MARADIV: Library of MAGIC-Based Approximate Restoring Array Divider Benchmark Circuits for In-Memory Computing Using Memristors

Chandan Kumar Jha, Sallar Ahmadi-Pour, and Rolf Drechsler

***Abstract*—Among various arithmetic circuits, dividers are one of the costliest in terms of area, energy, and delay. Hence, approximate divider designs have been implemented to reduce their cost for error-resilient applications. However, approximate dividers based on memristors for in-memory computing (IMC) have not received any attention. In this brief, we propose a library of 16 by 8 approximate restoring array dividers suitable for memristor-based IMC that can be used as benchmark circuits. We developed these benchmark circuits by performing an exhaustive design space exploration based on functional approximation. We used the state-of-the-art mapping tool, SIMPLER, to perform the mapping of the divider design to the memristor crossbar. We identified the Pareto optimal approximate divider designs based on the MAGIC design style with varying output qualities. We compare our Pareto optimal designs with the existing state-of-the-art CMOS-based approximate divider designs when mapped to IMCs. We also performed a case study on two image processing applications. Overall we propose MARADIV, a library of MAGIC design style-based approximate restoring array divider benchmark circuits tailored for memristor-based IMC. We believe that MARADIV's benchmark designs can be used for future research in this direction. Hence, all Pareto optimal approximate restoring array divider designs will be made available at https://github.com/agra-uni-bremen/tcasii2022-maradiv-lib.**

***Index Terms*—Approximate computing, approximate restoring array divider, in-memory computing, memristors.**

I. Introduction

MEMRISTORS are two-terminal devices capable of changing their resistance in response to the applied voltage across their terminals [1]. Memristors can either be in a high resistance state (logic 0) or a low resistance state (logic 1), depending upon the magnitude and polarity of the applied voltage across its terminals [2]. With memory bottleneck becoming a severe challenge, IMC is becoming quite popular at easing this issue [3]. Memristors can be used both as a storage and compute element [4], [5]. In this brief, we use memristors to perform digital in-memory computing (IMC). Several design styles are used to perform digital computations using memristors [6], [7], [8], [9], [10], [11]. We used one of the most popular MAGIC design style that maps NOR and NOT function to a memristor crossbar for IMC.

Recently, memristors have been used to design approximate arithmetic circuits [12], [13], [14]. Approximate circuits give significant savings in area, energy, and delay with acceptable output quality in error resilient applications [15], [16]. While approximate dividers have been proposed in literature based on CMOS-based circuits [17], [18], [19], [20], [21], to the best of our knowledge there is no work on approximate dividers using memristors. Hence in this brief, we propose MARADIV, the first library of approximate restoring array divider (RAD) benchmark circuits for IMC using memristor crossbars. In MARADIV, we have approximate RAD with varying output qualities to cater to a wide range of error resilience across applications [22]. Following are the contributions of our work.

- We developed a framework to perform an exhaustive design space exploration of the approximate RAD using MAGIC based design style. We identified the Pareto-optimal approximate divider designs for varying design and error metrics.
- We used the Pareto-optimal approximate RAD designs in two image processing applications namely change detection and background removal.
- To the best of our knowledge, MARADIV is the first library of approximate RAD benchmark circuits using the MAGIC design style.

II. Background and Related Work

In this section, we will discuss the MAGIC design style, the working of RAD, and the existing state-of-the-art approximate RAD designs.

A. MAGIC Design Style

The MAGIC based design style using memristors was proposed in [6]. One of the operations that can be mapped to a crossbar using MAGIC is the NOR operation. The MAGIC NOR gate requires three memristors as shown in Fig. 1. The inputs are stored in M_{in1} and M_{in2}, and the output is stored

Manuscript received 30 November 2022; accepted 23 January 2023. Date of publication 7 February 2023; date of current version 30 June 2023. This work was supported by the German Research Foundation (DFG) within the Project PLiM under Grant DR 287/35-1. This brief was recommended by Associate Editor J. Kulkarni. *(Corresponding author: Chandan Kumar Jha.)*

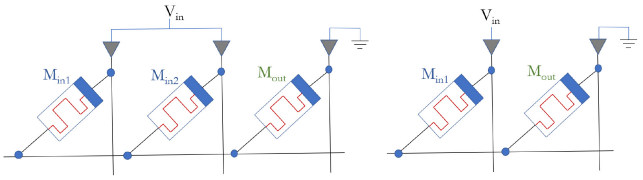
Chandan Kumar Jha is with the Department for Cyber-Physical Systems, German Research Center for Artificial Intelligence (DFKI), 28359 Bremen, Germany (e-mail: chandan.jha@dfki.de).

Sallar Ahmadi-Pour is with the Institute of Computer Science, University of Bremen, 28359 Bremen, Germany (e-mail: sallar@uni-bremen.de).

Rolf Drechsler is with the Institute of Computer Science, University of Bremen, 28359 Bremen, Germany, and also with the German Research Centre for Artificial Intelligence (DFKI), 28359 Bremen, Germany (e-mail: drechsler@uni-bremen.de).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2023.3242976.

Digital Object Identifier 10.1109/TCSII.2023.3242976



(a) NOR (b) NOT

Fig. 1. MAGIC NOR and NOT Gates.

in M_{out}. M_{out} is initialized to logic 1 (low resistance state) before the operation is performed. When M_{in1} and M_{in2} are set to logic 0, and V_{in} is applied, M_{out} remains in logic 1 state as no current flows through M_{out}. For all the other input cases the current flows from the V_{in} through M_{out} and changes its state to logic 0 (high resistance state). Similarly NOT operation can be performed as shown in Fig. 1.

B. Restoring Array Divider

The 8 by 4 RAD is shown in Fig. 2 [23]. The dividend is A[7:0], and the divisor is B[3:0]. The quotient and the remainder are Q[3:0] and R[3:0] respectively. In a *2n* by *n* RAD, to prevent overflow, the *n* most significant bits (MSBs) of the dividend must be lesser than the divisor. The basic block of the RAD is a controlled subtractor (CS) [23]. A CS is a subtractor with a multiplexer at the output. Each row of the divider performs a trial subtraction. Depending upon the value of the MSB of the minuend and the borrow out of the result, a decision is taken to either pass the subtraction output or the minuend to the next stage. The value passed to the next stage is called the partial remainder. The quotient bit is 1, when the MSB of the minuend is 1, or the borrow out of the subtraction is 0, i.e., the result of the subtraction is a positive number. In this case where the subtracted output is passed as the partial remainder. In all the other cases, the quotient bit is 0, and the minuend is passed to the next stage as the partial remainder. This process is repeated for the *n* stages to obtain the quotient and the remainder.

C. Related Work

Approximate RAD designs have been explored in prior works. The approximation is introduced in the CS of the RAD, and the CS is replaced by the approximated inexact controlled subtractor (ICS). In [17], pass transistor logic was used to design three approximate subtractors. The approximation was introduced by removing some of the transistors from the exact subtractor unit. These approximated units were used to design various approximate RAD. Different replacement strategies were proposed as shown in Fig. 2. In [18], three approximate subtractors were proposed. The approximation was introduced in the borrow output. The borrow output was connected to either one of the inputs or its complement. The difference output was also approximated. In [21], four different approximate subtractor designs were proposed. In [19], an approximate controlled subtractor unit was proposed to design the approximate RAD. In [20], three approximate dividers based on CMOS were proposed by minimizing the number of literals in the difference output and no approximation was introduced in the borrow output. All these prior works limit themselves to only a few approximations. We show that as a result of this the prior approximate RAD designs are not optimal and we overcome this by performing an exhaustive functional approximation of the CS.

III. MARADIV Framework

In this section, we discuss the entire framework to perform the design space exploration and obtain the Pareto-optimal set of approximate divider designs. The overall MARADIV framework is shown in Fig. 3.

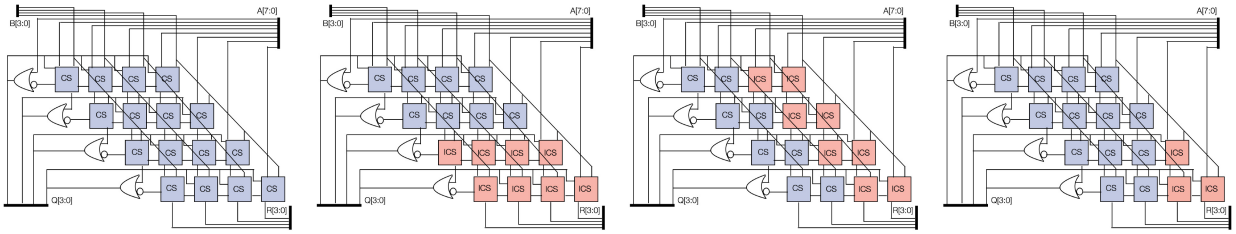
A. Approximate Divider Generation

The approximation is introduced in the CS of the RAD. The CS is replaced by the ICS in RAD, to generate approximate RAD designs. A subtractor has three inputs namely X, Y, and B_{in} and two outputs namely difference (*Diff*) and borrow output (B_{out}). Since the subtractor has three inputs, *Diff* and B_{out} can each be approximated in $2^{2^3} = 256$ ways. Thus, we can then have 256*256 = 65536 combinations of all possible approximations for the CS out of which one will be the same as that of the exact CS. Unlike prior works that have explored some approximations, i.e., three or four subtractor designs, we generated an exhaustive set of 65536 ICS. This allows us to explore larger design space and obtain a better Pareto-optimal design and will be discussed in detail in Section IV.

The CS that is to be approximated can vary. We have used three different *strategies* as also used in prior works for approximation [17]. We have used the row, column, and triangular approximation *strategies* for Type-II as shown in Fig. 2. In this brief, we refer to the number of rows, columns, or size of the triangle as the *type* of approximation. In this brief, we have used three approximation *strategies* (row, column, and triangular) and three *types* (Type-II, Type-IV, and Type-VI) for 16 by 8 dividers. For each *strategy* and *type* of approximation, the CS of the RAD is approximated to generate 65536 approximate RAD designs as shown in Fig. 3 ❶.

B. Synthesis and Crossbar Mapping

We generated the approximate RAD in *verilog*. The *verilog* designs are converted to the Berkeley Logic Interchange Format (.blif) designs as shown in Fig. 3 ❷. We have used the SIMPLER mapping tool to map the.blif designs to the memristor crossbar [24]. SIMPLER maps the.blif designs to a single row of magic NOR and NOT gates as shown in Fig. 3 ❸. We have kept the row size to be 100 for all the designs as SIMPLER was able to map most of the divider designs for this row size. SIMPLER finds the best scheduling and gives the gate count, i.e., number of NOR and NOT operations, as well as the cycle count, i.e., the number of time steps needed for the completion of the division operation. We used the gate count and the total cycles as the design metrics as shown in Fig. 3 ❹.



(a) Exact (b) Row (c) Column (d) Triangular

Fig. 2. Strategies for Type-II Approximation in Restoring Array Dividers.

C. Testbench and Functional Simulation

Error is an important metric in the design of approximate circuits. Since the evaluation of error is done using the functional simulation, we first generated the testbench for each of the approximated divider designs as shown in Fig. 3 ❺. We then generate a trace of 10,000 inputs, keeping the overflow into consideration, sampled from a uniform distribution generated using python NumPy library. We used the *verilator* tool to perform the functional simulation as shown in Fig. 3 ❻. We have used three error metrics namely mean square error (MSE), mean absolute error (MAE), and mean squared log error (MSLE) as shown in Fig. 3 ❼. The equation of the error metrics are given by equations (1) and (2) respectively.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)^2;\ \ MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - x_i|; \quad (1)$$

$$MSLE = \frac{1}{N}\sum_{i=1}^{N}(log_e(1+y_i) - log_e(1+x_i))^2 \quad (2)$$

IV. Pareto-Optimal Design of Approximate RAD

In this section, we discuss the results obtained using the MARADIV framework. We have performed the approximation of 16 by 8 RAD. For all the plots in Fig. 4, 5, and 6, blue circles are the Pareto-optimal obtained using existing works. We have compared our works against the four recent state-of-the-art CMOS-based divider designs as no prior work on approximate divider designs using memristors exist [18], [19], [20], [21]. The smaller gray circles denote the entire design space explored using MARADIV. The Pareto-optimal designs for the row, column, and triangular approximation strategy obtained using MARADIV are shown using brown, green, and red triangles respectively. The overall Pareto-optimal designs are highlighted using the black line. Having a Pareto-optimal design set line lower in the Fig. 4, 5, and 6 compared to existing works, shows that the proposed design set has lower gate count and total cycles for a given error metric.

The results for the Type-II approximation are shown in Fig. 4. The gates count lie between 550 to 850 for most of the designs. The total cycles required for operation lie between 600 to 900 for most of the designs. The MSE, MAE, and MSLE lie between 0.0 - 3.5, 0.0 - 1.6, and 0.0 - 0.012 respectively. The number of Pareto-optimal designs for Type-II approximation for gate count and MSE, MAE, and MSLE are 2694, 2691, and 2935 respectively. The number of Pareto-optimal designs for Type-II approximation for total cycles and MSE, MAE, and MSLE is 2951, 2948, and 3192 respectively.

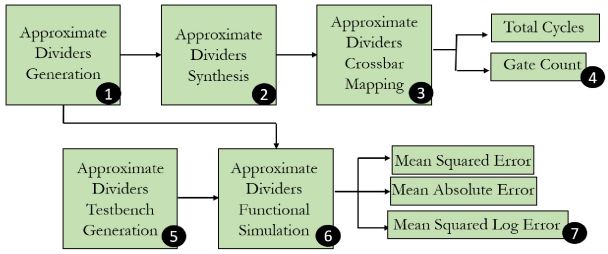


Fig. 3. Overall MARADIV Framework.

The results for the Type-IV approximation are shown in Fig. 5. The gates count lie between 300 to 900 for most of the designs. The total cycles required for operation lie between 300 to 900 for most of the designs. The MSE, MAE, and MSLE lie between 0.0 - 80.0, 0.0 - 8.0, and 0.0 - 0.01 respectively. The number of Pareto-optimal designs for Type-IV approximation for gate count and MSE, MAE, and MSLE are 896, 898, and 1151 respectively. The number of Pareto-optimal designs for Type-IV approximation for total cycles and MSE, MAE, and MSLE is 915, 917, and 1166 respectively.

The results for the Type-VI approximation are shown in Fig. 6. The gates count lie between 100 to 900 for most of the designs. The total cycles required for operation lie between 100 to 900 for most of the designs. The MSE, MAE, and MSLE lie between 0.0 - 1400.0, 0.0 - 35.0, and 0.0 - 0.5 respectively. The number of Pareto-optimal designs for Type-VI approximation for gate count and MSE, MAE, and MSLE are 877, 875, and 1176 respectively. The number of Pareto-optimal designs for Type-VI approximation for total cycles and MSE, MAE, and MSLE is 891, 889, and 1195 respectively.

For all the design and error metrics we see that our Pareto-optimal designs are overall better compared to the state-of-the-art CMOS based designs when mapped to memristors. Overall across all types of approximation for the RAD, we observed that triangular approximation is suitable when least error is required as shown by the red triangles in Fig. 4 - Fig. 6. For applications where error metrics MSE and MAE are important, row approximation is suitable when the application is resilient to larger errors as shown by the brown triangles in Fig. 4 - Fig. 6. For applications where MSLE is important column approximation is suitable for medium errors as shown by the green triangles in Fig. 4 - Fig. 6, while row

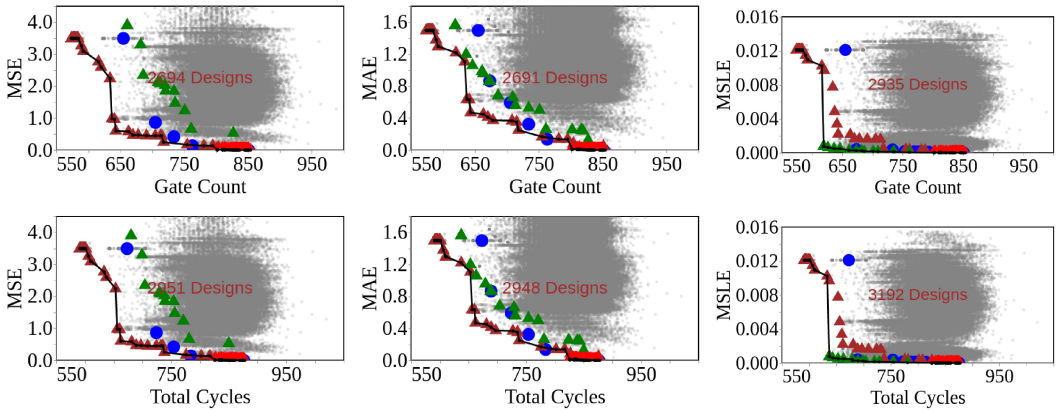


Fig. 4. Pareto optimal designs with respect to various metrics for Type-II Approximation for 16 by 8 RAD.

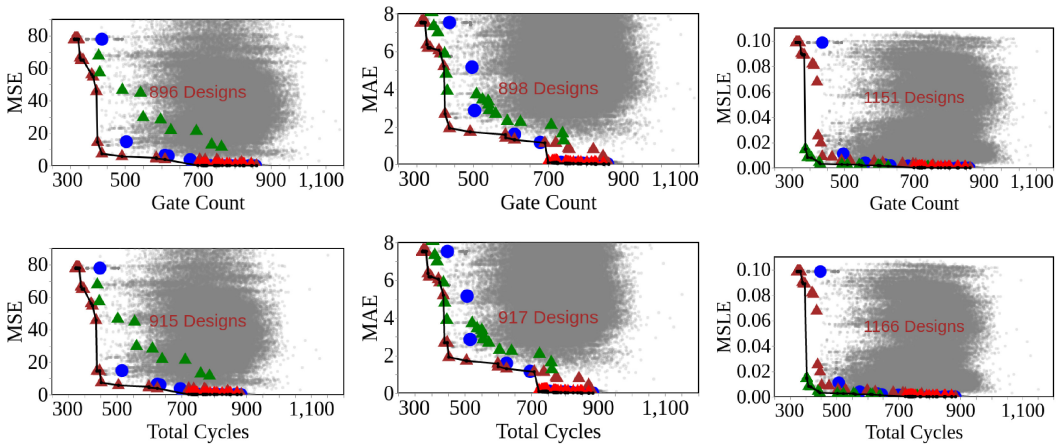


Fig. 5. Pareto optimal designs with respect to various metrics for Type-IV Approximation for 16 by 8 RAD.

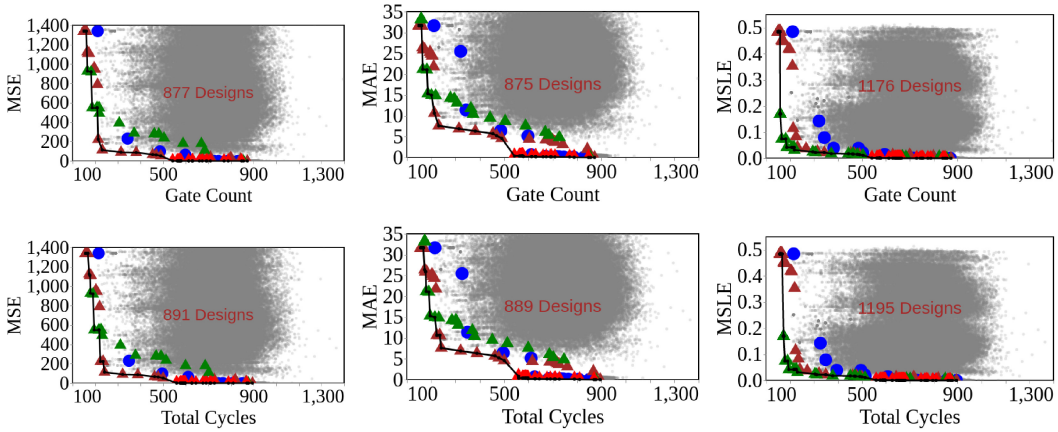


Fig. 6. Pareto optimal designs with respect to various metrics for Type-VI Approximation for 16 by 8 RAD.

approximation is suitable for larger errors as shown by the brown triangles in Fig. 4 - Fig. 6. The designer can select from the Pareto-optimal set depending on the design requirements.

V. Case Study: Applications

In this brief, we have used two applications namely *Change Detection (CD)* and *Background Removal (BR)* similar to prior works. In *CD*, one image is divided by another to highlight the difference in the images as shown in Fig. 7, while in BR the same is done to highlight the foreground of the image as shown in Fig. 8. The first image is multiplied by 64 before division to prevent any overflow as discussed in Section II-B. We perform pixel-wise division of the first image with the second image to obtain the output images in both these applications.

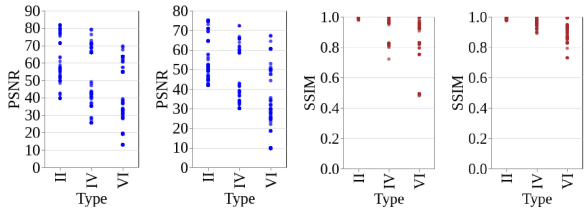
We have used all the Pareto-optimal approximate divider designs from the MARADIV benchmark circuits in these

(a) First (b) Second (c) Output

Fig. 7. Image Set for Change Detection.

(a) First (b) Second (c) Output

Fig. 8. Image Set for Background Removal.



(a) CD (b) BG (c) CD (d) BG

Fig. 9. PSNR and SSIM for CD and BG using Pareto-optimal approximate adder designs.

applications. The output quality metrics used are peak signal to noise ratio (PSNR) and structural similarity index (SSIM). The results of the application using the Pareto-optimal divider designs are shown in Fig. 9. We obtain varying PSNR and SSIM values for the output of the applications using the different types of approximation. Designers can select among these designs depending upon the desired application quality.

VI. Conclusion

In this brief, we have developed the first Pareto-optimal benchmark circuits for approximate RAD for the MAGIC-based memristor design style. We generated all possible functional approximations of the difference and the borrow outputs of the subtractor units in the RAD. We implemented row, column, and triangular replacement strategies for approximation. We then performed the mapping of the approximate divider designs to the memristor crossbar using the state-of-the-art SIMPLER tool. We also analyzed three error metrics that were used to perform the Pareto-optimal analysis. Overall we propose MARADIV, a library of MAGIC based design approximate RAD benchmark designs, that are suitable for IMC. We also showed that MARADIV gives a better Pareto-optimal design set compared to the state-of-the-art approximate RAD designs. We believe this brief will stimulate further research in this direction, hence, MARADIV designs will be made available as open-source.

References

[1] D. Strukov, G. S. Snider, D. Stewart, and R. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, May 2008.
[2] M. Di Ventra, Y. V. Pershin, and L. O. Chua, "Circuit elements with memory: Memristors, memcapacitors, and meminductors," *Proc. IEEE*, vol. 97, no. 10, pp. 1717–1724, 2009.
[3] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," in *Emerging Computing: From Devices to Systems*. Heidelberg, Germany: Springer, 2022, pp. 171–243.
[4] S. Hamdioui et al., "Memristor for computing: Myth or reality?" in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2017, pp. 722–731.
[5] R. Naous et al., "Theory and experimental verification of configurable computing with stochastic memristors," *Sci. Rep.*, vol. 11, no. 1, pp. 1–11, 2021.
[6] S. Kvatinsky et al., "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.
[7] J. Borghetti, G. Snider, P. Kuekes, J. Yang, D. Stewart, and R. Williams, "'Memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, no. 7290, pp. 873–876, 2010.
[8] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL—Memristor ratioed logic," in *Proc. IEEE 13th Int. Workshop Cellular Nanoscale Netw. Appl.*, 2012, pp. 1–6.
[9] J. Rajendran, H. Manem, R. Karri, and G. S. Rose, "An energy-efficient memristive threshold logic circuit," *IEEE Trans. Comput.*, vol. 61, no. 4, pp. 474–487, Apr. 2012.
[10] G. Snider, "Computing with hysteretic resistor crossbars," *Appl. Phys. A Mater. Sci. Process.*, vol. 80, no. 6, pp. 1165–1172, 2005.
[11] L. Guckert and E. E. Swartzlander, "MAD gates' memristor logic design using driver circuitry," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 2, pp. 171–175, Feb. 2017.
[12] S. Muthulakshmi, C. S. Dash, and S. Prabaharan, "Memristor augmented approximate adders and subtractors for image processing applications: An approach," *AEU Int. J. Electron. Commun.*, vol. 91, pp. 91–102, Jul. 2018.
[13] S. Froehlich and R. Drechsler, "Unlocking approximation for in-memory computing with cartesian genetic programming and computer algebra for arithmetic circuits," *Inf. Technol.*, vol. 64, no. 3, pp. 99–107, 2022.
[14] C. K. Jha, P. L. Thangkhiew, K. Datta, and R. Drechsler, "IMAGIN: Library of IMPLY and MAGIC NOR-based approximate adders for in-memory computing," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 8, no. 2, pp. 68–76, Mar. 2022.
[15] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proc. IEEE*, vol. 108, no. 12, pp. 2108–2135, Dec. 2020.
[16] P. Stanley-Marbell et al., "Exploiting errors for efficiency: A survey from circuits to applications," *ACM Comput. Surveys*, vol. 53, no. 3, pp. 1–39, 2020.
[17] L. Chen, J. Han, W. Liu, and F. Lombardi, "On the design of approximate restoring dividers for error-tolerant applications," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2522–2533, Aug. 2015.
[18] K. M. Reddy, M. Vasantha, Y. N. Kumar, and D. Dwivedi, "Design of approximate dividers for error tolerant applications," in *Proc. IEEE 61st Int. Midwest Symp. Circuits Syst. (MWSCAS)*, 2018, pp. 496–499.
[19] E. Adams, S. Venkatachalam, and S.-B. Ko, "Approximate restoring dividers using inexact cells and estimation from partial remainders," *IEEE Trans. Comput.*, vol. 69, no. 4, pp. 468–474, Apr. 2020.
[20] C. Jha and J. Mekie, "Design of novel CMOS based inexact subtractors and dividers for approximate computing: An in-depth comparison with PTL based designs," in *Proc. IEEE 22nd Euromicro Conf. Digit. Syst. Design (DSD)*, 2019, pp. 174–181.
[21] A. Gorantla and P. Deepa, "Design of approximate subtractors and dividers for error tolerant image processing applications," *J. Electron. Test.*, vol. 35, no. 6, pp. 901–907, 2019.
[22] C. K. Jha, S. Singh, R. Thakker, M. Awasthi, and J. Mekie, "Zero aware configurable data encoding by skipping transfer for error resilient applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 8, pp. 3337–3350, Aug. 2021.
[23] A. Gardiner and J. Hont, "Comparison of restoring and nonrestoring cellular-array dividers," *Electron. Lett.*, vol. 7, no. 8, pp. 172–173, 1971.
[24] R. Ben-Hur et al., "Simpler magic: Synthesis and mapping of in-memory logic executed in a single row to improve throughput," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2434–2447, Nov. 2019.