Approximated MAGIC-ReRAM Adder Circuits for Low-Latency In-Memory Computing

Saeideh Nabipour*, Chandan Kumar Jha†, Saeideh Shirinzadeh*‡, Rolf Drechsler*†
* Cyber-Physical Systems, DFKI GmbH, Germany
† Institute of Computer Science, University of Bremen, Germany
‡ Fraunhofer Institute for Systems and Innovation Research (ISI), Karlsruhe, Germany
{saeideh.nabipour, saeideh.shirinzadeh}@dfki.de, {chajha, drechsler}@uni-bremen.de

***Abstract*—Approximate computing improves performance and energy efficiency for error-tolerant applications such as machine learning. Prior work has proposed approximate adder libraries for memristive crossbars using IMPLY and MAGIC stateful logic, primarily focusing on area optimization or fixed crossbar mappings. However, the impact of functional approximation under fully parallel crossbar execution remains largely unexplored. This work presents a framework for generating, mapping, and evaluating approximate *Ripple Carry Adders (RCAs)* implemented using MAGIC logic in memristive ReRAM crossbars under fully parallel crossbar execution. We explore a large design space by generating 458,752 approximate 8-bit RCA variants. Each design is synthesized into NOR/NOT logic and mapped onto a MAGIC crossbar at the micro-operation level. The resulting implementations are evaluated in terms of latency, memristor count, and functional accuracy using *Mean Squared Error (MSE)* and *Mean Absolute Error (MAE)*. Pareto-optimal designs reveal key trade-offs between latency, area, and approximation error, highlighting the potential of MAGIC-based in-memory arithmetic for low-latency and energy-efficient computing.**

***Index Terms*—Approximate computing, in-memory computing, MAGIC design style, pareto-optimal, ripple carry adders**

I. INTRODUCTION

Modern big data and *Artificial Intelligence (AI)* workloads process large data volumes, leading to frequent memory–processor transfers and exposing the Von Neumann bottleneck, where limited memory bandwidth restricts performance [1]. In-memory computing with memristors addresses this issue by enabling logic operations directly within memory. Memristor crossbar architectures combine storage and computation, reducing data movement, latency, and energy consumption while enabling parallel processing [2]. Stateful logic performs computation inside memristor arrays using resistance states to represent data and logic. Several logic styles have been proposed, including *Memristor-Aided loGIC (MAGIC)* and *Material Implication (IMPLY)* logic [3], [4]. Among them, MAGIC is widely used because it implements universal Boolean functions using only NOR and NOT operations [5]. Many emerging applications can tolerate small computational inaccuracies in exchange for improved efficiency. This observation has motivated approximate computing, which relaxes exact computation to reduce power consumption, improve performance, and lower hardware cost [6]. Within this paradigm, arithmetic units have attracted significant attention because they dominate computational cost in many systems. Although approximate computations are widely studied in CMOS implementations [7]–[11], memristor-based approximate circuits remain relatively limited, with only a few works using the MAGIC logic style.

Approximate adders for *Logic-in-Memory (LiM)* using the MAGIC design style have recently attracted growing attention. The IMAGIN framework [12] introduced an initial library of approximate adders based on IMPLY and MAGIC logic, exploring the design space and evaluating designs in terms of energy, runtime, memristor count, and error metrics. Later, [13] proposed an input-distribution-aware library that identifies Pareto-optimal designs for different input distributions using the SIMPLER MAGIC tool [14]. A key difference lies in the mapping strategy. IMAGIN derives Pareto-optimal designs by minimizing the number of memristors, improving area efficiency but often leaving parts of the crossbar underutilized. The distribution-aware approach uses the SIMPLER MAGIC tool [14], which maps adders more flexibly to a given crossbar size and improves resource utilization. In contrast, our work employs a fully parallel mapping strategy using a state-of-the-art MAGIC-based tool [15]. This approach exploits crossbar-level parallelism at the micro-operation level, significantly reducing latency while enabling systematic evaluation of error and resource trade-offs. In this work, we adopt the circuit level approximation technique called functional approximation, where the Boolean function of an exact circuit is modified to satisfy design and error constraints [12]. The main contributions of this work are:

- We perform a large-scale exploration of MAGIC-based approximate 8-bit *Ripple Carry Adders (RCAs)*, generating 458,752 functionally approximated designs and analyzing their design and accuracy characteristics.
- We employ a parallel MAGIC synthesis and mapping flow with row-wise crossbar execution, enabling efficient extraction of total cycles and memristor count for each design at the micro-operation level.
- We evaluate all designs using Verilator-based functional simulation to compute MAE and MSE metrics, and derive Pareto-optimal sets across latency, area, and error to highlight the accuracy–cost trade-offs.

979-8-3315-8229-6/26/$31.00 ©2026 IEEE

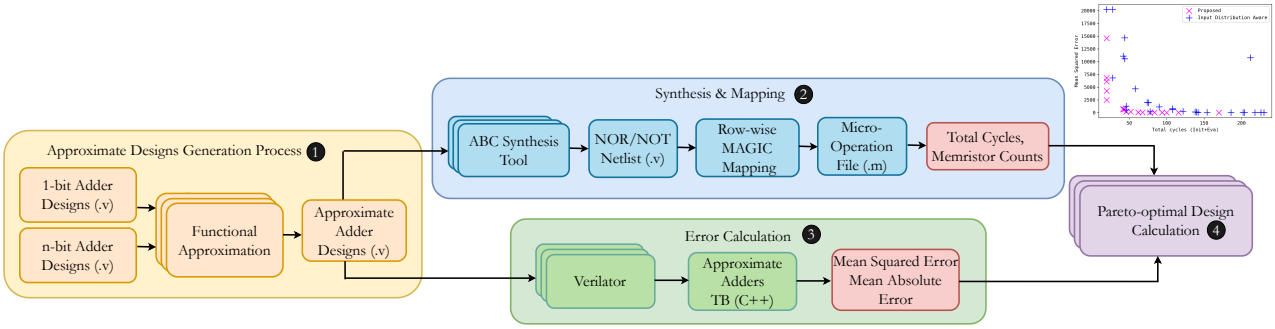


Fig. 1. Overall Framework for MAGIC-based Approximate Adder Generation

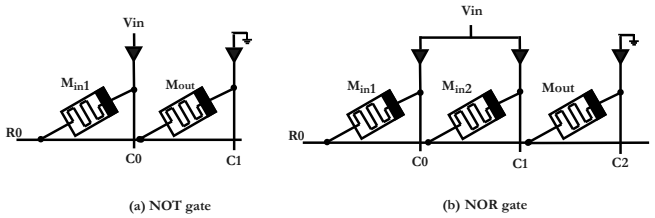


Fig. 2. Realization of NOT and 2-input NOR gate using MAGIC Design

TABLE I
PARETO-OPTIMAL DESIGNS FOR MAGIC-BASED APPROXIMATE 8-BIT RCA.

Designs	**Bit Size**	#Mem			L_T		
		Range	**MSE**	**MAE**	**Range**	**MSE**	**MAE**
RCA	8	13–295	18	19	20–170	29	29

A. MAGIC Design Style

MAGIC is a stateful logic design that performs Boolean operations directly in memristor crossbar arrays [3]. Logic values are stored as resistance states, where R_{on} represents logic '1' and R_{off} represents logic '0'. MAGIC natively supports NOT and NOR gates, and since NOR is universal, complex Boolean functions can be implemented using combinations of these operations. Each MAGIC operation consists of two phases: (1) *initialization*, where the output memristor is preset to a specific resistance state, and (2) *evaluation*, where input voltages are applied, and the output memristor switches to the correct logical value. Fig. 2 shows the implementation of NOT and NOR gates in the MAGIC design style.

II. OVERALL METHODOLOGY

This work presents a framework for generating and evaluating approximate adders in MAGIC-based memristor crossbars. As shown in Fig. 1, the methodology includes four steps: (i) approximate adder generation, (ii) synthesis and mapping process, (iii) error calculation, and (iv) Pareto-optimal designs of approximate adders.

A. Approximate Adder Generation

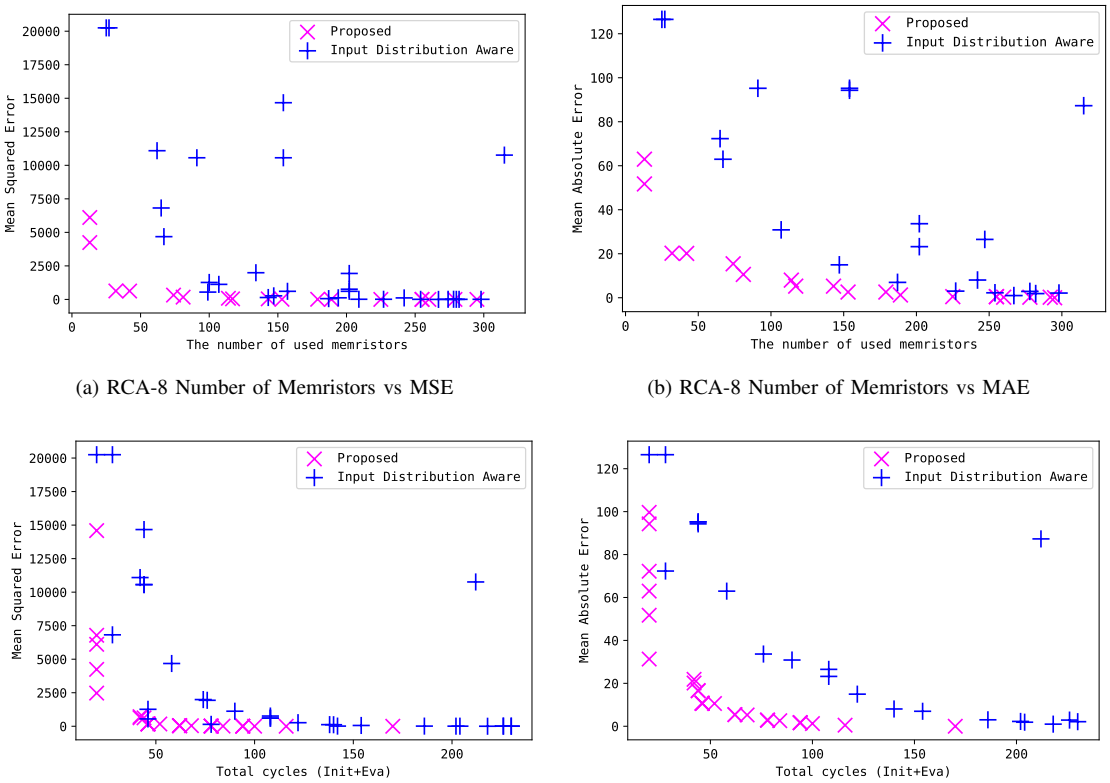
We employ functional approximation, where the Boolean functions of an exact circuit are replaced with simplified alternatives that reduce design cost (e.g., area or latency) while allowing controlled output errors. An n-bit *Ripple Carry Adder (RCA)* consists of n *Full Adders (FAs)*, whose Sum and Carry outputs can be approximated independently (Fig. 1 ❶). Each output can realize $2^{2^3} = 256$ different Boolean functions. Therefore, considering both Sum and Carry outputs together results in a total of 256 $\times$ 256 = 65,536 possible Boolean function combinations [12]. Approximation is applied to the *Least Significant Bits (LSBs)* of the RCA. For the 8-bit RCA, approximations range from 1 to 7 bits, generating 458,752 designs. All circuits were implemented in Verilog.

B. Synthesis and Mapping Process

This section outlines a framework for the synthesis and mapping of 8-bit approximate RCA designs onto MAGIC-based memristor crossbars. During the mapping stage, we employ the MAGIC-based mapping tool proposed in [15]. As depicted in Fig. 1 ❷, all approximate RCA adder designs generated in Verilog are first converted into NOR/NOT netlists using the ABC tool [16]. The mapping tool integrates three main algorithms. First, a level-wise scheduling algorithm named *As Late As Possible (ALAP)* assigns logic gates to their latest possible execution levels based on input dependencies. The scheduled netlist is then mapped onto a memristor crossbar, enabling row-wise parallel evaluation of logic gates at each level. Finally, the associated micro-operations are generated and stored in a `.m` file. The tool reports two primary design metrics: the total number of cycles and the number of required memristors. Since the importance of these metrics can vary across applications, both are evaluated to ensure a comprehensive design-space analysis.

C. Error Calculation

Approximate adders vary in accuracy, so error analysis is essential to assess precision and hardware efficiency. For this purpose, a testbench was implemented for each design (Fig. 1 ❸). An input trace of 10,000 samples was generated using a normal distribution with the Python NumPy library [17]. Functional simulations were performed using Verilator, and



(a) RCA-8 Number of Memristors vs MSE

(b) RCA-8 Number of Memristors vs MAE

(c) RCA-8 Total Cycles vs MSE

(d) RCA-8 Total Cycles vs MAE

Fig. 3. Comparison of all Pareto-optimal designs for 8-bit RCA across the number of memristors/total cycles and MSE/MAE.

the approximate outputs were compared with the exact design to compute the *Mean Squared Error (MSE)* and *Mean Absolute Error (MAE)* [13], [18].

D. Pareto-Optimal Designs of Approximate Adders

Identifying Pareto-optimal designs enables efficient trade-offs between hardware cost and computational accuracy. In our analysis, total cycles and memristor count are used as design metrics, whereas MAE and MSE are used as error metrics [13], [18]. The Pareto fronts were extracted using NumPy and the py-paretoarchive library [17], [19]. Considering two design metrics and two error metrics results in four Pareto analyses for 8-bit RCAs (Fig. 1 ❹), enabling designers to select the most suitable trade-off between cost and accuracy.

III. Performance Evaluation & Discussion

This section analyzes the approximate RCA designs generated using the methodology described in Section II. The Pareto-optimal results for the 8-bit RCAs over normal input distribution, are shown in Fig. 3. In total, 458,752 approximate circuits for the 8-bit RCA were generated. To accelerate synthesis and mapping, we ran the designs in batches of 256 designs per ABC and the MAGIC-based mapping flow. We ran 32 batches of 256 designs in parallel for synthesis and mapping. In the Pareto plots, the x-axis represents a design metric total cycles or memristor count, while the y-axis represents an error metric MAE or MSE. The observed trends and design counts are summarized in Table I.

A. Pareto-optimal Designs

For the 8-bit approximate RCA, Pareto-front analysis reveals consistent accuracy–cost trade-offs across both MSE and MAE evaluations. As shown in Table I, when using the memristor count as the design metric, the design space ranges from 13 to 295 memristors, yielding 18 Pareto-optimal designs under MSE and 19 under MAE. Low-resource designs (13–40 memristors) exhibit large approximation errors, mid-range designs (70–120 memristors) significantly improve accuracy, while high-resource designs ($>$ 150 memristors) provide only marginal additional improvements as the error approaches zero, reaching exact computation with 295 memristors. When total latency is used as the design metric, Pareto-optimal designs span 20 to 170 cycles, generating an identical set of 29 optimal designs for both MSE and MAE. Very low-latency designs (20 cycles) incur high error, intermediate latencies (40–80 cycles) enable substantial reductions in error, and higher latencies ($>$ 100 cycles) yield diminishing improvements until exact computation is achieved at 170 cycles. While our detailed analysis focuses on the 8-bit RCA, the same methodology can be applied to the 16-bit RCA, showing similar trade-off behavior across a larger design space.

B. Comparison with State-of-the-Art MAGIC-based Approximate Adders

We compare the proposed designs with the input-distribution-aware MAGIC adder library generated using SIMPLER in [13]. Although [13] reports Pareto-optimal designs for three input distributions, namely exponential, normal, and uniform, we focus on the normal distribution for comparison, as illustrated in Fig. 3. The proposed method consistently generates designs with significantly lower error for comparable hardware resources. To further illustrate these observations, we discuss one representative plots in detail to highlight key differences between our method and [13], which are also evident in the remaining plots. Fig. 3 (c) shows 8-bit RCA for total cycles versus MSE. Significant differences between the proposed designs and [13] can be observed. For this, the pink $\times$ marker (proposed method) versus blue $+$ marker (proposed method in [13]) can be compared. The proposed method consistently generates more designs with substantially lower error compared to [13]. In particular, under a clock cycle of 100, the proposed method provides multiple designs that achieve more than 50% lower errors (MSE, MAE) than the corresponding designs reported in [13]. Across all error levels, the proposed method achieves comparable accuracy using approximately half the total cycles required. In addition to reducing the required number of total cycles, the proposed method also demonstrates superior hardware efficiency. For the same or fewer number of used memristors, it consistently generates more designs with substantially less error compared to [13]. Overall, the proposed approach demonstrates superior latency and hardware efficiency compared to the state-of-the-art MAGIC-based approximate adder library [13].

IV. Conclusion and Future Work

This work introduces a comprehensive framework for the systematic design and evaluation of approximate RCAs using the MAGIC design style in ReRAM crossbars. Using a functional approximation approach, 458,752 approximate 8-bit RCA designs were generated and synthesized into NOR/NOT netlists. Each design was evaluated through parallel row-wise mapping to estimate latency and memristor count, while functional accuracy was assessed via Verilator simulations using MAE and MSE metrics. Compared to recent MAGIC-based approximate adders, the proposed approach achieves similar or better accuracy with significantly lower design costs. These results highlight the potential of approximate MAGIC-based arithmetic for resource-constrained, low-latency in-memory computing. Future work will extend the framework to other arithmetic units, support diverse input distributions, integrate detailed device and circuit models, and explore hybrid row–column mapping with gate reuse to further improve efficiency.

Acknowledgement

This research has been supported by the German Research Foundation (DFG) with project PLiM (DR 287/35-1, DR 287/35-2, and SH 1917/1-2), and by the German Ministry for Research, Technology and Space (BMFTR) with project ExaVerse (grant number 01IW25003).

References

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. Waltham, MA: Elsevier, 2011.

[2] 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, 2014.

[3] 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.

[4] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "'memristive' switches enable 'stateful' logic operations via material implication," *Nature*, vol. 464, no. 7290, pp. 873–876, 2010.

[5] N. Talati, S. D. Gupta, P. S. Mane, and S. Kvatinsky, "Logic design within memristive memories using memristor-aided logic (MAGIC)," *IEEE Trans. on Nanotechnology*, vol. 15, pp. 635–650, 2016.

[6] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 2013, pp. 1–6.

[7] M. A. Hanif, R. Hafiz, O. Hasan, and M. Shafique, "Quad," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017.

[8] S. Angizi, H. Jiang, R. F. DeMara, J. Han, and D. Fan, "Majority-based spin-CMOS primitives for approximate computing," *IEEE Transactions on Nanotechnology*, vol. 17, no. 4, pp. 795–806, 2018.

[9] J. Lee, H. Seo, H. Seok, and Y. Kim, "A novel approximate adder design using error reduced carry prediction and constant truncation," *IEEE Access*, vol. 9, pp. 119 939–119 953, 2021.

[10] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Rap-cla: A reconfigurable approximate carry look-ahead adder," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 8, pp. 1089–1093, 2016.

[11] F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Block-based carry speculative approximate adder for energy-efficient applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 137–141, 2019.

[12] 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 Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 8, no. 2, pp. 68–76, 2022.

[13] C. K. Jha, S. Ahmadi-Pour, and R. Drechsler, "Input distribution aware library of approximate adders based on memristor-aided logic," in *2024 37th International Conference on VLSI Design and 23rd International Conference on Embedded Systems (VLSID)*, 2024, pp. 577–582.

[14] R. Ben-Hur, R. Rotem, A. Haj-Ali, D. Bhattacharjee, A. Eliahu, N. Peled, and S. Kvatinsky, "Simpler MAGIC: Synthesis and mapping of in-memory logic executed in a single row to improve throughput," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2434–2447, 2020.

[15] S. Nabipour, K. Datta, L. Weingarten, A. Kole, and R. Drechsler, "Multi-input MAGIC synthesis and verification for in-memory computing design," in *2025 IEEE 55th International Symposium on Multiple-Valued Logic (ISMVL)*. IEEE Computer Society, 2025, pp. 178–183.

[16] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 24–40.

[17] EhwFIT, "py-paretoarchive," 2023, available at: https://github.com/ehw-it/py-paretoarchive.

[18] S. Ahmadi-Pour, S. Parvin, C. K. Jha, and R. Drechsler, "Fv-lidac: Formally verified library of input data aware approximate arithmetic circuits," *ACM Transactions on Design Automation of Electronic Systems*, vol. 30, no. 4, pp. 1–23, 2025.

[19] T. Glasmachers, "A fast incremental bsp tree archive for non-dominated points," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2017, pp. 252–266.