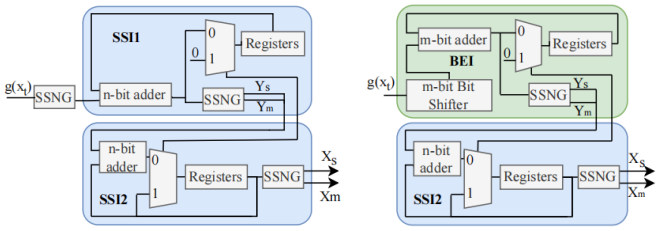
Ardakani et al. proposed a weighted FSM [16]. Its states are associated with real-valued output weights instead of just 0s and 1s, while the FSM state transition rules remain unchanged. The weights are then obtained by a regression. The weighted FSM achieves superior accuracy compared with the previous SC designs. To ensure a high accuracy, we employ this regression-based method to perform the generation of a tanh function.

2) Max-cut problems with the Ising model: An Ising model is an emerging method for solving combinatorial optimization problems (COPs). It can be implemented by either physical models such as oscillators [23] or heuristic algorithms such as simulated bifurcation [24]. Heuristic algorithms are able to achieve higher precision and scalability, yet they usually require complex hardware to implement. To mitigate this disadvantage, Zhang et al. [17] proposed SC-based and binary-SC-mixed simulated bifurcation cell (SC-SBC and BSC-SBC) circuits. Fig. 4 shows the two implementations of one Ising spin cell in an Ising machine. They can implement the simulated bifurcation model described by the following partial differential equations (PDEs),

$$\dot{x}_{i,t} = f\left(\boldsymbol{y_t}\right)_i = a_0 y_{i,t}, \tag{1}$$

$$\dot{y}_{i,t} = g\left(\boldsymbol{x_t}\right)_i = -\left(a_0 - a(t)\right) x_{i,t} + c_0 \sum_{j=1}^{N} J_{ij} x_{j,t}. \tag{2}$$

By encoding the signals $g(x_i)$, $y_{i,t}$ and $x_{i,t}$ as stochastic sequences, (1) and (2) can be solved by either the signed stochastic integrators (SSIs) or the binary Euler integrator (BEI) for each cell. $\boldsymbol{x_t}$ and $\boldsymbol{y_t}$ denote the position and momentum vectors of the spin cells at time t, respectively; a_0 and c_0 are constants; $a(t)$ is a linear function. $\{J_{ij}\}$ is a matrix describing the connection strength between spin i and j. The optimization results or the positions of the spin cells are provided by the registers in the second stage of the SSI in the two designs. Experimental results demonstrate that randomness facilitates the algorithm escaping local minima and reaching global minima. This can be achieved by using true RNGs in the signed SNGs (SSNGs). Note that dynamic stochastic computing [35] is employed in this design, the sequence length can be as low as 1 bit for solving the PDEs.



(a) SC-SBC (b) BSC-SBC

Fig. 4. Circuit design of the simulated bifurcation cells [17]. (a) stochastic computing simulated bifurcation cell (SC-SBC); (b) binary-stochastic computing simulated bifurcation cell (BSC-SBC). Since signed SC is used, the subscript s represents the sign bit, and m denotes the stochastic bit encoding the magnitude of the corresponding signal.

III. Proposed Design

We propose an SRAM-based SNG using voltage under-scaling as shown in Fig. 5. The SRAM block is first profiled to identify suitable SRAM cells for random bit generation with a reduced supply voltage each time when it is powered on. Each row is read 1,000 times, and the entropy of each bit is calculated. Shannon Entropy is used to evaluate the randomness of each bit, given as

$$H = -\sum_{i=0}^{1} p_i \log_2 p_i, \tag{3}$$

where p_0 represents the probability of reading a 0 from the SRAM cell, and p_1 the probability of reading a 1 in 1,000 read trials. A higher value indicates a larger randomness. The SRAM bit(s) with the highest Shannon entropy are then recorded by their addresses and bit positions in the n-bit data. The probabilities of these bits are around 0.5 and they are then selected as the entropy sources.

During the operation of the SC system, each entropy source is accessed through its recorded address and bit position using an SRAM read and a multiplexer. The random bit is then loaded into the w-bit shift register one by one. The w-bit number in the shift register then approximates a uniformly distributed random number within [0,1] (normalized by 2^w). It is then compared with the binary fractional number to be encoded as in a conventional SNG.

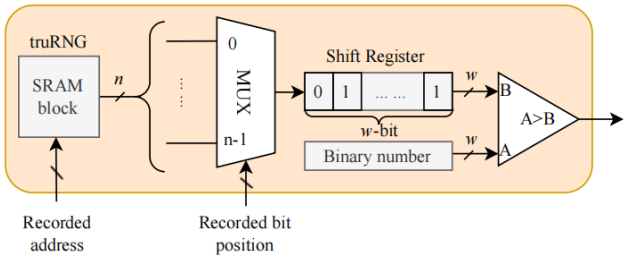


Fig. 5. Proposed SRAM-based SNG.

IV. Experiments and Results

The experiments are conducted on a Xilinx Zynq ZC702 FPGA board [19]. Fig. 6 depicts the setup of the experiments. This board provides an independent voltage rail called VCCBRAM. It can be used to control the supply voltage of the SRAM blocks or block RAM (BRAM) in the FPGA. To control the BRAM supply voltage, the TI USB interface adapter is connected to the voltage connection port on the back of the board via PMBus [32] through a PC. To profile the BRAM, the TI Fusion digital power designer on the PC is used to send commands to the adapter to reduce the VCCBRAM from nominal 1 V to 560 mV (the minimum voltage for the FPGA to function properly) on a PC. A 512-KB BRAM is used for the profiling. After the profiling, the processing system (PS) on the FPGA is used to access the SRAM cell with the largest entropy. The generated random bits are then used for the two aforementioned SC applications.

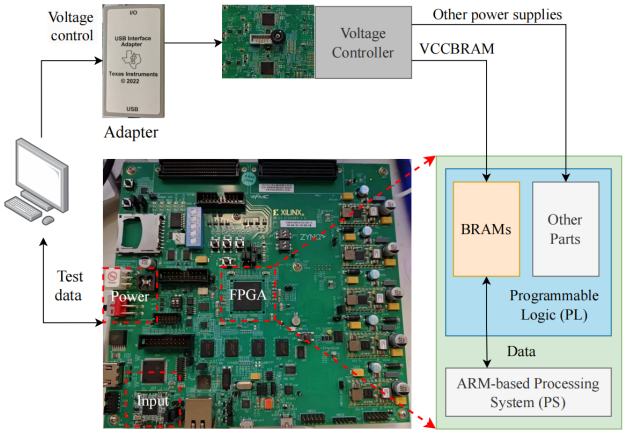


Fig. 6. Overall experiment setup on a ZC702 FPGA board.

A. Function generation using FSMs

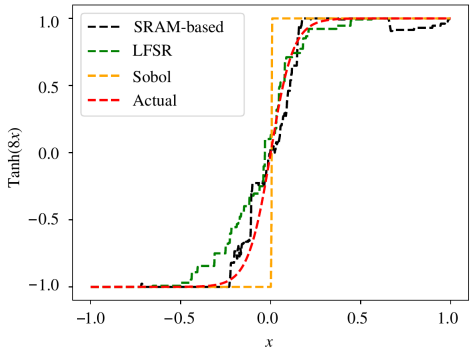


Fig. 7. Tanh$(8x)$ function generation using LFSR-based, Sobol-based and the proposed SRAM-based 8-bit SNGs.

Fig. 7 shows the simulation results using different RNGs to synthesize the function of tanh$(8x)$ with a sequence length of 256 bits. As shown in Fig. 7, Sobol-based tanh$(8x)$ implementation produces a hard threshold function and exhibits the largest error among the considered designs [36]. The MSE is about 4.67×10^{-2}. The designs using LFSR- and the proposed SRAM-based RNG leads to similar MSEs of 1.53×10^{-2} and 1.35×10^{-2}, respectively. The other functions, such as a 2-dimensional Gaussian distribution function and a sinusoidal function, are also tested. They show similar trends: the design using Sobol-based RNGs produces the largest MSE, while the proposed SRAM-based designs slightly outperform the LFSR-based designs. The results are summarized in Table I.

TABLE I
THE MSEs OF THE SC-BASED FUNCTION GENERATORS USING DIFFERENT RNGs WITH A SEQUENCE LENGTH OF 256 BITS.

RNG	tanh$(8x)$	2D Gaussian	cos $\in [-\pi, \pi]$
This work	$\mathbf{8.7 \times 10^{-3}}$	$\mathbf{9.38 \times 10^{-4}}$	$\mathbf{1.84 \times 10^{-2}}$
LFSR [6]	1.53×10^{-2}	1.03×10^{-3}	2.0×10^{-2}
Sobol [5]	4.67×10^{-2}	6.33×10^{-3}	6.47×10^{-2}

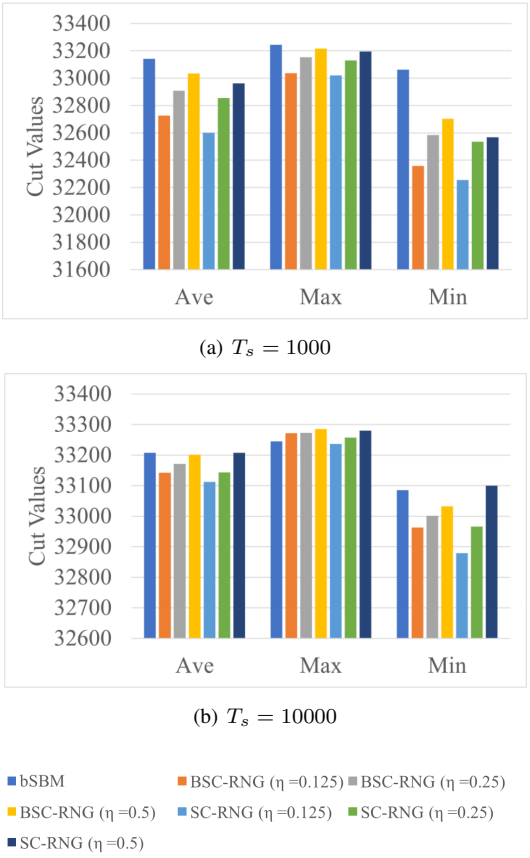


Fig. 8. Average (Avg), maximum (Max) and minimum (Min) cut values found by various simulated bifurcation implementations. (a) Experimental results when the number of steps is $T_s = 1000$; (b) experimental results with $T_s = 10000$.

B. Max-cut problems with Ising model

The SC-based Ising machine is used to solve a max-cut problem (MCP), which is the K_{2000} benchmark [20] with 2000 fully connected spins. The goal is to partition the spins into two sets in such a way that the weighted connections between these sets are maximized. While the maximum cut value found for this problem is 33,337, the solutions produced by the SC-based Ising machines are close to this value, as shown in Fig. 8. The average, maximum and minimum of the cut values are obtained through 100 trials, marked as "Avg", "Max" and "Min". "SC-RNG" and "BSC-RNG" denote the stochastic designs using SRAM-based SNGs in Fig. 4(a) and (b), respectively. The results using SRAM-based SNGs are compared to a baseline implementation using conventional binary arithmetic, referred to as "bSBM" in Fig. 8. Figs. 8(a) and (b) show the results when the simulated bifurcation is run for T_s=1,000 and $T_s = 10,000$ steps, respectively. Three different values 0.125, 0.25, and 0.5 are considered for manually tunable step sizes (η, not shown in (1) and (2)) when solving the PDEs of (1) and (2) using the Euler method.

When T_s is 1,000, "bSBM" attains larger max cut values because the solution converges in fewer steps due to the accurate multi-bit computations. When T_s increases to 10,000, "SC-RNG" and "BSC-RNG" generally exhibit an increase in the max cut values, even exceeding those of the "bSBM" designs. This may be due to the true randomness of the SRAM-based SNGs, and its randomness assists the algorithm escaping from the local minima in a long run. Due to the hybrid use of SC and conventional binary computing, BSC designs outperform the pure SC circuits in a short run when T_s is 1,000, while they perform similarly in a long run ($T_S = 10,000$) due to the same reason. It can also be seen from the results that the proposed designs using SRAM-based SNGs obtain larger cut values when the step size η is larger.

C. Hardware efficiency evaluation

Since the comparator and the register that stores the number to be encoded are the common components of these SNGs, they are excluded from the evaluation of hardware efficiency, i.e. only the RNGs are evaluated. The SRAM-based RNG is tested at a supply voltage of 560 mV and a working frequency of 200 MHz. The power is monitored over PMbus and the energy generating one RN is calculated. Since it requires no additional logic gates or registers, the utilization of LUTs and FFs is 0. Moreover, only the SRAM cell with the highest entropy is used for random number generation; therefore, one BRAM block that contains the SRAM cell is sufficient in our work. Conventional RNGs based on Sobol and LFSR are also tested at the same working frequency but with a nominal supply voltage of 1 V. The power is estimated by the Xilinx Vivado Design Suite on the same FPGA after implementation, and the energy that generates one RN is also calculated. Table II shows the utilization and energy consumption of different RNGs. For an 8-bit RNG design, the SRAM-based one consumes about 0.11 nJ per RN. In contrast, an 8-bit LFSR and Sobol sequence generator require 0.70 nJ and 1.47 nJ, respectively. It indicates an energy reduction of 84.3% and 92.5%, respectively.

TABLE II
UTILIZATION AND ENERGY EFFICIENCY OF THE PROPOSED SRAM-BASED RNG GENERATING 1 RN VS. THE OTHER RNGS.

RNG	# LUTs	# FFs	# BRAM	Energy
This work	0	0	1	**0.11 nJ**
LFSR [6]	5	8	0	0.70 nJ
Sobol [5]	20	19	0	1.47 nJ

V. CONCLUSION

In this paper, we propose an SRAM-based SNG that leverages voltage under-scaling for SC. The design aims to enhance the energy efficiency of SC without compromising accuracy. The performance is evaluated on two SC applications: function generation using FSMs and solving MCPs with the Ising model. Compared with traditional SNGs, such as Sobol- and LFSR-based ones, the SRAM-based SNG achieves similar accuracy while significantly reducing the energy consumption. These findings suggest that the SRAM-based SNG could be a potential solution for energy-efficient SC systems.

REFERENCES

[1] B. R. Gaines, Stochastic Computing Systems, 1969, pp. 37–172.

[2] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 3, pp. 449–462, Mar. 2014.

[3] C. Nie et al., "VSPIM: SRAM Processing-in-Memory DNN Acceleration via Vector-Scalar Operations," in IEEE Transactions on Computers, vol. 73, no. 10, pp. 2378-2390, Oct. 2024.

[4] Y. Hu et al., "A 28-nm 198.9-TOPS/W Fault-Tolerant Stochastic Computing Neural Network Processor," in IEEE Solid-State Circuits Letters, vol. 5, pp. 198-201, 2022.

[5] S. Liu and J. Han, "Energy efficient stochastic computing with Sobol sequences," Design, Automation Test in Europe Conference Exhibition (DATE), 2017, Lausanne, Switzerland, 2017, pp. 650-653.

[6] S. A. Salehi, "Low-Cost Stochastic Number Generators for Stochastic Computing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 4, pp. 992-1001, April 2020.

[7] A. Alaghi and J. P. Hayes, "Exploiting correlation in stochastic circuit design," in Proc. IEEE 31st Int. Conf. Comput. Design (ICCD), Oct. 2013, pp. 39–46.

[8] S. Mohajer, Z. Wang, K. Bazargan, M. Riedel, D. Lilja, and S. Faraji, "Parallel computing using stochastic circuits and deterministic shuffling networks," U.S. Patent Appl. 16/165 713, Apr. 25, 2019.

[9] D. E. Holcomb, W. P. Burleson, K. Fu et al., "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in Proceedings of the Conference on RFID Security, vol. 7, no. 2, 2007, p. 01.

[10] L. T. Clark, S. B. Medapuram, and D. K. Kadiyala, "SRAM circuits for true random number generation using intrinsic bit instability," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 10, pp. 2027–2037, 2018.

[11] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," IEEE Transactions on Computers, vol. 58, no. 9, pp. 1198–1210, 2008.

[12] Yüksel, İ. E., "TuRaN: True Random Number Generation Using Supply Voltage Underscaling in SRAMs", arXiv e-prints, Art. no. arXiv:2211.10894, 2022. doi:10.48550/arXiv.2211.10894.

[13] Q. Chen, H. Mahmoodi, S. Bhunia, and K. Roy, "Modeling and testing of SRAM for new failure mechanisms due to process variations in nanoscale cmos," in 23rd IEEE VLSI Test Symposium (VTS'05). IEEE, 2005, pp. 292–297.

[14] B. D. Brown and H. C. Card, "Stochastic neural computation. I. Computational elements," in IEEE Transactions on Computers, vol. 50, no. 9, pp. 891-905, Sept. 2001.

[15] P. Li, D. J. Lilja, W. Qian, K. Bazargan and M. Riedel, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 2012, pp. 480-487.

[16] A. Ardakani, A. Ardakani and W. J. Gross, "A Regression-Based Method to Synthesize Complex Arithmetic Computations on Stochastic Streams," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 2020.

[17] T. Zhang et al.,"A High-Performance Stochastic Simulated Bifurcation Ising Machine," DAC '24, June 23–27, 2024, San Francisco, CA, USA.

[18] Su et al., "A 252 spins scalable CMOS Ising chip featuring sparse and reconfigurable spin interconnects for combinatorial optimization problems," in CICC. IEEE, 2021, pp. 1–2.

[19] Xilinx, "Xilinx Zynq ZC702 FPGA Board," https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html.

[20] K. Yamamoto et al., "STATICA: A 512-spin 0.25 M-weight annealing processor with an all-spin-updates-at-once architecture for combinatorial optimization with complete spin-spin interactions," IEEE JSSC, vol. 56, no. 1, pp. 165–178, 2021.

[21] N. Saraf, K. Bazargan, D. J. Lilja and M. D. Riedel, "Stochastic functions using sequential logic," 2013 IEEE 31st International Conference on Computer Design (ICCD), Asheville, NC, USA, 2013, pp. 507-510.

[22] P. Li, D. J. Lilja, W. Qian, M. D. Riedel and K. Bazargan, "Logical Computation on Stochastic Bit Streams with Linear Finite-State Machines," in IEEE Transactions on Computers, vol. 63, no. 6, pp. 1474-1486, June 2014.

[23] T. Wang and J. Roychowdhury, "OIM: Oscillator-based Ising machines for solving combinatorial optimisation problems," in UCNC. Springer, 2019, pp. 232–256.

[24] H. Goto et al., "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," Sci. Adv., vol. 5, no. 4, p. eaav2372, 2019.

[25] W. Qian, X. Li, M. D. Riedel, K. Bazargan and D. J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic," in IEEE Transactions on Computers, vol. 60, no. 1, pp. 93-105, Jan. 2011.

[26] S. R. Faraji, P. Abillama, G. Singh and K. Bazargan, "HBUCNNA: Hybrid Binary-Unary Convolutional Neural Network Accelerator," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 2020.

[27] X. Zhang, C. Jiang, G. Dai, L. Zhong, W. Fang, K. Gu, G. Xiao, S. Ren, X. Liu, and S. Zou, "Improved performance of SRAM-based true random number generator by leveraging irradiation exposure," Sensors, vol. 20, no. 21, p. 6132, 2020.

[28] H. Aziza et al., "True Random Number Generator Integration in a Resistive RAM Memory Array Using Input Current Limitation," in IEEE Transactions on Nanotechnology, vol. 19, pp. 214-222, 2020.

[29] F. Pareschi, G. Setti and R. Rovatti, "Implementation and Testing of High-Speed CMOS True Random Number Generators Based on Chaotic Systems," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 12, pp. 3124-3137, Dec. 2010.

[30] Y. Hu et al., "A 28-nm 198.9-TOPS/W Fault-Tolerant Stochastic Computing Neural Network Processor," in IEEE Solid-State Circuits Letters, vol. 5, pp. 198-201, 2022.

[31] S. Aygun, L. Kouhalvandi, M. H. Najafi, S. Ozoguz and E. O. Gunes, "Hardware–Software Co-Optimization of Long-Latency Stochastic Computing," in IEEE Embedded Systems Letters, vol. 15, no. 4, pp. 190-193, Dec. 2023.

[32] PMBus, "PMBus Specification," http://pmbus.org/.

[33] S. Aygun and M. H. Najafi, "Sobol Sequence Optimization for Hardware-Efficient Vector Symbolic Architectures," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2024.

[34] S. Aygun, M. S. Moghadam and M. H. Najafi, "uHD: Unary Processing for Lightweight and Dynamic Hyperdimensional Computing," 2024 Design, Automation Test in Europe Conference Exhibition (DATE), Valencia, Spain, 2024.

[35] S. Liu, W. J. Gross and J. Han, "Introduction to Dynamic Stochastic Computing," in IEEE Circuits and Systems Magazine, vol. 20, no. 3, pp. 19-33, thirdquarter 2020.

[36] S. Liu and J. Han, "Toward Energy-Efficient Stochastic Circuits Using Parallel Sobol Sequences," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 7, pp. 1326-1339, July 2018.