Design and Fabrication of Flow-based Edge Detection Memristor Crossbar Circuits

A Massively Parallel Search using a Human Perception Objective

Jodh Singh Pannu *Student Member, IEEE,* Sunny Raj *Student Member, IEEE,*
Steven Lawrence Fernandes *Senior Member, IEEE,* Dwaipayan Chakraborty *Member, IEEE,*
Sarah Rafiq *Student Member, IEEE,* Nathaniel Cady *Member, IEEE,* Sumit Kumar Jha, *Member, IEEE*

***Abstract*—We design and fabricate a flow-based circuit for edge detection in images that exploits device-level parallelism in nanoscale memristor crossbars. In our approach, a corpus of human-labeled edges in BSDS500 images is used to learn an edge detection function with ternary values: true, false, and don't-care. A Boolean crossbar design implementing an approximation of this ternary function using in-memory flow-based computing is then obtained using a massively parallel simulated annealing search executed on GPUs. We demonstrate the success of our approach by fabricating the memristor circuit on a 300nm wafer platform using a custom 65nm CMOS/ReRAM process technology. We demonstrate that our flow-based computing approach is either faster, more energy-efficient or produces fewer incorrect edges than other competing approaches. We show that our design has power and area requirements that are 3.3x and 2.5x lower, respectively, than the previous state-of-the-art.**

I. Introduction

MULTIPLE computer vision and AI algorithms rely on edge detection as a preliminary step [1]. Fast and efficient edge detection in images using dedicated hardware can enhance the usability of these algorithms, specially in edge computing and IoT. Flow-based in-memory computations has been shown to be both time and energy-efficient for simple arithmetic operations. These advantages of flow-based computing are obtained by bypassing the memory bottleneck of traditional von Neumann architectures and exploiting the device-level parallelism of nanoscale memristor crossbars.

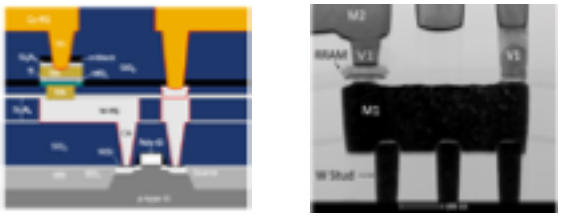
In this paper, we design and fabricate an edge detector that leverages the time and energy efficiency of flow-based computing on memristor crossbars. We create a ternary-valued function derived from manually segmented images of the BSDS500 dataset as the ground truth for edge detection [2]. The ternary function maps a pixel pair to a true, false, or don't-care value, which corresponds to an edge, not an edge, and an uncertainty about the existence of an edge between a pixel pair. A crossbar implementing this ternary function is then found using simulated annealing on more than 4000 GPU cores.

J. Pannu, S. Raj, S. Fernandes and S. Jha are with the Computer Science Department at the University of Central Florida, Orlando, FL.

D. Chakraborty is with the Future Technologies Group at the Oak Ridge National Laboratory, Oak Ridge, TN.

S. Rafiq and N. Cady are with the Center for Nanoscale Sciences and Engineering at the SUNY Polytechnic Institute, Albany, NY.

Manuscript received February 2, 2020.



(a) Schematic (b) TEM Cross-section

Fig. 1: A schematic and TEM cross-sectional image of the HfO_2 ReRAM device implementing our edge detection crossbar design [3]. The custom ReRAM module was fabricated on a 300nm wafer using a 65nm CMOS process.

An objective function based on the human perception of image similarity is used to control the simulated annealing based search. A massively parallel simulated annealing search is employed to find crossbar designs of multiple sizes with varying accuracy and energy requirements. We design memristor crossbars of sizes 5×5, 6×6, 7×7, and 8×8, and then fabricate them on a custom ReRAM module on a 300mm wafer platform using a 65nm CMOS process technology. We experimentally demonstrate that our designs can be successfully fabricated on physical devices. We make the following new contributions in this paper:

1) We exploit a massively-parallel simulated annealing search for the optimal crossbar design using two Tesla V100s with more than 4000 GPU cores. We design a new method to calculate the output of crossbar circuits efficiently. This parallel approach combined with an efficient calculation of crossbar output allows us to search for smaller crossbars designs that have lower power consumption. When compared to earlier work [4], our design produces crossbars that are up to 2.5x smaller and have up to 3.3x lower power requirements.
2) We employ the human perception of similarity between two images as the cost metric for the search algorithm. This allows us to create crossbar designs that generate edges that are visually similar to the ground truth. We compare the edges generated by our designs to those generated by earlier work [4] and find a 2.78x improvement in the perceptual difference score.

3) We have fabricated and tested our designs on a physical ReRAM module to show that our solution is both feasible and practical. A single fabricated ReRAM device implementing the edge detection crossbar design is shown in Figure 1. Previous work to create a flow-based ReRAM device was published in ISCAS 2016 by Alamgir and others [5]. But, that work only fabricated a relatively simple one-bit adder. Our work brings the advantages of in-memory flow-based computing to a fundamental computer vision problem.

II. Related Work

There has been continued interest in logic design using memristor crossbars [6]–[9]. Various techniques that leverage the unique properties of memristor crossbars to perform neuromorphic computing have been proposed in literature [10]–[13]. These machine learning applications of memristor crossbars are impressive, but rely on integration with CMOS devices to perform computations. Flow-based computing on memristor crossbars is a departure from the traditional computing paradigm and uses device-level parallelism to enable in-memory computations that can overcome the memory bottleneck of the von Neumann architecture. A memristor crossbar implementing a one-bit adder using flow-based computing was presented in [14]. Subsequent work on flow-based computing used binary decision diagrams (BDD) [15], free binary decision diagrams (FBDD) [16], automated synthesis via satisfiability modulo theory [17], and AI-based search procedures [18] to synthesize memristor crossbar circuits. Alamgir and others presented a full adder crossbar implementation on a ReRAM device [17].

The design of edge detection circuits using flow-based memristor crossbars has been presented in [4]. However, they did not demonstrate that their designs can be fabricated on real-world memristors. Our designs are both more compact and more energy-efficient that the designs reported in [4]. Pajouhi and Roy designed a memristor circuit based on ant colony optimization to perform edge detections [19]. Our approach is both space and energy-efficient when compared to the input-aware flow-based approach, while it is faster than the ant colony optimization based approach. A comparison of our approach to these approaches is presented in Table I. A flurry of recent activity in memristor crossbars makes it difficult to enumerate all related work in this area.

Designs	PSNR	PerceptualDiff Score	Switching Power	Time
Input-aware [4]	8.9	20458	1.188mW	-
Swarm-based [19]	-	-	**0.220mW**	68μS
8×8 crossbar	13.7	6563	0.858mW	16μS
7×7 crossbar	**14.6**	**5359**	0.528mW	16μS
6×6 crossbar	14.5	5401	0.440mW	**16μS**
5×5 crossbar	14.4	5818	0.352mW	16μS

TABLE I: Comparison of the performance of our approach to input-aware flow-based [4] and swarm-based approaches [19]. The 8×8 to 5×5 crossbars have been generated using our approach. Our design generates edges that are similar to the ground truth, as demonstrated by the high Peak signal-to-noise ratio (PSNR) values and lower perceptual difference scores [20]. Our design uses less power than [4], and it is faster than [19].

III. Approach

The design of our edge detection circuit is based on flow-based computing [5], where an electric pulse is applied to one nanowire, and the output current is observed from another nanowire. The crossbar array consists of programmable and non-programmable memristors. Programmable memristor contains the current input value and can change depending upon the input, whereas non-programmable memristors stay constant and do not change. An illustration of this approach implementing a full adder on a 4×5 crossbar is shown in Figure 2. The value of the input determines the programming of the memristors. Memristors in green have been set to ON, and memristors in gray have been set to OFF. Current flow to calculate the sum when the values of $A = 1, B = 1$ and $C = 1$ is shown by the red path. Current flow in the top row implies that the sum is 1, whereas no flow implies that the sum is 0.

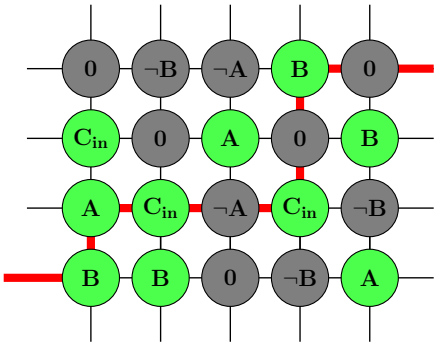


Fig. 2: A crossbar design that implements a full adder. Each memristor is labeled with an input. The states of the memristors after receiving an input is updated according the input. For inputs $A = 1, B = 1$ and $C = 1$ the memristors labeled A, B and C are set to ON, shown by green, and memristors labeled $\neg A$, $\neg B$ and $\neg C$ are set to OFF, shown by grey. The red path shows the current flow from the input wire (bottom row) to the output wire (top row).

More complex boolean functions that perform edge detection can be implemented on larger crossbars. Figure 3 shows a crossbar synthesised by our approach to perform edge detection in images. Each crossbar accepts a pixel pair and a flow in the rightmost column indicates an edge between a pixel-pair. The problem of finding the memristor design that can effectively find edges between the pixel pairs in the image is solved in two steps: (1) finding the ternary function which performs edge detection, and then (2) finding a crossbar design that implements the ternary function found in step (1).

A. Ternary Value Function for Edge Detection

The ternary function maps pixel pairs to true, false, and don't-care values. True value denotes the presence of an edge between the pixel, false value denotes the absence of an edge, and the don't-care values denote pixel pairs where the human-labeled data set does not indicate a consistent response. Edge information obtained from the human-annotated BSDS500 dataset is used to calculate the pixel pair to value mapping using the following equations:

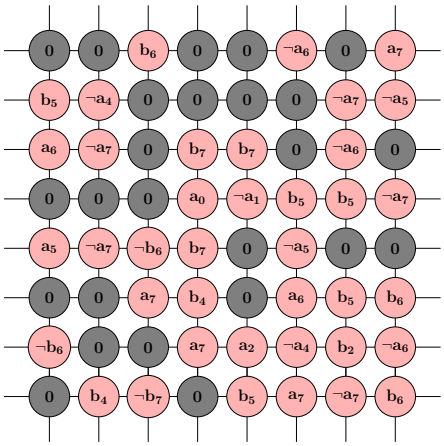
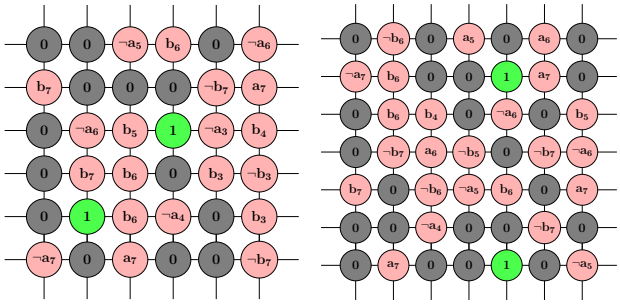


Fig. 3: A 8×8 memristor crossbar design for edge detection. An electric pulse is applied to the bottom row, and the output current observed from the right column. Current flow in the column implies an edge between the pixel pair, whereas the lack of a flow implies that no edge exists between the pixels.

$$V(x,y) = \begin{cases} \text{T, if } f_e(x,y)/f_p(x,y) \geq \theta^E \text{ and } f_e(x,y) \geq \theta^P \\ \text{F, if } f_e(x,y)/f_p(x,y) < \theta^E \text{ and } f_e(x,y) \geq \theta^P \\ \text{Don't-care, otherwise} \end{cases}$$

Here, x, y are values of the pixel pair, $f_p(x,y)$ is the frequency of occurrence of pixel having values x and y in the image dataset, and $f_e(x,y)$ is the frequency of observing an edge between pixel pairs x and y in the human-annotated dataset. The parameter θ^P and θ^E are the values of the threshold of f_p and f_e, respectively, that determines the mapping from pixel pair to the ternary value.



(a) A 6x6 design (b) A 7x7 design

Fig. 4: Edge detection memristor crossbar designs implemented using 7x7 and 6x6 crossbars.

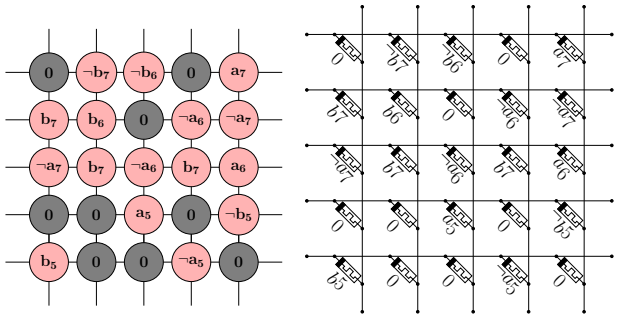
B. Ternary Function to Crossbar Design

A massively parallel implementation of the simulated annealing algorithm is used to search for crossbar designs that implement the approximated ternary function. The cost function of the simulated annealing algorithm is designed to penalize disagreement between the ternary function and the crossbar output, and the disagreement between the generated edges and the human-annotated edges. Flow-based crossbar designs produce Boolean values as output and generate a true or false result on a given input, whereas the ternary function produces a true, false, and don't-care as output. The disagreement D_T between the crossbar output $C(x,y)$ and the ternary function $V(x,y)$ is calculated using the following function:

$$d(x,y) = \begin{cases} 0, & C(x,y) = V(x,y) \\ 0, & \text{if } V(x,y) = \text{Don't Care} \\ 2, & \text{if } V(x,y) = \text{T and } C(x,y) \neq \text{T} \\ 1, & \text{if } V(x,y) = \text{F and } C(x,y) \neq \text{F} \end{cases}$$

$$D_T = \sum_{x,y} d(x,y)$$

Here, x, y are values of the pixel pair, $V(x,y)$ is the ternary value function output for a given pixel pair, and $C(x,y)$ is the crossbar output. The disagreement D_T is then combined with an image similarity score that can capture the perception of a human observer to find the final disagreement. In our method, we have used the inverse of the perceptual difference (PerceptualDiff) score to obtain the total disagreement D.



(a) A 5x5 design (b) Circuit Representation

Fig. 5: Edge detection memristor crossbar design implemented using 5x5 and the corresponding circuit representation.

Obtaining the crossbar output $C(x,y)$ to calculate the disagreement between the ternary function and the crossbar output in simulation can be time-consuming and can lead to long search time. Evaluating the truth table entries from a crossbar naively has a time complexity of $O(2^b RC)$, where R and C are the numbers of rows and columns respectively in the crossbar, and b is the size of the input. To quickly calculate the output of crossbars circuits, we model the circuit as a directed acyclic graph (DAG) and then only compute incremental changes in the total disagreement D as the design evolves during our search process.

Modelling crossbar circuit as a DAG: For a crossbar of size $R \times C$ with R rows and C columns, a Directed Acyclic Graph (DAG) G can be constructed to emulate the dynamics of the crossbar. The DAG G is divided into components G^k arranged temporally, where each component emulates the dynamics happening within the time it takes for current to cross one memristor in the crossbar. Each component G^k consists of nodes $r_i^{(k)}$ and $c_j^{(k)}$, which represent ith row and jth column wires respectively, and directed edges $e(r_i^{(k)}, c_j^{(k)})$ that can capture the flow of current through a memristor from wire r_i to c_j at time step k. The components G^{k-1} and G^k are connected by directed edges $e(c_i^{(k-1)}, r_j^{(k)})$ which capture the

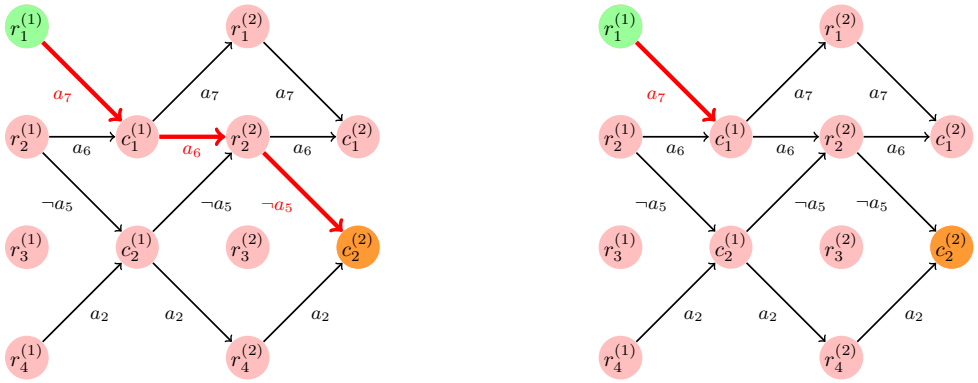


Fig. 6: The input and output nodes of the DAG are represented by $r_1^{(1)}$ and $c_2^{(2)}$ respectively. (**left**) Evidence $E = \{a_7, a_6, \neg a_5\}$ of true in the truth table. Knowing the values of three variables tells us that output of the crossbar is true. (**right**) Evidence $E' = \{\neg a_7\}$ of false in the truth table. Knowing the value of one variable tells us that the output of the crossbar is false.

flow of current through a memristor from wire c_i to r_j at time step k-1. An edge between two wires exists if memristor connecting the wires is a programmable turned-on memristor or a non-programmable memristor that is always turned-on. The total number of components K in the graph depends upon the size of the crossbar and is equal to RC. The input pulse is applied to the bottom row and is denoted by the node $r_1^{(1)}$ in the DAG. The output node is denoted by $c_C^{(K)}$.

Given a pair of input pixel, an edge exists between the pixels if there is a flow from the input node to the output node. Such a path is called evidence E for a true truth table entry for the input. We avoid the explicit calculation of flow from the input to the output for a given crossbar by utilizing a special case of the max-flow min-cut problem, where the flow of each edge is equal to unity, and the source and sink nodes are the input and output nodes respectively. The set of edges that form the min-cut of DAG is the evidence E' for a false entry of the truth table. There exists a set of evidences $\mathcal{E} = \{E_1, E_2.., E_t\}$ and $\mathcal{E}' = \{E'_1, E'_2.., E'_{t'}\}$ which account for all the output of the crossbar. Knowing the input value of only a few variables that satisfy an evidence can allow us to know the output of the whole crossbar and removes the need for expensive calculations. An example of this approach is shown in Figure 6. For an 8-bit input, observing a false value of input a_7 allows us to know that the output of the whole crossbar is false irrespective of other values of the input.

IV. Results

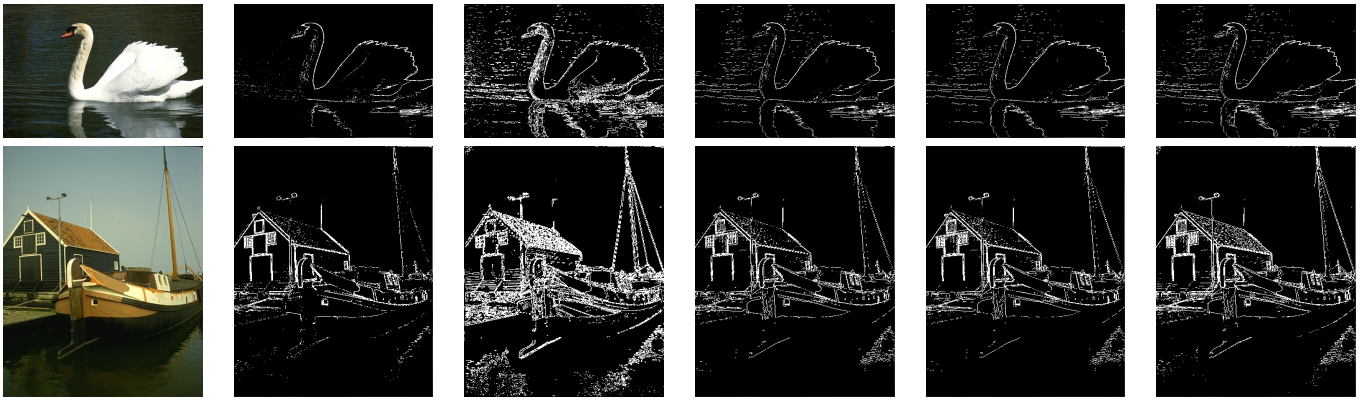
We synthesized memristor crossbar designs of sizes 5×5, 6×6, 7×7 and 8×8 using our approach. Our approach of using a ternary function with a don't-care condition allowed us to skip 54% of the input pixel pairs while searching for crossbar designs. The crossbar designs are presented in Figures 3, 4 and 5. We tested our design on the BSDS500 dataset and show that the edges computed by our design have lower power to signal noise ratio (PSNR) and higher perceptual difference (PerceptualDiff) score when compared to earlier results [4]. We use the memristor programming circuits provided in [19] and [5] to compare the performance of the fabricated devices and designs. We compare the speed of edge detection by our crossbar-based memristive computing design to the swarm-based approach presented in [19], and show that our memristor crossbar design takes less time to compute edges than the swarm-based approach.

In Table I, we observe that the best PSNR and PerceptualDiff score between the ground truth and the computed edge are 14.6 and 5359 respectively, whereas the PSNR value of the input-aware method is lower at 8.9, while the PercetualDiff score is higher at 20458. A higher PSNR and a lower perceptual difference (PerceptualDiff) score denote higher conformance of the ground truth with the computed edge. Similarly, we observe in Figure 7 that the edges computed by our method have less noise and are closer to the ground truth when compared to the input-aware crossbar design. Our 5×5 design is 2.5 times smaller than the designs produced by the input-aware method, and requires 0.418mW of power, which is 3.3 times less than the power required by the input-aware crossbar design.

Pixel A	Pixel B	Expected logical output	Observed output
01100000	01101001	0	3.5V
00110110	00100110	0	3.5V
01101111	10010100	1	1.7V
10011110	01111100	1	1.6V
10010000	01111100	1	1.7V
01001100	01110111	1	1.5V

TABLE II: Outputs on an 8×8 1T1R device using randomly selected pixel pairs. Logical 0 corresponds to about 3.5V and logical 1 corresponds to 1.5V – 1.7V.

The synthesized designs have been verified experimentally on an 8×8 1T1R device array fabricated on a 300nm wafer. We chose input pixel pairs, programmed them on the ReRAM device, applied a current of 2μA to the input, and observed the output voltages difference across the input and the output. A high resistance state (HRS), leading to an observed high voltage implies no edge, while a low resistance state (LRS) leading to a low voltage implies the existence of an edge. The result of the experiments is presented in Table II. We observe a voltage difference of 1.9V between high and low resistance states. These results experimentally verify the correctness of the first flow-based edge detection design implemented on a ReRAM device.



(a) (b) (c) (d) (e) (f)

Fig. 7: (a) Original image, (b) Human annotated ground truth, (c) Edges generated using input-aware crossbar design [4], (d),(e) and (f) Edges generated using our 5x5, 6x6 and 8x8 crossbar designs respectively. The number of pixels contributing to noise in (c) is greater when compared to our designs. For example, in (c), there is a lot of noise on the roof of the house; our designs generate edges that are similar to the ground truth.

V. Conclusions and Future Research

We present the design and fabrication of an edge detection circuit using flow-based computing in nanoscale memristor crossbars. Our work is the first to experimentally fabricate a flow-based computing for a practical application, such as edge detection. We demonstrate that our approach produces designs that are 2.78x better in the perceptual difference score, 2.5x smaller and 3.3x more energy-efficient that the state-of-the-art in flow-based computing using memristor crossbars [4].

Crossbar design for a broader distribution of images is an important direction we are going to pursue in the immediate future. We will focus on crossbar synthesis for other relevant applications like convolution, clustering, regression and pattern-matching.

References

[1] Lu Fang, O. C. Au, Y. Yang, Weiran Tang, and Xing Wen, "A new adaptive subpixel-based downsampling scheme using edge detection," in *2009 IEEE International Symposium on Circuits and Systems*, May 2009, pp. 3194–3197.

[2] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423.

[3] S. Rafiq, K. Beckmann, J. Hazra, M. Liehr, S. K. Jha, and N. C. Cady, "Investigation of multi-level reram in 65nm cmos for logic-in-memory applications," *AVS 66th International Symposium and Exhibition*, 2019.

[4] D. Chakraborty, S. Raj, S. L. Fernandes, and S. K. Jha, "Input-aware flow-based computing on memristor crossbars with applications to edge detection," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pp. 1–1, 2019.

[5] Z. Alamgir, K. Beckmann, N. Cady, A. Velasquez, and S. K. Jha, "Flow-based computing on nanoscale crossbars: Design and implementation of full adders," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 1870–1873.

[6] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Magic—memristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, 2014.

[7] D. N. Yadav, P. L. Thangkhiew, and K. Datta, "Look-ahead mapping of boolean functions in memristive crossbar array," *Integration*, 2018.

[8] S. Shirinzadeh and R. Drechsler, "Logic synthesis for in-memory computing using resistive memories," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 375–380.

[9] T. Vatwani, A. Dutt, D. Bhattacharjee, and A. Chattopadhyay, "Floating point multiplication mapping on reram based in-memory computing architecture," in *VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), 2018 31st International Conference on*. IEEE, 2018, pp. 439–444.

[10] I. K. Schuller, R. Stevens, R. Pino, and M. Pechan, "Neuromorphic computing–from materials research to systems architecture roundtable," USDOE Office of Science (SC)(United States), Tech. Rep., 2015.

[11] M. Chu, B. Kim, S. Park, H. Hwang, M. Jeon, B. H. Lee, and B.-G. Lee, "Neuromorphic hardware system for visual pattern recognition with memristor array and cmos neuron," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2410–2419, 2015.

[12] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2011.

[13] T. Serrano-Gotarredona, T. Prodromakis, and B. Linares-Barranco, "A proposal for hybrid memristor-cmos spiking neuromorphic learning systems," *IEEE cIrcuIts and systEms magazInE*, vol. 13, no. 2, pp. 74–88, 2013.

[14] A. Velasquez and S. K. Jha, "Parallel computing using memristive crossbar networks: Nullifying the processor-memory bottleneck," in *2014 9th International Design and Test Symposium (IDT)*. IEEE, 2014, pp. 147–152.

[15] S. Chakraborti, P. V. Chowdhary, K. Datta, and I. Sengupta, "BDD based synthesis of boolean functions using memristors," in *Design & Test Symposium (IDT), 2014 9th International*. IEEE, 2014, pp. 136–141.

[16] A. U. Hassen, D. Chakraborty, and S. K. Jha, "Free binary decision diagram-based synthesis of compact crossbars for in-memory computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 622–626, 2018.

[17] Z. Alamgir, K. Beckmann, N. Cady, A. Velasquez, and S. K. Jha, "Flow-based computing on nanoscale crossbars: Design and implementation of full adders," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1870–1873.

[18] D. Chakraborty and S. K. Jha, "Automated synthesis of compact crossbars for sneak-path based in-memory computing," in *Design Automation and Test in Europe (DATE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 770–775.

[19] Z. Pajouhi and K. Roy, "Image edge detection based on swarm intelligence using memristive networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1774–1787, Sep. 2018.

[20] H. Yee, "Perceptual metric for production testing," *Journal of Graphics Tools*, vol. 9, no. 4, pp. 33–40, 2004.