Approximate In-Memory Computing using Memristive IMPLY Logic and its Application to Image Processing

Seyed Erfan Fatemieh[†], Mohammad Reza Reshadinezhad[†], and Nima TaheriNejad[*]
[†] University of Isfahan, Isfahan, Iran
E-mail: {erfanfatemieh, m.reshadinezhad}@eng.ui.ac.ir
[*] TU Wien, Vienna, Austria
E-mail: nima.taherinejad@tuwien.ac.at

Abstract—**Approximate computing is a new way of performing calculations in digital systems. By applying this method, performance metrics, e.g., speed, are improved, but in return for this, the accuracy of the calculations is reduced. Memristors are electrical elements that can be used to perform logical calculations along with data storage. This makes memristors a good choice for In-Memory Computation (IMC). IMPLY logic is the first stateful logic proposed for memristive IMC. Approximate computing in memory, particularly using memristive stateful logic, has not been explored yet. In this paper, we combine these two concepts and propose a novel algorithm for serial IMPLY-based adders to implement an approximate full-adder. The proposed approximate full-adder was assessed in an image processing application, and image quality metrics like Peak Signal to Noise Ratio (PSNR) were calculated. In addition, different error quality metrics like Error Distance (ED) and Mean ED (MED) were assessed. Our study shows that the proposed method can achieve up to 40% improvement whereas maintaining the introduced error in an acceptable range (i.e., a PSNR above 32.4).**

Index Terms—**Full-adder, Approximate Computing, IMPLY, Processing In-Memory, Image Processing**

I. Introduction

One of the basic operations in arithmetic is addition. Half-adder and full-adder are the basic cells for implementing addition in digital circuits. Approximate computing is a promising solution for reducing circuits complexity, and as a result, energy consumption, speed, and area factors improved, but this novel computation method can be applied only in error-resilient applications such as image processing [1]. By applying approximate computing, image quality is reduced due to the reduced computational accuracy. Therefore, image quality metrics such as PSNR, Structural Similarity Index (SSIM), and Mean SSIM (MSSIM) should be used to ensure output quality [1]. In addition, different error quality metrics are introduced to assess the accuracy of calculation, like Error Rate (ER), ED, MED, and Normalized MED (NMED) [2].

In-Memory Computation (IMC) is a new computational method as an alternative to the Von-Neumann computer architecture [3]. Utilizing this architecture tackles problems such as transferring a large amount of data to the processor and reducing energy consumption and processing time [4], [5]. Among various technologies for IMC, memristive technology stands out. Memristors are non-volatile devices that can store data [6]–[10] and perform logical operations [4], [11]–[15], and calculations [16]–[20]. An integrated storage and calculation unit further decreases unnecessary data transfer and alleviates the Von-Neumann bottleneck further [21]. This advantage is more significant when working with big data (e.g., sorting data [22]).

TABLE I: The truth table of IMPLY logic

a	b	$a \rightarrow b = b'$
0	0	1
0	1	1
1	0	0
1	1	1

In this article, a novel memristor-based approximate full-adder is proposed that can be used for IMC. In the next section, the Material Implication (IMPLY) logic and its application in boolean logic with memristors are reviewed. The proposed approximate full-adder is introduced in the third section. The fourth section presents circuit-level simulation results, and a circuit-level comparison is made between the exact memristor-based adder structure and the proposed design in this article. The proposed approximate full-adder was assessed in three different scenarios, and in this section, error analysis metrics were assessed. In the fifth section, the proposed approximate full-adder is assessed in an error-resilient image processing application. Simulation results and image quality metrics reached from each scenario were assessed and reported. We conclude the paper in section VI.

II. IMPLY Logic and Adders

The memristor is an electrical element that can play an important role in the IMC [3], [5]. There are several ways to implement memristor-based digital circuit, as listed in [15] too. This article uses the IMPLY method [11], [12], which is the first stateful logic for memristors proposed by Hewlett Packard (HP). Stateful logic is a logic in which both inputs and outputs are represented by the state of the input and output memristors (as opposed to voltage and current in traditional CMOS). Applying this logic with memristors allows data storage and

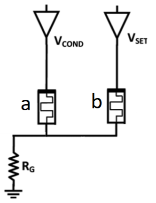

Fig. 1: Circuit implementation of IMPLY logic gate [12].

processing to be done together and in a complementary way and without data ever leaving the array. The truth table of the IMPLY logic is seen in Table I. Ain and Bin are the inputs, and the result of Ain IMPLY Bin (Ain $\rightarrow$ Bin) is stored in Bin. Here, logic zero is equivalent to High Resistance State (HRS), and logic one is equivalent to Low Resistance State (LRS) [23]. The circuit-level implementation of IMPLY is shown in Figure 1. For a correct operation, the following must hold [24]: $R_{on} << R_G << R_{off}$, $V_{COND} < V_C < V_{SET}$, and $V_{SET} - V_{COND} < V_C$, where V_C is the memristor threshold voltage (i.e., the necessary voltage to change the memristor's state).

There are several full-adder designs that use IMPLY logic for their implementation. These adders are implemented using a serial structure [24]–[27], parallel structure [24], [27], or a combination that is neither fully serial nor fully parallel [23], [28], [29]. Among these structures, serial is the simplest one and most compatible with cross-bar architecture. In the serial structure, as shown in Figure 2, all memristors are located on one row (or column). In other structures, memristors are located on different sections (rows or columns) and, when necessary, are connected to each other using external switches. The simplicity and compactness of the serial topology come at the cost of slow calculations since only one IMPLY can be performed at each cycle (no parallelism in operations). In this work, we use the serial topology due to its advantages and tackle its disadvantage by proposing a new approximate algorithm that reduces the number of steps required for the addition.

III. Proposed IMPLY-Based Approximate Full-Adder

In IMPLY logic, if the first operand is 0 and the second operand is 0 or 1, the output is 1. If the first operand is 1, and the second operand is 0, the output is 0. In [26], which is the fastest serial IMPLY-based adder, the authors assessed different solutions to propose Sum and Cout logic of an exact full-adder. The key aspect was to design based on IMPLY properties and in the IMPLY-logic domain as opposed to Boolean logic [26]. In this paper, the same solution is applied to propose the approximate full-adder's outputs. If P= [01011111] and Q= [11001100], P $\rightarrow$ Q is [11101100]. By comparing this output with the exact full-adder's Sum (which is [01101001]), we can see that the proposed output is inexact in three states (AinBinCin="000,101, and 111"). Now, if P= [11101100] and Q= [00000000], P $\rightarrow$ Q is [00010011]. This output is the same as exact Cout in seven states (except in "AinBinCin=101"). The exact full-adder and the proposed approximate full-adder's truth table are shown in Table II. The inexact output cases are specified by ✕ and the exact ones specified by ✓. The proposed Sum is inexact in three states out of eight, and the Cout is exact in seven states out of eight. Hence, the Error Rate (ER= $\frac{\text{Number of incorrect outputs}}{\text{The total number of outputs}}$ [31]) of the proposed full-adder's Sum is $\frac{3}{8}$, and the error rate of its Cout is $\frac{1}{8}$. The proposed memristor-based approximate full-adder's Error Distance (ED= "Arithmetic difference of Exact full-adder's output and approximate full-adder's output" [31]) is 3. The step-by-step implementation of the proposed approximate full-adder in the serial method using two work memristors is listed in Table III.

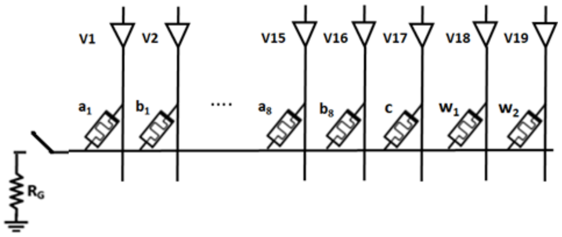

Fig. 2: The serial topology for IMPLY-based adders [30].

TABLE II: The truth table of the exact full-adder and the proposed approximate full-adder

Ain	Bin	Cin	Exact Sum	Exact Cout	Approximate Sum	Approximate Cout
0	0	0	0	0	1 ✕	0 ✓
0	0	1	1	0	1 ✓	0 ✓
0	1	0	1	0	1 ✓	0 ✓
0	1	1	0	1	0 ✓	1 ✓
1	0	0	1	0	1 ✓	0 ✓
1	0	1	0	1	1 ✕	0 ✕
1	1	0	0	1	0 ✓	1 ✓
1	1	1	1	1	0 ✕	1 ✓

TABLE III: Proposed approximate full-adder's algorithm and its steps using IMPLY and FALSE (s1 and s2 are the work memristors)

Step	Operation	Equivalent logic
1	$s1 = 0$	$FALSE(s1)$
2	$s2 = 0$	$FALSE(s2)$
3	$A_{in} \rightarrow s1 = s1'$	$NOT(A_{in})$
4	$B_{in} \rightarrow s2 = s2'$	$NOT(B_{in})$
5	$s1' \rightarrow C_{in} = C'_{in}$	$NOT(A_{in}) \rightarrow C_{in}$
6	$C'_{in} \rightarrow s2' = s2''$	$Sum = \left(\overline{A_{in}} \rightarrow C_{in}\right) \rightarrow \overline{B_{in}}$
7	$s1 = 0$	$FALSE(s1)$
8	$s2'' \rightarrow s1 = s1'$	$C_{out} = \overline{Sum}$

IV. Simulation and Comparison of Full-Adder

A. Simulations

1) Setup: The functionality of the algorithm of the proposed approximate full-adder was simulated in LTSpice by using the VTEAM model [32]. The values of the parameters we used in this model are shown in Table IV. The values applied for the IMPLY function simulation are

TABLE IV: VTEAM model and setup values [33]

Parameter	v_{off}	v_{on}	α_{off}	α_{on}	R_{off}	R_{on}
Value	0.7 V	-10 mV	3	3	1 MΩ	10 kΩ

k_{on}	k_{off}	w_{off}	w_{on}	w_C	a_{off}	a_{on}
-0.5 nm/s	1 cm/s	0 nm	3 nm	107 pm	3 nm	0 nm

[33]: $\{V_{SET}, V_{COND}, V_{RESET}, R_G,\ t_{pulse}\}$ = $\{1V,\ 900mV$,-$5V,\ 40k\Omega,\ 30\mu s\}$.

B. Simulation Results

Based on the algorithm of Table III, the initialization of work memristors is done in a single cycle. In the sixth step (120us-150us), the Sum value and in the eighth step (180 μs-210 μs), the Cout value are calculated. All cases produced the correct approximate output as listed in Table II. The output waveform of the proposed approximate full-adder for two sample inputs (AinBinCin="000,110") is shown in Figure 3. That is, one case of the expected approximate (inexact) and one case of expected exact output, respectively. The simulation results confirm the correctness of the proposed algorithm.

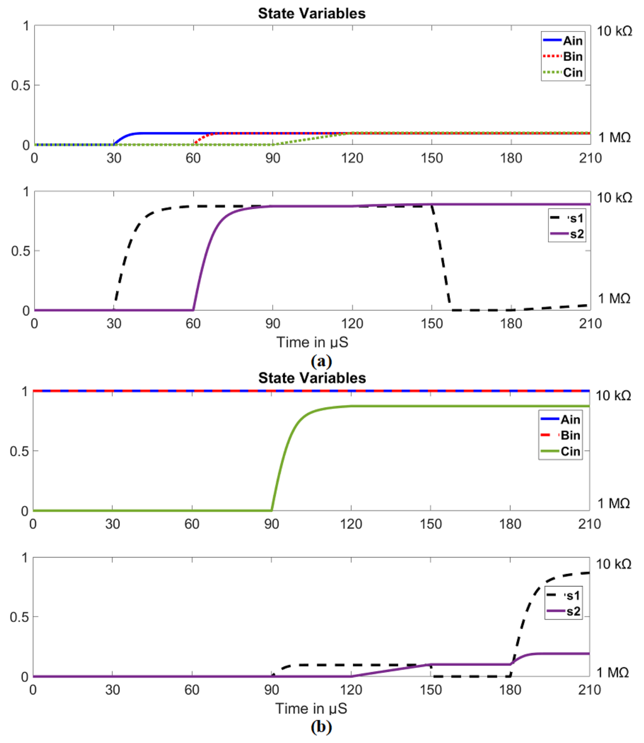

Fig. 3: Proposed approximate full-adder's simulation outputs: (a) AinBinCin="000"-Sum='1'-Cout='0', (b) AinBinCin="110"-Sum='0'-Cout='1'.

C. Comparison

In this section, a comparison is made between the proposed approximate adder and the fastest exact adder using serial structure [26]. The proposed approximate full-adder can be employed in an 8-bit ripple carry adder structure in combination with exact full-adders; first, the five most significant full-adders are exact, and the three least significant ones are approximate (**Scenario 1**). Then four most significant bits were calculated using exact full-adders, and the other four were calculated by applying approximate full-adders (**Scenario 2**). In the last scenario, the three most significant full-adders were exact, and the other five were inexact (**Scenario 3**). The number of steps and the accuracy of computations increase by increasing the number of exact full-adders in the most significant bits (See Table V, VI).

TABLE V: Comparison between the proposed adders and the State-of-the-Art (SoA) [26]

Algorithm	Number of Steps		Number of Memristors	
	n	n=8-bit	n	n=8-bit
Exact: Serial [26]	22n	176	2n+3	19
Approx.: Proposed	8(n-5)+22(n-3)	134	4(n-5)+2(n-3)+1	23
	8(n-4)+22(n-4)	120	4(n-4)+2(n-4)+1	25
	8(n-3)+22(n-5)	106	4(n-3)+2(n-5)+1	27

TABLE VI: Different scenarios of approximate ripple carry adder and error metrics analysis

Scenario	MED	NMED
1	2.0625	0.004
2	4.3516	0.0085
3	8.8555	0.0174

By applying the proposed approximate full-adder in the scenarios mentioned, the number of steps improved between 24%-40% in the 8-bit ripple carry adder structure. The number of memristors needed in each structure is also inserted in Table V. The number of memristors applied in the proposed circuit is reduced to 2n+3 by re-using the input memristors, which is the same as SoA [26].

This reduction in the number of steps has been achieved by applying approximate computing. Therefore, the proposed structure cannot be applied in all computational applications (the ED is 3), and it should only be applied in error-resistant applications such as image processing. The accuracy of the approximate computations can be examined in the 8-bit carry ripple adder structure. ER and ED were assessed in Section III. There exist other error metrics to assess the errors applied in the mentioned computational structure. MED results from the summation of ED divided by the number of outputs [2], [34], [35]. Normalized MED (NMED) is another error metric that is the normalized value of MED by applying the maximum value of the exact 8-bit ripple carry adder structure [2]. All 65536 possible inputs applied to the 8-bit ripple carry adder structure in MATLAB and the error metrics calculated are shown in Table VI. We see that as the number of exact full-adders increases in the most significant bits of the 8-bit ripple carry adder, the MED and NMED values decrease.

V. Application in Image Processing

Image processing is one of the error-resilient applications that its computational complexity can be reduced by applying approximate computing. Masking an image is a function that can be done using image addition [2]. The computation complexity is reduced if the approximate full-adders are applied in the 8-bit adder architecture for image addition. Not only complexity of computations is reduced, but also the accuracy of computations. Image quality metrics like PSNR, SSIM, and MSSIM are assessed in approximate image processing to evaluate the quality of computation in this error-resilient application [1]. SSIM is an image quality metric that indicates the errors and differences between two same-sized (P_1 and P_2) images [31], [36]. The structure (s), contrast (c), and luminosity (l) comparison are assessed in this metric [31], [36]–[38]. In [2], [31], [36]–[38], the authors explained how to calculate the mentioned image quality metrics in detail.

$$SSIM(P_1, P_2) = s(P_1, P_2) \times c(P_1, P_2) \times l(P_1, P_2) \quad (1)$$

The proposed approximate full-adder is used in an 8-bit ripple carry adder alongside the exact full-adder to assess its functionality in the image addition application. Three scenarios were selected to evaluate image quality metrics in this application. The proposed approximate full-adder applied in three (Scenario 1), four (Scenario 2), and five (Scenario 3) least significant full-adders in the 8-bit ripple carry adder. Two sets of images (256*256 grayscale and 512*512 grayscale [39]) were added by the mentioned ripple carry adder structures and the image quality metrics calculated in MATLAB. The results of the image addition simulation are shown in Table VII (Figure 4) and Table VIII (Figure 5). These simulations were done to assess the applicability of these approximate structures in image addition as an error-resilient application. The simulation results are acceptable in terms of image quality in all three scenarios. That is, their PSNR is above 30dB, which is the common threshold in the literature for acceptable image quality [40]. In the first scenario, the image quality metrics and error metrics were the best compared to the other scenarios. As expected, image quality metrics decreased in the third scenario in comparison to the first scenario. However, as we see in Figure 4 and Figure 5, this decrease in the image quality is not visible to human eyes, which makes this scenario acceptable.

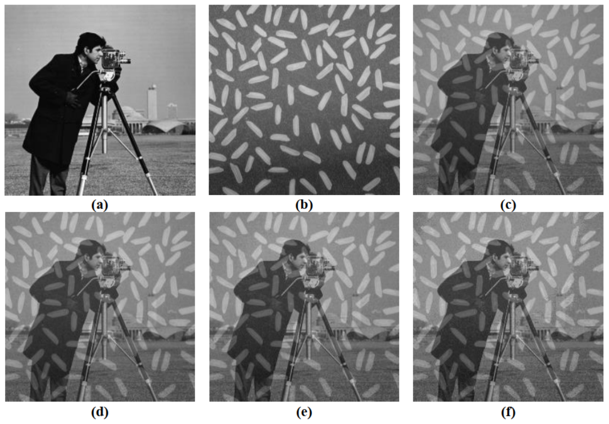

Fig. 4: Three scenarios of approximate image addition: (a) cameraman, (b) rice, (c) exact image addition, (d) Scenario 1, (e) Scenario 2, (f) Scenario 3.

TABLE VII: First image addition simulation and its image quality metrics

Scenario	PSNR (dB)	SSIM	MSSIM
1	44.5148	0.9899	0.99
2	38.67	0.9644	0.9649
3	32.9823	0.8974	0.8996

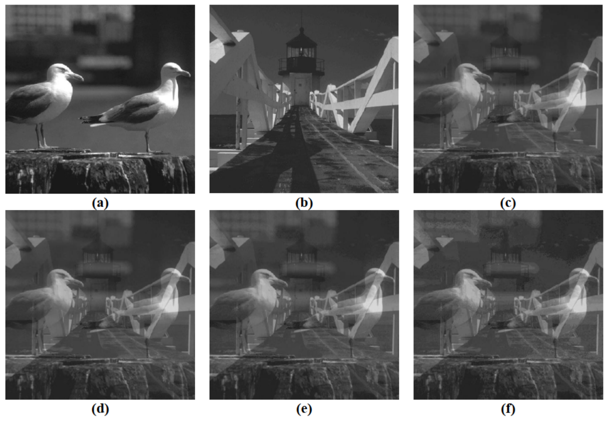

Fig. 5: Three scenarios of approximate image addition: (a) first image [39], (b) second image [39], (c) exact image addition, (d) Scenario 1, (e) Scenario 2, (f) Scenario 3.

TABLE VIII: Second image addition simulation and its image quality metrics

Scenario	PSNR (dB)	SSIM	MSSIM
1	43.0406	0.9887	0.9952
2	38.2047	0.9541	0.9802
3	32.4464	0.8565	0.9268

VI. Conclusion

This paper introduces a new algorithm for an approximate full-adder using the IMPLY logic and serial topology for memristor-based IMC. By applying approximate computing, the number of computational steps was reduced by 24-40%, but it should be noted that the accuracy of the calculations also decreased, which is a common compromise in approximate computing. The ER_{Sum} of the proposed full-adder is $\frac{3}{8}$, the ER_{Cout} is $\frac{1}{8}$, and the ED is 3. The proposed approximate full-adder can be used along with the exact full-adder in error-resistant applications such as image processing. The proposed approximate full-adder was examined in three different scenarios for image addition application. Different error metrics like MED, NMED, PSNR, SSIM, and MSSIM were assessed to examine the results of these three scenarios. The first scenario has the most accurate results compared to the other scenarios, but the number of steps needed was the highest. On the other hand, as expected, scenario 3 as the fastest adder has the largest error. However, the errors introduced due to using this scenario are still undetectable by the human eye and well beyond the minimum required PSNR, which is 30dB (the PSNR of this scenario was 32.4 and above).

References

[1] S. E. Fatemieh, S. S. Farahani, and M. R. Reshadinezhad, "Lahaf: Low-power, area-efficient, and high-performance approximate full adder based on static cmos," *Sustainable Computing: Informatics and Systems*, vol. 30, p. 100529, 2021.

[2] H. A. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 660–665.

[3] S. Haghiri, A. Nemati, S. Feizi, A. Amirsoleimani, A. Ahmadi, and M. Ahmadi, "A memristor based binary multiplier," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2017, pp. 1–4.

[4] S. Gupta, M. Imani, and T. Rosing, "FELIX: Fast and energy-efficient logic in memory," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, November 2018, pp. 1–7.

[5] N. Revanna, L. Guckert, and E. E. Swartzlander, "The future of computing—arithmetic circuits implemented with memristors," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2017, pp. 745–749.

[6] H. Kim *et al.*, "Memristor-based multilevel memory," in *CNNA2010*, Feb 2010, pp. 1–6.

[7] M. Zangeneh and A. Joshi, "Design and optimization of nonvolatile multibit 1T1R resistive RAM," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 8, pp. 1815–1828, Aug 2014.

[8] N. Taherinejad, P. D. S. Manoj, and A. Jantsch, "Memristors' potential for multi-bit storage and pattern learning," in *2015 IEEE European Modelling Symposium (EMS)*, Oct 2015, pp. 450–455.

[9] N. Taherinejad, S. M. P. D., M. Rathmair, and A. Jantsch, "Fully digital write-in scheme for multi-bit memristive storage," in *2016 13th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, Sept 2016, pp. 1–6.

[10] D. Radakovits and N. TaheriNejad, "Implementation and characterization of a memristive memory system," in *2019 IEEE 32nd Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2019, pp. 1–5.

[11] E. Lehtonen and M. Laiho, "Stateful implication logic with memristors," in *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, 2009.

[12] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. Stanley Williams, "'Memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, pp. 873–876, April 2010.

[13] S. Kvatinsky *et al.*, "MAGIC; memristor-aided logic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, Nov 2014.

[14] P. Huang, J. Kang, Y. Zhao, S. Chen, R. Han, Z. Zhou, Z. Chen, W. Ma, M. Li, L. Liu, and X. Liu, "Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits," *Advanced Materials*, vol. 28, no. 44, pp. 9758–9764, 2016.

[15] N. TaheriNejad, "SIXOR: single-cycle in-memristor XOR," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 29, no. 5, pp. 925–935, 2021.

[16] S. Li *et al.*, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *DAC2016*. IEEE, 2016, pp. 1–6.

[17] P.-E. Gaillardon *et al.*, "The programmable logic-in-memory (PLiM) computer," in *DATE2016*. Ieee, 2016, pp. 427–432.

[18] G. Papandroulidakis, I. Vourkas, A. Abusleme, G. C. Sirakoulis, and A. Rubio, "Crossbar-based memristive logic-in-memory architecture," *IEEE Transactions on Nanotechnology*, vol. 16, no. 3, pp. 491–501, May 2017.

[19] M. R. Alam, M. H. Najafi, and N. TaheriNejad, "Exact in-memory multiplication based on deterministic stochastic computing," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2020, pp. 1–5.

[20] M. R. Alam, M. Hassan Najafi, and N. TaheriNejad, "Exact stochastic computing multiplication in memristive memory," *IEEE Design Test*, pp. 1–8, 2021.

[21] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature electronics*, vol. 1, pp. 22–29, 2018.

[22] M. R. Alam, M. H. Najafi, and N. TaheriNejad, "Sorting in memristive memory," 2021.

[23] S. Ganjeheizadeh Rohani, N. Taherinejad, and D. Radakovits, "A semi-parallel full-adder in imply logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 297–301, Jan 2020.

[24] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2054–2066, October 2014.

[25] M. Teimoory *et al.*, "Optimized implementation of memristor-based full adder by material implication logic," in *ICECS2014*, 2014, pp. 562–565.

[26] S. G. Rohani and N. TaheriNejad, "An improved algorithm for IMPLY logic based memristive full-adder," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, April 2017, pp. 1–4.

[27] A. Karimi and A. Rezai, "Novel design for a memristor-based full adder using a new imply logic approach," *Journal of Computational Electronics*, vol. 17, no. 3, pp. 1303–1314, Sep 2018.

[28] K. C. Rahman *et al.*, "Memristor based 8-bit iterative full adder with space-time notation and sneak-path protection," in *MWSCAS2017*. IEEE, 2017, pp. 695–698.

[29] N. TaheriNejad, T. Delaroche, D. Radakovits, and S. Mirabbasi, "A semi-serial topology for compact and fast IMPLY-based memristive full adders," in *2019 IEEE New Circuits and Systems symposium (NewCAS)*, 2019, pp. 1–5.

[30] S. G. Rohani, N. Taherinejad, and D. Radakovits, "A semiparallel full-adder in imply logic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 297–301, 2019.

[31] S. E. Fatemieh and M. R. Reshadinezhad, "Power-efficient, high-psnr approximate full adder applied in error-resilient computations based on cntfets," in *2020 20th International Symposium on Computer Architecture and Digital Systems (CADS)*. IEEE, 2020, pp. 1–5.

[32] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "VTEAM: A General Model for Voltage-Controlled Memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 786–790, August 2015.

[33] D. Radakovits, N. Taherinejad, M. Cai, T. Delaroche, and S. Mirabbasi, "A memristive multiplier using semi-serial imply-based adder," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1495–1506, 2020.

[34] K. Chen, F. Lombardi, and J. Han, "Matrix multiplication by an inexact systolic array," in *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH´15)*. IEEE, 2015, pp. 151–156.

[35] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 4, pp. 1–34, 2017.

[36] H. Waris, C. Wang, W. Liu, and F. Lombardi, "Axsa: On the design of high-performance and power-efficient approximate systolic arrays for matrix multiplication," *Journal of Signal Processing Systems*, vol. 93, no. 6, pp. 605–615, 2021.

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[38] P. Premaratne and M. Premaratne, "Image similarity index based on moment invariants of approximation level of discrete wavelet transform," *Electronics letters*, vol. 48, no. 23, pp. 1465–1467, 2012.

[39] "Dataset of standard 512x512 grayscale test images," https://ccia.ugr.es/cvg/CG/base.htm, accessed: 2021-10-27.

[40] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–33, 2016.