Hardware–Software Co-Optimization of Long-Latency Stochastic Computing

Sercan Aygun, *Member, IEEE*, Lida Kouhalvandi, *Senior Member, IEEE*, M. Hassan Najafi, *Member, IEEE*, Serdar Ozoguz, and Ece Olcay Gunes

***Abstract*—Stochastic computing (SC) is an emerging paradigm that offers hardware-efficient solutions for developing low-cost and noise-robust architectures. In SC, deterministic logic systems are employed along with bit-stream sources to process scalar values. However, using long bit-streams introduces challenges, such as increased latency and significant energy consumption. To address these issues, we present an optimization-oriented approach for modeling and sizing new logic gates, which results in optimal latency. The optimization process is automated using hardware–software cooperation by integrating Cadence and MATLAB environments. Initially, we optimize the circuit topology by leveraging the design parameters of two-input basic logic gates. This optimization is performed using a multiobjective approach based on a deep neural network. Subsequently, we employ the proposed gates to demonstrate favorable solutions targeting SC-based operations.**

***Index Terms*—Analog optimization, co-processing, latency reduction, stochastic computing (SC).**

I. Introduction

STOCHASTIC computing (SC) is a reemerging computation paradigm experiencing a resurgence due to its ability to reduce area and power consumption. In SC, deterministic logic gates are driven by random pulses, taking into account the probability values of each input. The resulting output is obtained as another pulse train after processing the input pulses. Previous research has demonstrated how each digital logic gates perform a specific function based on input probabilities [1]. Specifically, multiplication is achieved using AND gates, while addition is achieved using multiplexers (MUXs).

The random pulses serve as inputs, carrying binary information with N logic values. The correlation between input pulses plays a crucial role in SC operations. Logic gates exhibit different behaviors based on correlation level. At the midpoint of positive and negative correlation, where there is no correlation, logic gates correspond to well-known arithmetic operations such as the AND multiplier [1], [2]. Increasing N or the bit-stream length improves the accuracy. In theory, deterministic results equivalent to traditional binary arithmetic operations can be achieved as N approaches infinity. However, larger values of N introduce significant latency. Various solutions have been proposed to address this latency issue, including area–delay optimization [3], deterministic shuffling [4], variable-latency approaches [5], and D flip-flop insertion methods [6]. Instead of focusing on higher-level solutions, this work addresses low-level design challenges. We employ multiobjective optimization methods at the transistor level to model and size new logic gates to minimize latency and power consumption. We focus on circuit-level design optimization and address the latency problem through stream processing techniques. Our approach aims to leverage low-level optimized design primitives.

Recently, there has been a growing interest in employing deep neural networks (DNNs) as an optimization framework due to their ability to deliver accurate outcomes [7]. In this study, we employ DNNs to make the best decisions regarding transistor geometry. We draw inspiration from the work by Kouhalvandi et al. [7], who successfully handle data regression for gallium nitride (GaN)-based transistors to predict optimal design parameters, considering nonlinear multiobjective design specifications. We employ multiobjective optimization methods, such as long short-term memory (LSTM) and Thompson sampling efficient multiobjective optimization (TSEMO) algorithm, due to their proven effectiveness [7]. Then, we train the neural network to generate regression points corresponding to the most suitable combinations of width (W) and length (L) to achieve desired latency and power consumption levels. The TSEMO algorithm [8] functions as a multiobjective optimization approach, evaluating optimal values based on various criteria. Identifying the Pareto-optimal front (POF) is crucial to establish reference points for the DNN. We optimize the latency × power value in the neural network output by leveraging transistor geometries at the network input. Unlike [7], where power amplifier parameters are used, we employ transistor geometries as inputs. We utilize a shorter network consisting of two LSTM layers. The LSTM structure is chosen to modify transistor geometries individually in a 1-D time-like sequence and feed them to the neural network. By feeding the network with consecutive geometries, we evaluate the optimal output for latency × power within the DNN. In contrast to [7], our output layer does not involve parameters related to a power amplifier model but rather incorporates information associated with latency and stochastic computation linked to transistor power. Moreover, considering multiple optimization criteria, we determine Pareto points within a multiobjective optimization framework and train the DNN accordingly. We explore 10^6 data points, covering various transistor geometry possibilities, to determine the optimal geometry within this design space. We introduce a novel two-input basic logic gate library tailored for SC,

Manuscript received 13 July 2023; accepted 22 July 2023. Date of publication 25 September 2023; date of current version 28 November 2023. This work was supported in part by the National Science Foundation (NSF) under Grant 2019511; by the Louisiana Board of Regents Support Fund under Grant LEQSF(2020-23)-RD-A-26; and by the generous gifts from Cisco, Xilinx, Nvidia, and Cadence. This manuscript was recommended for publication by P. R. Panda. *(Corresponding author: Sercan Aygun.)*

Sercan Aygun and M. Hassan Najafi are with the School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70503 USA (e-mail: sercan.aygun@louisiana.edu; najafi@louisiana.edu).

Lida Kouhalvandi is with the Department of Electrical and Electronics Engineering, Dogus University, 34775 Istanbul, Turkey (e-mail: lida.kouhalvandi@ieee.org).

Serdar Ozoguz and Ece Olcay Gunes are with the Department of Electronics and Communication Engineering, Istanbul Technical University, 34469 Istanbul, Turkey (e-mail: ozoguz@itu.edu.tr; gunesec@itu.edu.tr).

Digital Object Identifier 10.1109/LES.2023.3298734

with optimization focusing on latency and power consumption (i.e., objective optimization function).

The process begins by updating the logic gates' functionality through Shannon decomposition [9], which involves remodeling each transistor-based logic gate. "Dummy transistors" are added where necessary to equalize the number of transistors used in the pull-up and/or pull-down branches. Subsequently, the presented regression DNN method from [7] is employed to determine the circuit design parameters, W and L, automatically. We utilize the TSEMO algorithm [8], to optimize latency and power consumption, which leverages the POF and effectively generates the set of optimal tradeoffs. This algorithm offers improved efficiency compared to other methods reported in [8].

Our proposed SC logic gate (SLG) library is specifically beneficial for SC-based image processing (e.g., edge detection and mean filtering) and machine learning applications such as quantized neural networks (QNNs). Weight quantization in SC-based systems needs a cascaded topology with finer-grained divisions for network weight values. Deeper modules are employed to achieve smaller weights, requiring a latency-aware solution to meet critical path delay constraints. The proposed optimization technique uses the LSTM network architecture to model and size logic gates. This approach enables the construction of stochastic logic gate libraries that are both latency and power efficient. Subsequently, we apply a previously proposed module by Li et al. [10], [11] in a QNN application to measure the delay by further checking the accuracy performance. Finally, we provide a co-simulation of circuit design within the overall system, along with a preliminary latency analysis.

II. Proposed Optimization Method Using Deep Neural Network

A. Modeling of SLGs

As discussed in [5], latency issues are prevalent in SC designs, leading to the misalignment of rise-time and fall-time in the output phase. To address this problem, equalizing the worst-case and best-case propagation delays for both high-to-low and low-to-high input patterns is crucial. This can be achieved by implementing parallel branches of transistors, where one branch is active (logic 1), and the other is inactive (logic 0). Consequently, Shannon decomposition ($F_{\text{Shannon}} = x_i f_{x_i} + \bar{x}_i f_{\bar{x}_i}$) is employed. By utilizing this expansion, we can ensure that $x_i f_{x_i}$ evaluates to logic 1 and $\bar{x}_i f_{\bar{x}_i}$ evaluates to logic 0. Hence, each parallel branch will be active only once. The propagation delays can be equal for all input patterns by equalizing the number of transistors in each parallel branch and incorporating necessary dummy transistors.

Our study introduces CMOS logic gates. Our proposed modeling approach becomes essential for CMOS-based logic gates, specifically `NOR-2` and `NAND-2` gates. We apply Shannon decomposition and incorporate dummy transistors to address this. `XOR-2` and `MUX 2:1` gates have their own Shannon decomposition equations and do not require dummy transistors. Fig. 2 for CMOS-based gates.

B. Multiobjective Optimization for Sizing Transistors

The optimization of transistor sizing for analog circuits, particularly for SC designs, involves considering two crucial factors: 1) latency and 2) power consumption. A multiobjective optimization process is necessary to address this, utilizing objective functions related to these two metrics. Kouhalvandi et al. [7] employed the TSEMO approach, which relies on POF and demonstrates high calculation accuracy. Similarly, we adopt the TSEMO process to optimize and determine the POF for latency and power. The sizing of transistor width (W) and length (L) is performed within this optimization. Fig. 1 illustrates the automated optimization process based on a DNN for sizing the SLGs.

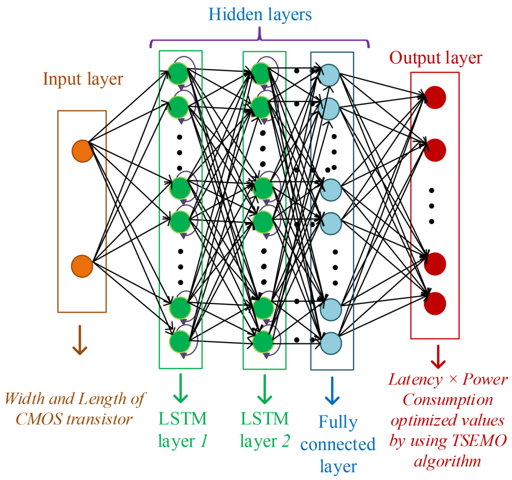

Hidden layers
Input layer
Output layer
Width and Length of CMOS transistor
LSTM layer 1
LSTM layer 2
Fully connected layer
Latency × Power Consumption optimized values by using TSEMO algorithm

Fig. 1. DNN construction for multiobjective optimized transistor sizes.

TABLE I
Comparative Delay Ratios of Existing Logic Library, and CMOS Libraries

Gates	Logic Gate Library in AMS 180 nm					
	WR/WF	BR/BF	WR/BF	BR/WF	WR/BR	WF/BF
Inverter	2.15	2.15	2.15	2.15	1.00	1.00
Buffer	1.53	1.53	1.53	1.53	1.00	1.00
XOR	1.40	3.00	6.63	3.09	2.20	9.31
MUX 2:1	1.06	1.34	2.51	1.98	1.86	2.67
NOR	4.56	4.57	5.66	3.68	1.23	1.24
NAND	1.70	1.06	2.44	1.34	2.29	1.43

Gates	Proposed CMOS Logic Gates					
	WR/WF	BR/BF	WR/BF	BR/WF	WR/BR	WF/BF
Inverter	1.00	1.00	1.00	1.00	1.00	1.00
XOR	1.01	1.06	1.38	1.48	1.47	1.39
MUX 2:1	1.06	1.44	1.18	1.82	1.71	1.26
NOR	1.02	1.00	1.13	1.17	1.14	1.16
NAND	1.08	1.03	1.08	1.14	1.04	1.18

After obtaining suitable models for the logic gates in Section II-A, the sizes of the transistors are optimized using the proposed multiobjective optimization method. To achieve this, a DNN is trained using data generated from a co-simulation environment between MATLAB and Cadence [12]. The logic gate circuits described in Section II-A are redesigned to incorporate transistor sizing. The transistor width (W) and length (L) are randomly iterated, and the simulation results of the circuits are fed into MATLAB to generate appropriate training data.

We apply the TSEMO method in MATLAB, utilizing the training data to construct an accurate DNN model representing the SLGs. Fig. 1 illustrates the structure of the LSTM-based DNN, which consists of two LSTM layers, each with 50 neurons and one fully connected layer. During the training phase, we obtain the labeled values of W and L for the desired latency–power consumption values. Fig. 2 presents the optimized transistor sizes for CMOS-based logic gates.

Table I compares the ratios of worst-case rise (WR) time, worst-case fall (WF) time, best-case rise (BR) time, and best-case fall (BF) time between the standard built-in logic libraries and the proposed SLG library. The results demonstrate that the proposed SLG library achieves a latency ratio of approximately 1, indicating consistent delays across all input patterns. This finding highlights the successful resolution of latency issues in SC circuit designs. We simulated and optimized the

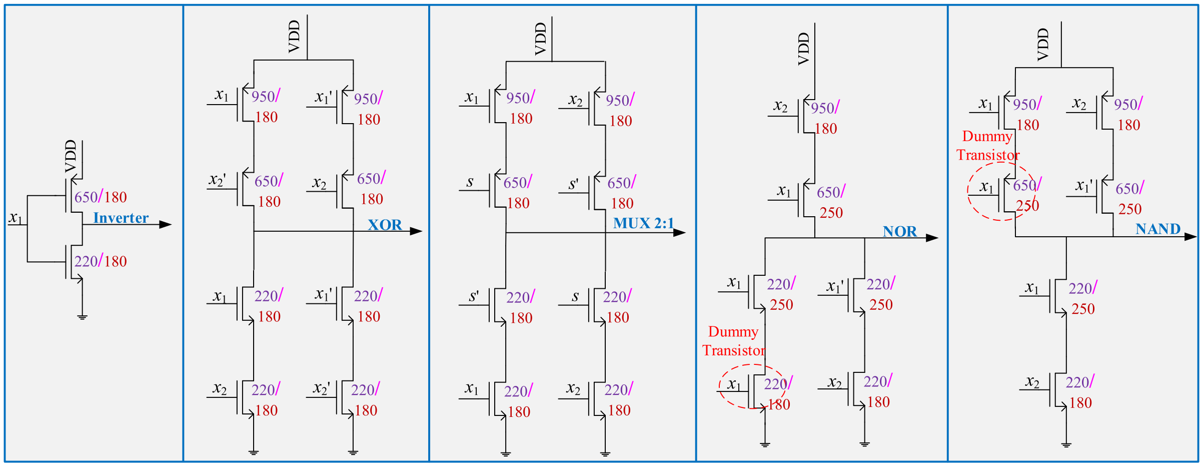


Fig. 2. CMOS-based logic gates. From left to right, the gates are the inverter, two-input XOR, MUX, NOR, and NAND. The dimension of each transistor is in nm unit, and "/" denotes width/length.

TABLE II
SC-Based Edge Detectors From SOTA Works

Ref.	Implemented Filter	Design
[15]	Roberts cross	2 × NOT gate, 3 × 2:1 MUX, 2 × SC abs. value
[16]	Gaussian	8 × 2:1 MUX
[17]	Roberts cross (using correl. inputs)	2 × NOT, 2 × XOR, 1 × 2:1 MUX
[18]	Prewitt	4 × 2:1 MUX, 1 × XOR gate
[19]	Sobel	4 × 2:1 MUX, 1 × XOR gate, 2 × AND gate
[20]	Prewitt	5 × 2:1 MUX, 1 × NOT gate

TABLE III
Power Consumption (mW) of the SOTA SC-Based Edge Detector Designs With and Without SLG

Optimization	[15]	[16]	[17]	[18]	[19]	[20]
w/o SLG	17.56	25.03	12.51	13.22	18.11	15.12
w/ SLG	15.47	22.51	11.24	10.56	16.18	12.97

logic gates using the AMS 180-nm technology in the Cadence environment.

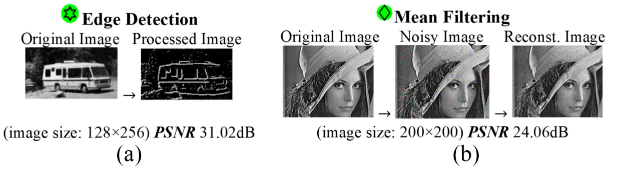
III. Application of the Proposed SLG Library

A. SC Image Processing

We first evaluate the performance of the proposed library for the state-of-the-art (SOTA) SC image-processing edge detection architectures. We utilize a Cadence library and the proposed circuit-level logic gates to implement the SC designs. Table II lists the implemented SC edge detection designs. Table III compares the implemented designs in terms of power consumption *with* (w/) and *without* (w/o) using the proposed SLG library. As can be seen, using the proposed library reduces power consumption in all cases. Fig. 3(a) and (b) visually depicts the performance of the SLG library. The *Caravan* sample image was processed using the Cadence-MATLAB co-operated environment for the edge detection application. The noise removal mean filtering application was also evaluated using the *Lena* image in Fig. 3(b). We observed that the accuracy performance of the SLG library is in the acceptable range of peak signal-to-noise ratio (PSNR) values. Like edge detection, the mean filtering application also proves the power effectiveness of the proposed SLG.

B. SC QNNs

Prior work has applied SC to design hardware-efficient neural network systems [19], [20]. Li et al. [10], [11] redesigned the SC-based NN with the quantization property [21]. By using their SOTA stochastic unary code adder (SUC-adder), we measure the performance of the SLG in terms of critical path delay. For the experiments, we set the quantization sensitivity to 2 bits for the network parameters. This representation requires 1/4, 1/2, and 3/4 fractional values in the weight bit-streams. Fig. 4 depicts the proposed simulation environment. The overall system is simulated in MATLAB Simulink, co-processing the Cadence Virtuoso environment. Thus, the optimized circuit module is set for measurement on the same platform by using the partial design property of the simulation platform. Li et al. [10], [11] efficiently used the SUC-adder for quantized weight processing. The number of logic 1s in the weight stream is adjusted fractionally depending on the quantization sensitivity. This affects the topology of the SUC-adder in terms



(image size: 128×256) ***PSNR*** 31.02dB
(a)

(image size: 200×200) ***PSNR*** 24.06dB
(b)

✡ → N=256, using circuit topology of [18], $w/$ optimized library, SLG, total P=10.56mW. ◇ → N=256, using circuit topology of [20], $w/$ optimized library, SLG, total P=8.56mW, while w/o optimization P=11.67mW.

Fig. 3. Real applications of the cooperative environment using SLG: visual result of (a) edge detection and (b) mean filtering for noisy ($\mu = 0$ and $\sigma^2 = 0.008$) image.

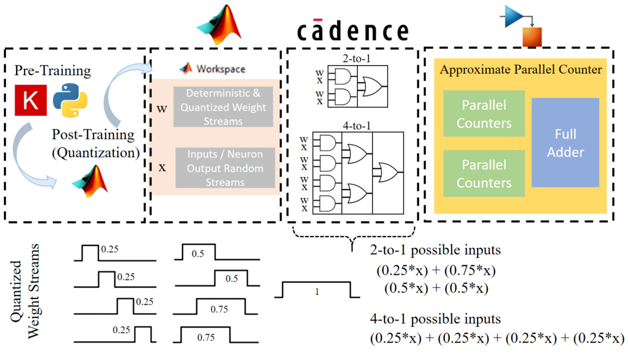


Fig. 4. Proposed simulation for QNN with 2-bit quantization.

TABLE IV
LATENCY OF THE SUC-ADDER IN QNN FOR DIFFERENT DESIGNS

Critical Path (ns) [CMOS] - MUX			
2:1	**4:1**	**8:1**	**16:1**
0.992	1.272	1.864	2.017

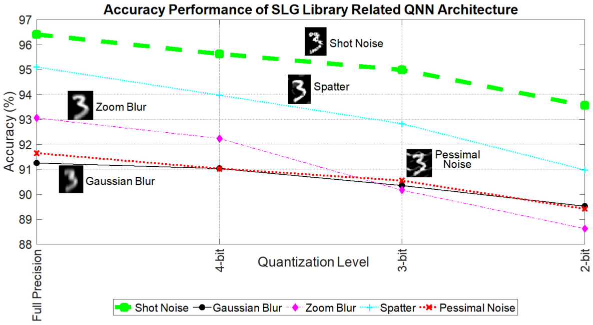


Fig. 5. Accuracy of QNN architecture constructed by SLG ($N = 64$).

of the cascaded elements. The SUC-adder can be considered a MUX-like structure, where the *select* input stream is trivially applied like the weight inputs in fractions.

In our experiments, we obtained model inferences using a 2-hidden-layer multilayer perceptron (784-200-100-10) for classifying the MNIST handwritten digit dataset. We start the training process on Python using the Keras framework for full precision training. Subsequently, we apply a post-training method to quantize weights and activations. The activation function is a sigmoid, selected to produce positive-only outputs for neurons. To simulate the hardware, we built a co-processing environment using MATLAB Simulink and Cadence Virtuoso, combining digital and analog components. When using 2-bit quantization, our network model achieves an accuracy of 97.39% for $N = 64$-bit SC bit-streams. Table IV compares the critical path latency of the CMOS design approach applied to the SUC-adder module design. For $N = 64$, the CMOS design provides better critical path delays, while the latency changes are reported for different MUX structures.

Finally, we evaluate the performance of classifying corrupted handwritten images using the MNIST-C dataset. For the experiments, we employ a multilayer perceptron (784-200-100-10) with two hidden layers for classification. Like in previous experiments, the training process begins in the Keras framework for training with full precision. Subsequently, we employ a post-training approach to quantize weights and activations, employing a fine-tuning technique, as mentioned earlier. The chosen activation function is again sigmoid, which ensures positive-only neuron outputs. Fig. 5 illustrates the classification accuracy for five different corruption types based on 2-bit, 3-bit, and 4-bit quantization levels facilitated by the SLG-based MUXs. The performance of the QNN can be compared by considering the corruption types and the quantization levels. Our results demonstrate the applicability of the SLG approach and its efficiency. The proposed platform is flexible not only from a hardware perspective but also for the exploration of multiple datasets considering the application performance.

IV. CONCLUSION

In this study, we introduced a novel methodology for optimizing basic logic gates for SC designs. Initially, we modeled the logic gates by applying Shannon decomposition to standard logic gates. To determine the sizes of the transistors, we utilized an LSTM-based DNN trained through regression to optimize latency versus power using the POF. The optimization process was automated and performed using the Cadence 180-nm technology in the MATLAB platform. Once the optimal logic gates were obtained, we proceeded to simulate the SUC-adder in the QNN. We evaluated the critical path latency in the 2-to-1, 4-to-1, 8-to-1, and 16-to-1 MUX structures, considering CMOS-based design primitives. This work demonstrates how low-level optimization techniques can be applied to high-level applications. We compared the power consumption of the SOTA SC-based image processing architectures with and without the optimized library utilization. QNN performance in the case of multiple datasets and quantization levels was evaluated using the designed SLG and the cooperative platform. Finally, we presented an analog and digital design co-simulation on a unified platform leveraging the capabilities of MATLAB and Cadence tools.

REFERENCES

[1] M. Parhi, M. D. Riedel, and K. K. Parhi, "Effect of bit-level correlation in stochastic computing," in *Proc. IEEE Int. Conf. DSP*, 2015, pp. 463–467.

[2] V. T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," in *Proc. DATE*, 2018, pp. 1417–1422.

[3] W. Qian, C. Wang, P. Li, D. J. Lilja, K. Bazargan, and M. D. Riedel, "An efficient implementation of numerical integration using logical computation on stochastic bit streams," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2012, pp. 156–162.

[4] Z. Wang, D. Larso, M. Barker, S. Mohajer, and K. Bazargan, "Deterministic shuffling networks to implement stochastic circuits in parallel," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 8, pp. 1821–1832, Aug. 2020.

[5] A. J. Groszewski and E. E. Swartzlander, "A variable-latency architecture for accelerating deterministic approaches to stochastic computing," in *Proc. 53rd Asilomar Conf.*, 2019, pp. 608–613.

[6] Z. Li, Z. Chen, Y. Zhang, Z. Huang, and W. Qian, "Simultaneous area and latency optimization for stochastic circuits by d flip-flop insertion," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1251–1264, Jul. 2019.

[7] L. Kouhalvandi, O. Ceylan, and S. Ozoguz, "Automated deep neural learning-based optimization for high performance high power amplifier designs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 12, pp. 4420–4433, Dec. 2020.

[8] E. Bradford, A. M. Schweidtmann, and A. Lapkin, "Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm," *J. Global Optim.*, vol. 71, no. 2, pp. 407–438, Jun. 2018.

[9] P. Kerntopf, "New generalizations of Shannon decomposition," in *Proc. Int. Workshop Appl. Reed-Muller Exp. Circuit Design*, 2001, pp. 1–9.

[10] B. Li, M. H. Najafi, and D. J. Lilja, "Low-cost stochastic hybrid multiplier for quantized neural networks," *J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 2, p. 18, Mar. 2019.

[11] B. Li, M. H. Najafi, B. Yuan, and D. J. Lilja, "Quantized neural networks with new stochastic multipliers," in *Proc. ISQED*, 2018, pp. 376–382.

[12] "Cadence virtuoso AMS designer simulator." Accessed: May 15, 2023. [Online]. Available: http://mathworks.com/products/connections/product_detail/cadence-virtuoso-ams-designer.html

[13] P. Li and D. J. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *Proc. IEEE 29th Int. Conf. Comput. Design (ICCD)*, Amherst, MA, USA, 2011, pp. 154–161.

[14] H. Abdellatef, M. K. Hani, and N. Shaikh-Husin, "Accurate and compact stochastic computations by exploiting correlation," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 27, no. 1, pp. 547–564, 2019.

[15] M. H. Najafi and D. J. Lilja. "High-speed stochastic circuits using synchronous analog pulses," in *Proc. 22nd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Chiba, Japan, 2017, pp. 481–487.

[16] M. Ranjbar, M. E. Salehi, and M. H. Najafi, "Using stochastic architectures for edge detection algorithms," in *Proc. 23rd Iran. Conf. Electr. Eng.*, Tehran, Iran, 2015, pp. 723–728.

[17] H. Joe and Y. Kim, "Novel stochastic computing for energy-efficient image processors," *Electronics*, vol. 8, no. 6, p. 720, 2019.

[18] S. Aygun, "Stochastic bitstream-based vision and learning machines," Ph.D. dissertation, Istanbul Tech. Univ., Dept. Electron. Eng., Istanbul, Turkey, 2022.

[19] B. Li, Y. Qin, B. Yuan, and D. J. Lilja, "Neural network classifiers using stochastic computing with a hardware-oriented approximate activation function," in *Proc. IEEE ICCD*, 2017, pp. 97–104.

[20] A. Ren et al. "SC-DCNN: Highly-scalable deep convolutional neural network using stochastic computing," in *Proc. ASPLOS*, 2017, pp. 405–418.

[21] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 187, pp. 1–30, 2018.