High-Speed MAGIC-Based Exact and Approximate Adders for In-Memory Computing

Nandit Kaushik, Fabian Seiler, Nima Amirafshar, Nima TaheriNejad and B. Srinivasu

***Abstract*—As data-intensive applications such as Machine Learning (ML) continue to drive escalating computational demands, traditional approaches face increasing difficulty in meeting performance requirements. Memory-centric computing has emerged as a promising paradigm by enabling computations to be performed directly within memory, thereby reducing the latency and energy overhead associated with data movement. Approximate computing (AxC) offers another complementary approach, improving performance metrics such as speed at the cost of reduced accuracy. In this work, the stateful Memristor Aided loGIC (MAGIC) is employed to integrate these paradigms for the design of exact and approximate full adders, forming the basis for modern computing tasks. The proposed exact MAGIC-based Full Adder (MFA), implemented on a parallel architecture, achieves a 27.7% to 65.7% reduction in latency compared to State-of-the-Art (SoA) exact MAGIC adder, with a minimal 4×5 crossbar footprint. Additionally, a novel approximation methodology is proposed, yielding three MAGIC-based Approximate Full Adder (MAFA) designs, which are integrated with the MFA to construct Ripple Carry Adders (RCAs) with varying degrees of approximation. Compared to the best existing exact MAGIC adder the MAFA designs improve latency by 37.3% to 68.6%, require 19.6% to 49.6% less area, and reduce the energy consumption by 20% to 54.4%. We evaluated the proposed MAFA designs across four image-processing applications. The results demonstrate that they achieve acceptable output quality comparable to the best SoA approximate memristive adders, while reducing latency by up to 78% and improving energy efficiency by up to 35×. Evaluation on CNN models, including ResNet, DenseNet, and VGG families, shows that MAFA-based adders incur a slight accuracy loss while significantly improving latency and energy efficiency, highlighting their suitability for ML workloads.**

***Index Terms*—Memristor, MAGIC, Adders, Approximate computing, IMPLY**

I. Introduction

Arithmetic circuits constitute the foundation of modern processors, performing addition and multiplication operations that are central to workloads including image and video processing as well as Multiply–Accumulate (MAC) operations in neural networks [1], [2]. However, the Von-Neumann architecture faces inherent performance and energy limitations due to frequent data transfers between processing and memory units. With the slowdown of Moore's Law [2], transistors approaching their physical limits [3], and the exponentially growing computational footprint [4], the need for alternative computing paradigms has become increasingly evident. With the growing demands of data-intensive applications in machine learning (ML), Internet-of-Things (IoT), and big data analytics, these limitations have become even more critical. To address these challenges, alternative architectures such as In-Memory Computing (IMC) and Near-Memory Computing (NMC), enabled by emerging device technologies such as memristors [5], [6] and MRAM [7], are being actively investigated. Memristors inherently combine non-volatile data storage, realized through resistive states, with the ability to efficiently perform logic operations [8], particularly through stateful logic schemes such as Material Implication (IMPLY) [9] and Memristor Aided loGIC (MAGIC) [10]. Approximate Computing (AxC) is an emerging computing paradigm that has seen rapid growth in recent years as it presents a viable low-power alternative to traditional computing [1], [2]. The adoption of approximate computing improves key performance metrics, including speed, area, and energy efficiency, at the cost of reduced computational accuracy [11], [12]. For error-resilient applications such as image/video processing or machine learning, approximating certain parts of the computation can yield substantial gains in processing time and energy savings [1], [11], [13].

In this work, we propose an exact MAGIC-based Full Adder (MFA) that enables parallel-bitwise (commonly referred to as parallel in IMC literature [14]) computation and achieves the highest speed efficiency among existing MAGIC-based adders while maintaining an area comparable to the most optimized parallel design [14]. Additionally, we introduce novel MAGIC-based Approximate Full Adder (MAFA) designs, which are integrated within the proposed MFA architecture to construct an approximate 8-bit Ripple Carry Adder (RCA). To the best of our knowledge, this is the first circuit-level implementation of a MAGIC-based approximate RCA employing a parallel topology. The key contributions of this work can be summarized as follows:

1) A high-speed, area-efficient MFA is proposed, requiring 11 computational steps and utilizing 16 memristors in a 4×5 crossbar array, making it the most compact parallel MAGIC-based adder currently in the literature.
2) Three approximation techniques, termed MAFA-1, MAFA-2, and MAFA-3, are introduced. These techniques enable the design of an approximate RCA using the proposed MFA and MAFA architectures, offering improvements in area, latency, and energy efficiency over exact addition.
3) The applicability of the proposed approximate designs has been extensively evaluated in various image process-

Nandit Kaushik is with the Rashtriya Raksha University, Gandhinagar, Gujarat, 382305, India (e-mail: knandit312@gmail.com) Nima Amirafshar, Fabian Seiler and Nima TaheriNejad are with the Institute of Computer Engineering (ZITI), Heidelberg University, 69120 Heidelberg, Germany.(e-mail: {nima.amirafshar, fabian.seiler, nima}@uni-heidelberg.de) B. Srinivasu is with the SCEE, IIT Mandi- 175005, India (e-mail: srinivasu@iitmandi.ac.in)

ing tasks & datasets and multiple CNN architectures for image classification on CIFAR-10.

The rest of the paper is organized as follows. Section II covers the related works. The proposed exact and approximate adders are presented in Section III. Section IV provides simulation results and comparisons. We applied the approximate MAFA designs in four image processing applications in Section V. Further, Section VI evaluates the proposed approximate design through CNN networks for image classification tasks and evaluates the impact of approximation on the accuracy. Section VII concludes the paper.

II. Related Works

A. Memristors and Memristor Aided loGIC (MAGIC)

Memristors store binary information in their resistance states, where a high resistance (R_{off}) and a low resistance (R_{on}) represent logic '0' and logic '1', respectively and provide several benefits over competing cell technologies, including high density, low power consumption, non-volatility, and compatibility with CMOS production [5], [6], [8], [15]. Stateful logic exploits this resistive representation to perform logical operations directly within a memristive crossbar array, using the stored resistance states as both operands and outputs. There exist multiple stateful logic families for memristors such as IMPLY [9], TMSL [16], FELIX [17] SIXOR [18], and the most popular approach in the literature, MAGIC [10].

The MAGIC logic family supports NOR and NOT operations within memristive crossbar arrays. As illustrated in Fig. 1, a MAGIC NOR operation is performed along the rows and columns between the input resistive states A and B, with the result stored in the output memristor, as detailed in [10], [14]. Correct operation relies on appropriate selection of the evaluation and isolation voltages, namely the $Row\ Isolation\ Voltage\ (V_{IR})$ and $Column\ Isolation\ Voltage\ (V_{IC})$, to ensure reliable logic execution and to prevent unintended state changes within the crossbar, as illustrated in Fig. 2(a) and Fig. 2(b), and discussed in [14].

B. MAGIC-based Adders

Two commonly used architectures for memristive circuit design are serial [9] and parallel [14]. Serial architectures perform all computations in a single row/column sequentially, enabling device reuse and resulting in area-efficient designs at the cost of higher latency. In contrast, parallel architectures exploit multiple rows and columns to execute independent operations concurrently, reducing the number of computational steps while increasing the memristor count. Serial MAGIC-based exact adders are reported in [19]–[21]. The most efficient serial design is presented in [21] computing the 8-bit addition in 94 steps by requiring 50 memristors compared to 97 steps and 113 memristors needed in [20], [22]. These designs prioritize area efficiency but incur higher latency due to sequential operation. Parallel MAGIC-based exact adders aim to reduce latency by exploiting bit-wise or block-wise parallelism across the crossbar. Designs introduced in [14], [22], [23] distribute logic operations across multiple rows and

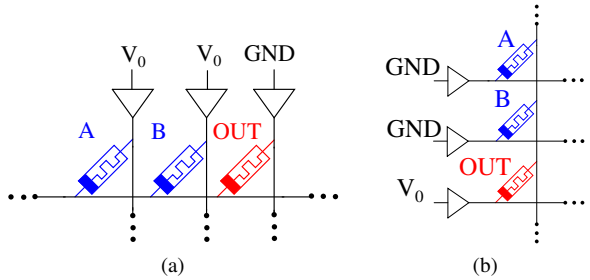


Fig. 1. MAGIC NOR gate : (a) On a row (b) On a column in crossbar.

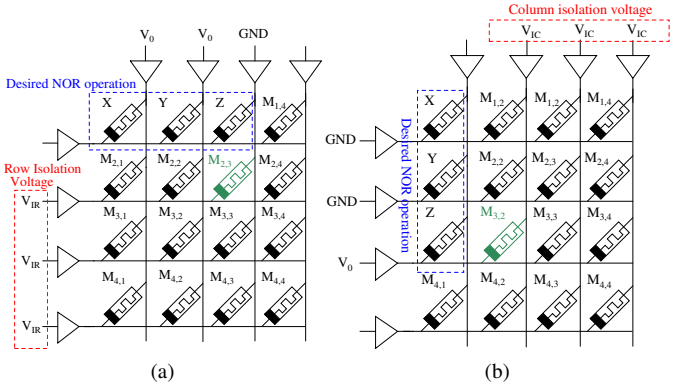


Fig. 2. 4×4 memristor crossbar array (a) Row isolation (b) Column isolation.

columns, aiming to reduce computational steps at the cost of increased memristor usage. For instance, the most efficient parallel design shown in [14], Scheme 2, requires 83 steps and 127 memristors for 8-bit addition.

C. Approximate In-Memory Computing

The approximation of functions leverages the inherent error resilience of applications such as machine learning, image, and video processing. Small errors are introduced by modifying or omitting parts of the underlying logic, which in turn improves latency, area, and energy efficiency [1], [2]. In the SoA, the degree of inaccuracy is measured by error metrics such as the Error distance (ED), Mean Error Distance (MED), Normalized Mean Error Distance (NMED) and Mean Relative Error Distance (MRED) as defined in more detail in [1], [11], [12], [24]. To evaluate image-processing quality, metrics such as the Peak Signal-to-Noise Ratio (PSNR) [25], where values above 30dB are considered acceptable, and the Structural Similarity Index Measure (SSIM) [26], which assesses image similarity on a scale from 0 to 1, are commonly used [11], [12], [27]. There exist multiple CMOS-based approximations [1], [2], [12]. Recently, significant research efforts have focused on approximate adder designs using memristive stateful logic forms such as IMPLY [11], [13], [27]–[30]. Combining CMOS and memristor logic for designing flexible frameworks is explored in [31], [32]. Works in [33], [34] propose libraries of MAGIC-based approximate adders using automated synthesis and scheduling at the logic level. These frameworks operate at a behavioral abstraction and do not provide design-specific metrics. In contrast, the present work focuses on the circuit-level implementation and integration of proposed exact and approximate MAGIC-based adders using parallel architectures.

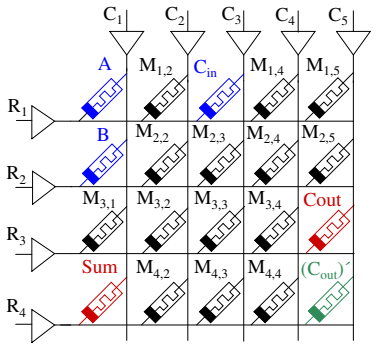


Fig. 3. Proposed MFA implementation in a memristive crossbar.

III. Proposed MAGIC-based Adders

This section proposes the MAGIC-based adders for IMC. First, a MAGIC-based exact full adder is presented, then extended to a multi-bit adder. Later, this section presents the proposed approximate adders.

A. Exact MAGIC-based Full Adder (MFA)

A full adder is proposed using the MAGIC NOR and NOT operations. Considering the full adder with A, B, and C_{in} as the inputs stored in the memristors, the outputs *SUM* and C_{out} can be defined as $A \oplus B \oplus C_{in}$ and $AB + BC_{in} + C_{in}A$ respectively. The outputs of the full adder are thus defined in Equations (2) and (3), respectively, using NOT and NOR logic for MAGIC operations inside a memristive crossbar.

$$X = A \oplus B = \overline{(\overline{\overline{A} + \overline{B}}) + (\overline{A + B})} \quad (1)$$

$$SUM = X \oplus C = \overline{(\overline{\overline{X} + \overline{C_{in}}}) + (\overline{X + C_{in}})} \quad (2)$$

$$C_{out} = AB + XC_{in} = \overline{((\overline{\overline{A} + \overline{B}}) + (\overline{\overline{X} + \overline{C_{in}}}))} \quad (3)$$

To achieve parallel computations and minimize crossbar size, careful input placement is essential. The XOR of inputs A and B (Equation 1) is implemented by mapping them to rows R_1 and R_2, enabling both independent and combined operations in steps 1–3, as shown in Fig. 3 and listed in Table I. Similarly, C_{in} is placed at R_1, C_3 to facilitate its participation in parallel operations during steps 4–5. These placements reduce resource usage and accelerate computation, thereby enabling a compact and high-speed MFA architecture. Fig. 3 shows the overall architecture, illustrating how the mapped inputs collectively realize a compact and efficient MFA design within a 4×5 memristive array. Here, parallelism refers to the bitwise concurrent execution of Steps 1, 2, 4 and 5 within the memristive crossbar, while carry propagation across bits follows a serial ripple-carry structure as discussed in Section III-B. Initializing all the working memristors to logic '1' is necessary before applying the MAGIC voltages, The sequence of the voltages applied to the circuit is described in Table I with their equivalent Boolean logic. Once the crossbar is initialized in two steps, the proposed full adder takes 9 MAGIC computational steps and 16 memristors to compute the sum and carry. The advantages of the proposed full adder are given in Remark 1.

Remark 1: Since the intermediate result of SUM is used in the generation of C_{out}, it helps in reducing the number of computational steps for the full adder. A MAGIC-based full adder proposed in Fig. 3 takes 11 steps and requires a memristive crossbar array of 4×5 (compared to the best existing 13 steps and 4×9 crossbar size [14]).

TABLE I
Proposed MAGIC Operations for Full Adder

Step	Logic Operation	V_0	GND	V_{IR}	V_{IC}
1	$M_{1,2} = \overline{A}$ $M_{2,2} = \overline{B}$	C_1	C_2	$R_3,\ R_4$	-
2	$M_{3,1} = \overline{A + B}$ $M_{3,2} = \overline{\overline{A} + \overline{B}}$	R_3	R_1, R_2	-	C_3, C_4, C_5
3	$M_{3,3} = X$	C_1, C_2	C_3	R_1, R_2, R_4	-
4	$M_{3,4} = \overline{X}$ $M_{1,4} = \overline{C_{in}}$	C_3	C_4	R_2, R_4	-
5	$M_{4,3} = \overline{X + C_{in}}$ $M_{4,4} = \overline{\overline{X} + \overline{C_{in}}}$	R_4	R_1, R_3	-	C_1, C_2, C_5
6	$M_{4,1} = \mathbf{SUM}$ $= (M_{4,3} + M_{4,4})'$	C_3, C_4	C_1	R_1, R_2, R_3	-
7	$M_{4,2} = \overline{\overline{A} + \overline{B}}$	R_4	R_1, R_2	-	C_1, C_3, C_4, C_5
8	$M_{4,5} = \overline{Cout}$ $= (M_{4,2} + M_{4,4})'$	C_2, C_4	C_5	R_1, R_2, R_3	-
9	$M_{3,5} = M_{4,5}$ $= \mathbf{Cout}$	R_3	R_4	-	C_1, C_2, C_3, C_4

B. Proposed MAGIC-based n-bit Adder

The proposed MFA can be extended to multi-bit addition using ripple-carry topology. Table II illustrates a 4-bit RCA constructed using the proposed MFA, with its equivalent MAGIC Boolean operations. Operations such as the NOT of input bits, $A_0 ... A_3$, $B_0 ... B_3$ as well as well as the XOR of $A_i \oplus B_i$, are executed concurrently by mapping their outputs to different rows of the crossbar, enabling bit-parallel execution. Carry propagation and sum generation proceed sequentially across bit positions. While the MAGIC operation sequence remains identical for all bits, the spatial placement of the carry output memristor (C_{out}) is adjusted to maintain locality with the subsequent bit slice. As illustrated in Table II, this results in minor, structured differences in carry placement for successive bits, including parity-dependent shifts, without introducing additional logic primitives or modifying the pulse sequence. In general, an n-bit adder requires initialization step, two parallel steps (steps 1 and 3), and the output can be computed in $7n$ steps. Thus, the overall design requires a total of $7n + 4$ computational steps and utilizes $16n$ memristors. The crossbar size for an n-bit RCA can be estimated using $((4n + \frac{n}{2} - 1) \times 5)$, which simplifies to $(\frac{9n}{2} - 1) \times 5$. The linear scaling of both computational steps and area demonstrates that the proposed approach extends naturally to larger bit-widths through spatial replication of the same MFA mapping.

C. A MAGIC-based Approximate Adder Designs

This section introduces the different MAGIC-based approximate adders, named as MAFA-1, MAFA-2 and MAFA-3, respectively, in Sections III-C1, III-C2, and III-C3.

1) MAFA-1: Equation 2 describes the exact SUM output of the full adder. The majority of computational operations listed in Table I are employed in executing the SUM, primarily because of the XOR between carry in (C_{in}) with the XOR of inputs A and B. In the proposed MAFA-1 approach, the reliance on C_{in} is eliminated, resulting in the proposed

TABLE II
EQUIVALENT MAGIC BOOLEAN OPERATIONS FOR 4-BIT RCA

A_0	$\overline{A_0}$	C_{in}	$\overline{C_{in}}$	
B_0	$\overline{B_0}$			
$\overline{A_0 + B_0}$	$\overline{\overline{A_0} + \overline{B_0}}$	X_0	$\overline{X_0}$	
S_0	$\overline{\overline{A_0} + \overline{B_0}}$	$\overline{X_0 + C_{in}}$	$\overline{\overline{X_0} + \overline{C_{in}}}$	$\overline{C_0}$
A_1	$\overline{A_1}$		$\overline{C_0}$	C_0
B_1	$\overline{B_1}$			
$\overline{A_1 + B_1}$	$\overline{\overline{A_1} + \overline{B_1}}$		$\overline{X_1}$	X_1
S_1	$\overline{\overline{A_1} + \overline{B_1}}$	$\overline{C_1}$	$\overline{\overline{X_1} + \overline{C_0}}$	$\overline{X_1 + C_0}$
		C_1	$\overline{C_1}$	
A_2	$\overline{A_2}$			
B_2	$\overline{B_2}$			
$\overline{A_2 + B_2}$	$\overline{\overline{A_2} + \overline{B_2}}$	X_2	$\overline{X_2}$	
S_2	$\overline{\overline{A_2} + \overline{B_2}}$	$\overline{X_2 + C_1}$	$\overline{\overline{X_2} + \overline{C_1}}$	$\overline{C_2}$
A_3	$\overline{A_3}$		$\overline{C_2}$	C_2
B_3	$\overline{B_3}$			
$\overline{A_3 + B_3}$	$\overline{\overline{A_3} + \overline{B_3}}$	C_3	$\overline{X_3}$	X_3
S_3	$\overline{\overline{A_3} + \overline{B_3}}$	$\overline{C_3}$	$\overline{\overline{X_3} + \overline{C_2}}$	$\overline{X_3 + C_2}$

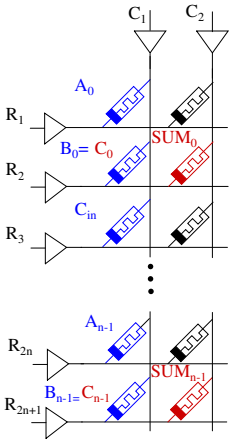


Fig. 4. Proposed n-bit adder using MAFA-1 inside a memristive crossbar.

approximation full adder output as $SUM = \overline{B}$ and $C_{out} = B$. Table III describes the truth table of the proposed approximate full adder, where the approximate output SUM is equally correct (✓) and incorrect (✗) in four cases. Error Distance (ED) is defined as the "Arithmetic difference between the outputs of the exact full adder and the proposed approximate full adder [24]". The total ED for the MAFA-1 is 4 as seen in Table III. The memristive implementation of the proposed approximate n-bit adder in a crossbar array is shown in Fig. 4, and MAGIC operational steps for 1-bit MAFA-1 is described in Table IV. The proposed n-bit MAFA-1 utilizes a crossbar size of $(2n+1) \times 2$. This implementation allows simultaneous generation of SUM and the final C_{out} in a single step, resulting in 2 steps, one for initialization and the other for executing the n-bit approximate addition. The proposed approximate design MAFA-1 takes $3n+1$ memristors.

2) MAFA-2: The approximation logic introduced in MAFA-1 has been specifically developed to optimize the speed and area efficiency. To further enhance the precision of the approximate adder, an additional approximation is proposed,

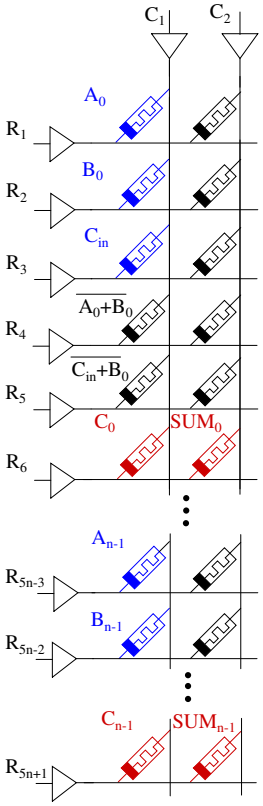


Fig. 5. Proposed n-bit adder using MAFA-2 inside a crossbar array.

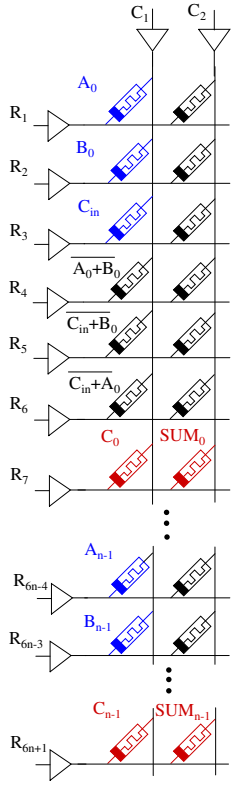


Fig. 6. Proposed n-bit adder using MAFA-3 inside a crossbar array.

referred to as MAFA-2. A NOR logic can be executed in a single step in the MAGIC design principle. Thus, to improve the accuracy, input combination "101" is corrected, which results in outputs, $SUM = \overline{\overline{(A+B)} + \overline{(B+C_{in})}}$ and $C_{out} = \overline{SUM}$. Therefore, the proposed MAFA-2 requires 4 steps as mentioned in Table IV to compute the outputs with 7 memristors. The truth table for MAFA-2 is presented in Table III. It has three incorrect SUM and one incorrect C_{out} output, with a total ED of 3. The approximation logic of MAFA-2 is derived by explicitly optimizing for efficient realization using MAGIC-compatible NOR and NOT operations under memristive crossbar constraints. While the resulting Boolean behavior is comparable to the approximation reported in [35], the derivation and implementation are fundamentally driven by stateful logic requirements. When implemented in a crossbar as depicted in Fig. 5, it requires two steps for initialization and $3n+1$ steps to perform the approximate addition. Thus, it takes a total of $3n+3$ computational steps and $6n+1$ memristors.

3) MAFA-3: An additional approximation technique is suggested to improve the precision of the designed MAFA-2, denoted as MAFA-3. The erroneous output for the input combination ABC="010" in Table III is rectified, resulting in an ED of 2, as indicated in Table III. This results in an extra computational step for executing $\overline{A+C_{in}}$ as shown in Table IV. Hence, the SUM for MAFA-3 is $\overline{\overline{(A+B)} + \overline{(B+C_{in})} + \overline{(A+C_{in})}}$ and C_{out} is $\overline{SUM}$. A n-bit adder using the proposed MAFA-3 inside a crossbar is illustrated in Fig. 6, which requires $4n+3$ computational steps

TABLE III
TRUTH TABLE OF THE PROPOSED APPROXIMATE MAFA DESIGNS

Inputs			**Exact (MFA)**		**MAFA-1**			**MAFA-2**			**MAFA-3**		
A	B	C_{in}	SUM	C_{out}	SUM	C_{out}	ED	C_{out}	SUM	ED	C_{out}	SUM	ED
0	0	0	0	0	1 ✗	0✓	1	1 ✗	0✓	1	1 ✗	0 ✓	1
0	0	1	1	0	1✓	0✓	0	1✓	0✓	0	1 ✓	0 ✓	0
0	1	0	1	0	0 ✗	1 ✗	1	0 ✗	1 ✗	1	1 ✓	0 ✓	0
0	1	1	0	1	0✓	1✓	0	0✓	1✓	0	0 ✓	1 ✓	0
1	0	0	1	0	1✓	0✓	0	1✓	0✓	0	1 ✓	0 ✓	0
1	0	1	0	1	1 ✗	0 ✗	1	0✓	1✓	0	0 ✓	1 ✓	0
1	1	0	0	1	0✓	1✓	0	0✓	1✓	0	0 ✓	0 ✓	0
1	1	1	1	1	0 ✗	1✓	1	0 ✗	1✓	1	0 ✗	0 ✓	1

TABLE IV
PROPOSED MAGIC OPERATIONS FOR 1-BIT MAFAS

Adder	MAFA-1				
Step	Logic Operation	V_0	GND	V_{IR}	V_{IC}
1	$M_{2,2} = \overline{B_0} = S_0$	C_1	C_2	$R_1,\ R_3$	-
Adder	MAFA-2				
1	$M_{4,1} = \overline{A_0 + B_0}$	R_4	$R_1,\ R_2$	-	C_2
2	$M_{5,1} = \overline{C_{in} + B_0}$	R_5	$R_2,\ R_3$	-	C_2
3	$M_{6,1} = \overline{M_{4,1} + M_{5,1}} = C_0$	R_6	$R_4,\ R_5$	-	C_2
4	$M_{6,2} = \overline{M_{6,1}} = \overline{C_0} = S_0$	C_1	C_2	$R_1,\ R_2$ $R_3,\ R_4,\ R_5$	-
Adder	MAFA-3				
1	$M_{4,1} = \overline{A_0 + B_0}$	R_4	$R_1,\ R_2$	-	C_2
2	$M_{5,1} = \overline{C_{in} + B_0}$	R_5	$R_2,\ R_3$	-	C_2
3	$M_{6,1} = \overline{C_{in} + A_0}$	R_6	$R_1,\ R_3$	-	C_2
4	$M_{7,1} = \overline{M_{4,1} + M_{5,1}}$ $+\overline{M_{6,1}} = C_0$	R_7	$R_4,\ R_5$ R_6	-	C_2
5	$M_{7,2} = \overline{M_{7,1}} = \overline{C_0}$ $= S_0$	C_1	C_2	$R_1,\ R_2,\ R_3$ $R_4,\ R_5,\ R_6$	-

and $7n + 1$ memristors. The MAFA-3 design is shaped by MAGIC execution constraints within a memristive crossbar. Although its Boolean behavior resembles the approximation reported in [36], the proposed realization is suited to MAGIC NOR/NOT operations, whereas [36] targets majority-logic-based CMOS implementations.

Remark 2: The primary objective of the proposed MAFA-1 is to develop a design that maximizes efficiency in both area and speed. The design achieves this by employing a single-step computation to approximate the adder output. The subsequent approximation techniques are specifically developed to enhance the accuracy compared to MAFA-1. Specifically, MAFA-2 is designed to improve the ED of MAFA-1 by a magnitude of 1, while MAFA-3 is intended further to enhance the ED by a magnitude of 2. However, this enhancement is accompanied by an increase in latency and area. A n-bit adder using MAFA-1 takes 2 steps to compute the output, whereas MAFA-2 and MAFA-3 take $3n+3$ and $4n+3$ steps, respectively.

IV. SIMULATION RESULTS AND COMPARISONS

This section presents the simulation results and comparisons of the proposed designs with the existing adders. First, the simulations are presented to verify the functionality of the proposed circuits. Then, the proposed circuits are compared with SoA designs in terms of area, the number of steps, and energy consumption. For performance evaluation (in terms

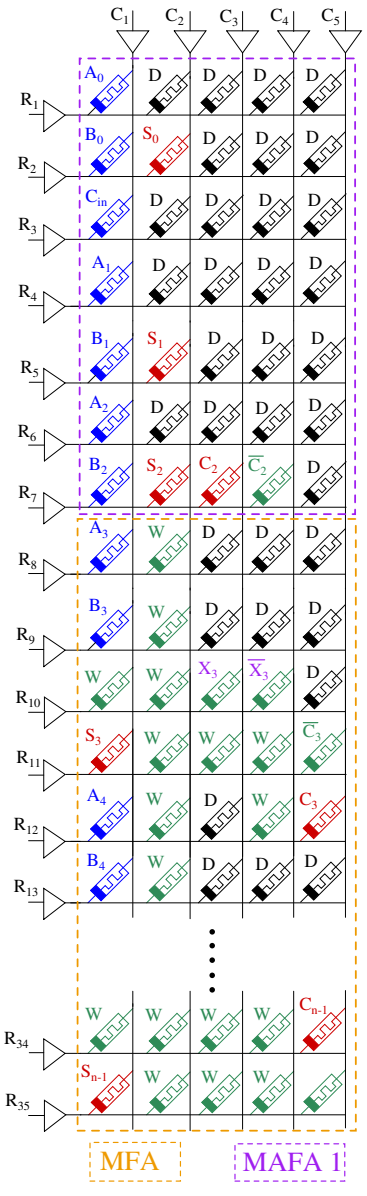
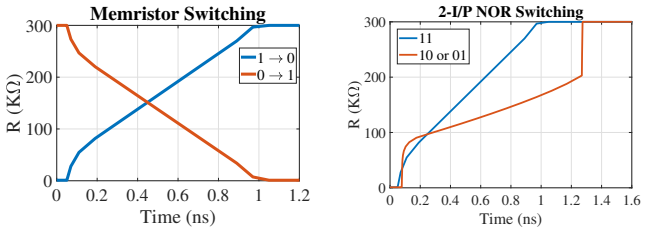


Fig. 7. 8-bit implementation of Outline 1 in a memristive array. MFA denotes the proposed MAGIC Exact Full Adder. **D** indicates the dummy memristors that do not participate in the operations, whereas **W** are the work memristors for performing intermediate operations. Input and output memristors are denoted by blue and red colors, respectively. MAFA-1 can be replaced with MAFA-2 or MAFA-3 for different proposed design implementations.

TABLE V
VTEAM SIMULATION PARAMETER [37]

Parameter	k_{on}	k_{off}	R_{on}	R_{off}
Value	$-216.2\,m/sec$	$0.091\,m/sec$	$1K\Omega$	$300K\Omega$
Parameter	$V_{T,on}$	$V_{T,off}$	x_{on}	x_{off}
Value	$-1.5\,V$	$0.3\,V$	0	$3\,nm$



(a) (b)

Fig. 8. (a) A 2 V pulse switches the state from '0' to '1' and a 1 V pulse resets it back to '0' (b) 2-input NOR gate.

of area and latency) of the approximate adders, an 8-bit RCA is used, where different Least Significant Bits (LSBs) are approximated. Three different conditions are considered, with varying numbers of approximated LSBs. Outlines 1, 2, and 3 approximated k=3, 4, and 5 LSB additions, where k represents the number of adders replaced by an approximate version. The remaining bits use the proposed exact MFA. An 8-bit memristive crossbar implementation for Outline-1 with MAFA-1 is shown in Fig. 7. The crossbar architecture for the remaining outlines, and other approximate adders can be derived along similar lines.

A. Simulation of Proposed MAGIC-based Full Adders

The proposed designs, which are based on MAGIC NOR, are simulated in Cadence Virtuoso using the VTEAM model [37]. The parameters are set similarly to Table V. V_0 as reported in [10], [14] can be in the range of the {0.6 V to 1.5 V} and it is set to 1.2V with $V_{IR} = 290\ mV$ and $V_{IC} = 910\ mV$ in our experiments.

Fig. 8(a) illustrates a single memristor switching with a delay of 1.05 ns for SET and RESET operations, with a 2 V pulse switching the state from '0' to '1' and a 1 V pulse resetting to '0'. These voltages are selected to achieve a 1 ns switching delay to fit into the real-world device as reported in [14]. The MAGIC NOR simulations are shown in Fig. 8(b). For inputs "10" and "01", the circuit reported a maximum delay of 1.27 ns. The functionality of the proposed full adder (MFA) is verified through circuit-level simulations. The memristance of the output nodes after all computational steps for the input combination A = '0', B = '1' and C_{in} = '1' is shown in Fig. 9. After six MAGIC steps, the SUM settles at logic '0', while C_{out} converges to logic '1' after nine steps, confirming correct operation (see Table I). To further validate scalability, simulations are extended to an 8-bit RCA (Outline-1 in Fig. 7). In this configuration, the 3 LSBs are computed using the approximate MAFA-1, while the remaining bits are realized with the exact MFA. The test inputs are A = "10101010", B = "01010101" and C_{in} = '0'. The MAFA-1 block produces its outputs in a single step. In Fig. 10, the individual output memristances of S_1 *to* S_7

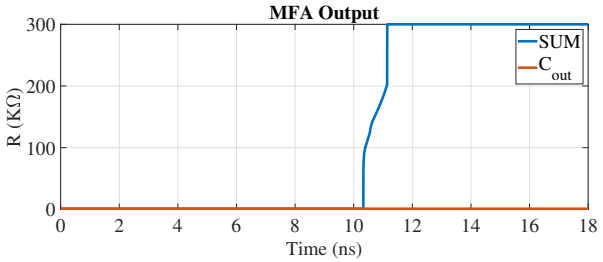


Fig. 9. Full adder output for inputs A = '0', B = '1' and C_{in} = '1', Each MAGIC step denotes 2 ns; Memristance of (a) SUM is high indicating logic '0' (Step 6); (b) C_{out} is low indicating logic '1' (Step 9).

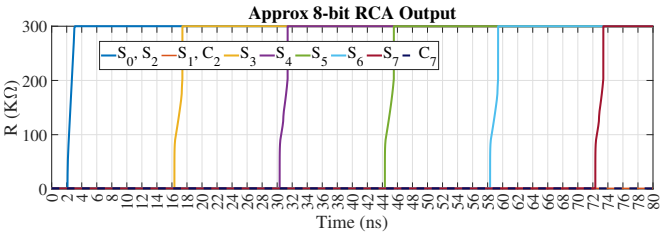


Fig. 10. 8-bit RCA (for Outline 1) output for inputs A = "10101010", B = "01010101" and C_{in} = '0', Each MAGIC step denotes 2 ns; Output of (a) MAFA-1 results in $S_2 - S_0$ as "010" (Step 2) and $C_2 = 1$ (b) MFA results in $S_7 - S_3$ as "00000" and $C_7 = 1$. (After 40 Steps)

and C_7 are superimposed over each other after the execution of computational steps for the 8-bit approximate RCA. As shown in Fig. 10, the approximate sum bits are $S_0 = 0$, $S_1 = 1$, and $S_2 = 0$, with a carry-out $C_2 = 1$. The remaining outputs (S_3–S_7) are generated through the exact MFA. Fig. 10 illustrates the memristance values, where S_3–S_7 changes to 300 $k\Omega$ (logic '0') and C_7 is at 1 $k\Omega$ (logic '1').

Consequently, the complete outputs of the 8-bit RCA are SUM = 00000010 and $C_{out} = 1$. In decimal form, inputs A = 170 and B = 85 give an addition result of 2, compared to the exact arithmetic sum of 255. These simulation outcomes align with theoretical calculations, thereby validating the correctness of the proposed approximate 8-bit RCA.

B. Comparison and Discussion of the MFA Design

In this section, we will compare the proposed exact MFA with SoA exact MAGIC and IMPLY-based adder. Consistent with established practice in MAGIC-based IMC, performance is evaluated using computational steps and memristor count, as these metrics allow technology-independent comparison across architectures. Peripheral and initialization overheads depend on system-level implementations and are therefore not included in the array-level latency results reported here. Table VI shows our design compared to the other MAGIC-based full adders, as reported in the literature [14], [19], [20], [22], [23]. The proposed MFA requires the least number of steps (i.e., 11) and has the smallest crossbar size (i.e., 4×5) compared to the other parallel adder designs. Further gains are achieved by the proposed approximations, which we will cover in Section IV-C.

Latency and Area: The latency and area requirements for n-bit RCA, based on MAGIC and IMPLY-based SoA exact designs [38]–[42] are illustrated in Table VII. We presented the evaluations at $n = 8$, as these structures will be used for the application-level analysis in later sections. Fig. 11(a) depicts

TABLE VI
COMPARISON OF PROPOSED 1-BIT ADDERS

Design	Steps	Memristors	Crossbar Size
Serial Adder [19]	13	15	1×15
SIMPLER [20]	13	15	1×15
Parallel Adder [23]	19	19	7×3
Parallel Adder [22]	35	36	12×3
Parallel Scheme 1 [14]	16	24	4×7
Parallel Scheme 2 [14]	13	15	4×9
Proposed MFA	11	16	4×5
Proposed MAFA-1	2	4	3×2
Proposed MAFA-2	6	7	6×2
Proposed MAFA-3	7	8	7×2

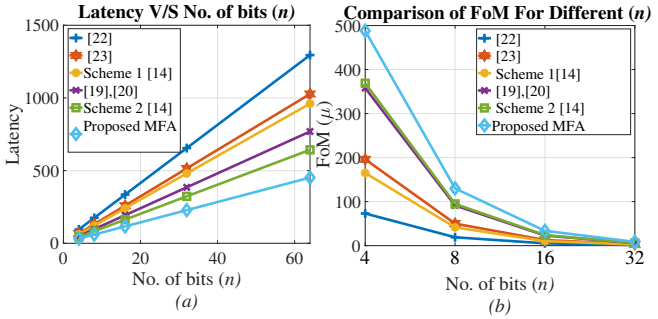


Fig. 11. Comparison of the proposed exact MFA n-bit adder with existing MAGIC-based exact designs (a) Latency (b) Figure-of-Merit (FoM).

the latency of the proposed MFA design compared to other MAGIC-based exact designs, indicating better latency with an increasing n. We additionally report a Figure-of-Merit (FoM), defined in Equation 4, which corresponds to the inverse of the product of p (the number of memristors) and the latency in cycles q, as specified in [43]. A higher value of FoM is preferred.

$$FoM = \frac{1}{p \times q} \tag{4}$$

The FoM over increasing n is shown in Fig. 11(b), where the proposed MFA achieves a meaningful improvement over SoA designs, indicating a better balance between speed and area-efficiency. Fig. 12 depicts the latency gains of the proposed MFA compared to all other exact adder designs in both MAGIC and IMPLY logic. The proposed MFA reduces the latency by 27.7% [14] to 65.7% [22], when compared to exact MAGIC-based designs. A reduction in latency in the range of 56% [41] to 66% [38] is observed for exact designs based on IMPLY. However, MFA requires four more steps, compared to the parallel IMPLY adder from [42].

In terms of area requirements, as observed from Table VII, the proposed MFA utilizes only one additional memristor which adds to just 0.78% more compared to the leading parallel exact adder design in MAGIC [14]. However, the serial adder design outlined in [21] employs the least number of memristors, utilizing 50 compared to the 127 required for the MFA. This is due to the serial architecture's capability to reuse memristors, resulting in reduced memristor requirements. IMPLY-based approaches generally require fewer memristors compared to MAGIC as reported in Table VII. This is attributed to the fact that IMPLY-based designs reuse the input memristors and do not require a dedicated output memristor, as is the case in the MAGIC design style.

TABLE VII
COMPARISON OF PROPOSED n-BIT ADDERS WITH IMPLY AND MAGIC-BASED ADDERS

Design	Number of steps		Number of Memristors	
	n-bit	n = 8	n-bit	n = 8
IMPLY-based Adder				
Serial [38]	$22n$	176	$2n+3$	19
Serial 2 [39]	$20n$	160	$2n+4$	20
Semi-Serial [40]	$10n+2$	82	$2n+6$	22
Semi-Parallel [41]	$17n$	136	$2n+3$	19
Parallel [42]	$5n+16$	56	$4n+1$	33
MAGIC-based Adder				
Adder [22]	$20n+15$	175	$36n$	288
Adder [23]	$16n+3$	131	$19n$	152
Scheme 1 [14]	$15n+1$	121	$25n-1$	199
Adder [19]	$12n+1$	97	$14n+1$	113
SIMPLER [20]	$12n+1$	97	$14n+1$	113
Serial RC [21]	Not Available	94	Not Available	50
Serial SE [21]	Not Available	85	Not Available	50
Scheme 2 [14]	$10n+3$	83	$16n-1$	127
Proposed MAGIC Exact Adder (MFA)	$7n+4$	60	$16n$	128
IMPLY-based Approximate Adder (for k=5)				
SIAFA1,3,4 [13]	$22(n-k)+8k$	106	$2n+3$	19
SIAFA2 [13]	$22(n-k)+10k$	116	$2n+3$	19
SAFAN [28]	$22(n-k)+7k$	101	$2n+3$	19
SAID1 [29]	$22(n-k)+2k$	76	$2(n-k)+2k+3$	19
SAID2 [29]	$22(n-k)+6k$	96	$2n+k+3$	24
SEMIS1 [27]	$4k+10(n-k)+3$	53	$2n+6$	22
SEMIS2, 3, 4 [27], [30]	$5k+10(n-k)+3$	58	$2n+6$	22
SINC [11]	$3k+22(n-k)$	81	$2n+3$	19
SINC+ [11]	$3k+22(n-k)+3$	84	$2n+3$	19
S-PINC [11]	$17(n-k)+3k$	66	$2n+3$	19
S-PINC+ [11]	$17(n-3)+3k+2$	68	$2n+3$	19
S-SINC [11]	$10(n-k)+2k+3$	43	$2n+6$	22
S-SINC+ [11]	$10(n-k)+2k+5$	45	$2n+6$	22
PINC/PINC+ [11]	$5(n-k)+18$	33	$4(n-k)+3k+1$	28
Proposed Approximate Design MAFA-1				
Outline 1 (k=3)	$7(n-3)+5$	40	$16(n-3)+3k+1$	90
Outline 2 (k=4)	$7(n-4)+5$	33	$16(n-4)+3k+1$	77
Outline 3 (k=5)	$7(n-5)+5$	26	$16(n-5)+3k+1$	64
Proposed Approximate Design MAFA-2				
Outline 1 (k=3)	$7(n-3)+3k+5$	49	$16(n-3)+6k+1$	99
Outline 2 (k=4)	$7(n-4)+3k+5$	45	$16(n-4)+6k+1$	89
Outline 3 (k=5)	$7(n-5)+3k+5$	41	$16(n-5)+6k+1$	79
Proposed Approximate Design MAFA-3				
Outline 1 (k=3)	$7(n-3)+4k+5$	52	$16(n-3)+7k+1$	102
Outline 2 (k=4)	$7(n-4)+4k+5$	49	$16(n-4)+7k+1$	93
Outline 3 (k=5)	$7(n-5)+4k+5$	46	$16(n-5)+7k+1$	84

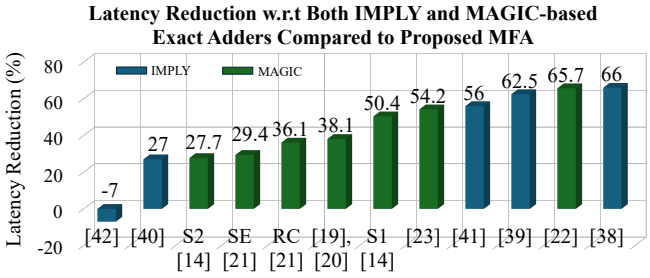
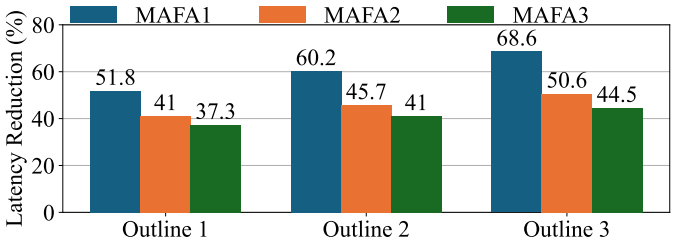


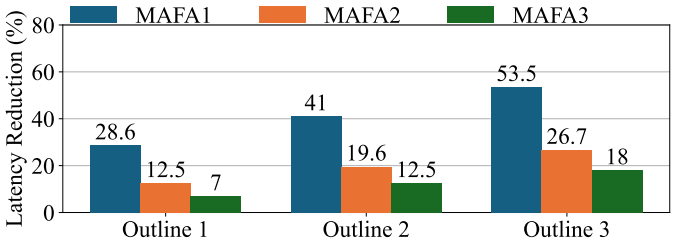
Fig. 12. Latency reduction of the proposed MFA for 8-bit RCA with SoA IMPLY and MAGIC designs.

C. Comparisons and Discussion of the MAFA Designs

As shown in Table VI, the approximate MAFA-1 full adder exhibits superior performance in terms of latency and area, requiring only two steps and a crossbar size of 3×2, respectively. Table VII lists the computational steps and memristor count for an 8-bit RCA, under different Outlines and for approximate



(a) Compared to MAGIC [14]



(b) Compared to IMPLY [42]

Fig. 13. Comparison of latency reduction (%) for 8-bit addition of MAFA's with best (a) MAGIC-based [14] and (b) IMPLY-based [42] exact adders.

IMPLY approaches [11], [13], [27]–[29]. The number of steps and memristors required for different combinations of n and k can be calculated using Equation 5 and Equation 6, where $s_i \in \{0, 3, 4\}$ and $m_i \in \{3, 6, 7\}$ corresponding to each MAFA-i design.

$$\text{Steps}(n, k) = 7(n - k) + s_i k + 5 \tag{5}$$

$$\text{Memristors}(n, k) = 16(n - k) + m_i k + 1 \tag{6}$$

Compared to Exact Adder Designs (Latency and Area): The comparison of the proposed MAFA's is made with the existing best exact adder design in both MAGIC and IMPLY logic, and the results for the percentage reduction in latency are presented in Fig. 13. Compared to the best existing exact adder design in MAGIC [14], the proposed MAFA's demonstrates enhanced speed efficiency in the range of 37.3% to 68.6% across all the Outlines, as illustrated in Fig. 13(a). Similarly, a latency reduction in the range of 7% to 53.5% is observed for the best IMPLY-based exact adder design [42], as depicted in Fig. 13(b).

For the proposed MAFA designs, the area savings are in the range of 29% [14] to 77.7% [22] over the MAGIC-based exact adders, except the serial designs presented in [21]. Compared to the proposed MFA, the proposed MAFA designs exhibit substantial area savings, with a 20% to 50% decrease in memristors, and require 13.3% to 56.6% fewer computational steps, showcasing significant gains in latency and area.

Compared to Approximate Adder Designs (Latency and Area): We illustrate the latency reduction of the proposed MAFA designs compared to SoA IMPLY-based approximate adder in Fig. 14 for $k = 5$. MAFA-1 achieves the highest latency reduction, improving by 21.2% [11] to 77.6% [28]. MAFA-2 shows an improvement ranging from 4.7% [11] to 64.7% [28], while MAFA-3 achieves a latency reduction between 13.2% [27] and 60% [28]. However, MAFA-2 requires more steps than PINC/PINC+, and MAFA-3 requires more than S-SINC, S-SINC+, and PINC/PINC+ designs.

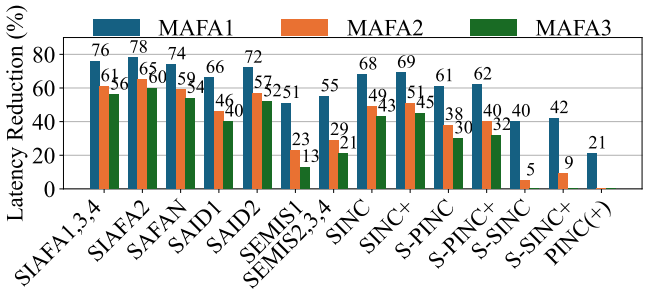


Fig. 14. Latency reduction of the proposed MAFA designs for Outline-3 compared to the SoA approx IMPLY-based adder.

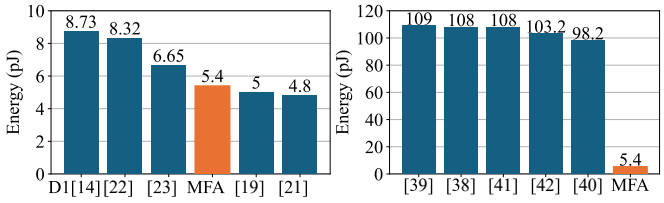
D. Energy Consumption

SPICE simulations are used to evaluate the energy consumption of MAGIC NOR and NOT operations by considering all unique input combinations. It is then averaged to obtain the mean energy per MAGIC NOR/NOT operation. The simulations are performed on a 4×4 array as shown in Fig. 2(a), including the sneak path currents. Using this methodology, the average energy consumption is determined to be 52 fJ for the MAGIC NOT/NOR operation and 613.75 fJ for the IMPLY operation. The total energy consumption of an n-bit addition is subsequently obtained by accumulating this per-operation energy over the total number of operations required for addition, following the approach adopted in prior works [11], [13], [14], [19], [23], [28].

A recent study on energy estimation for MAGIC operations by authors in [44] recommends including the initialization energy for total energy measurements, as it is not negligible, contrary to earlier reported works in [19], [23]. The initialization energy (i.e., *RESET* $\rightarrow$ *SET*) is 280 fJ. The average initialization energy for the proposed MAFA designs for 8-bit addition is 53 pJ. Since prior works only report the execution energy and do not include the initialization energy, we have adopted the same approach to ensure a fair comparison with those works. However, we have included the initialization energy as a distinct energy component for our design. This enables future comparisons with our work while considering the initialization energy. Memristors exhibit negligible static power consumption due to their non-volatile nature [14], [20], [44], [45], the energy reported in this work reflects only switching and initialisation events, which are the dominant contributors in MAGIC-based circuit design. Table VIII lists the energy consumption and Energy-Cycle Product (ECP) of all the proposed approximate designs when employed in an 8-bit RCA under different outlines. The Energy–Delay Product (EDP), commonly used for CMOS-based designs, is less applicable to memristor-based IMC due to the strong technology dependence of absolute delay and energy characteristics. We therefore adopt the ECP, which replaces absolute delay with cycle-level latency and provides a more appropriate metric for comparing memristive approximate computing systems.

TABLE VIII
ENERGY CONSUMPTION (PJ) & ECP (PJCYC) OF THE PROPOSED DESIGNS.

Outline	MFA		MAFA-1		MAFA-2		MAFA-3	
	pJ	pJcyc	pJ	pJcyc	pJ	pJcyc	pJ	pJcyc
1	5.40	324	3.53	141.2	3.84	188.2	4.00	208.0
2			2.91	96.0	3.328	149.9	3.53	186.2
3			2.28	59.28	2.808	114.8	3.06	140.8



(a) MAGIC based exact adders (b) IMPLY-based Exact Adders

Fig. 15. Energy comparison of MAGIC and IMPLY-based 8-bit adders.

Compared to Exact Adder Designs: In Fig. 15(a), the energy consumption of an $n = 8$-bit RCA is compared between the proposed MFA and other MAGIC-based adder designs. The proposed approach consumes 0.6 pJ more energy than the most energy-efficient existing design [21], primarily due to the larger number of parallel computations in the proposed exact architecture. Similarly, Fig. 15(b) compares the energy consumption of IMPLY-based exact adders with the proposed MFA. It can be observed that the IMPLY operation typically consumes more energy than MAGIC, due to the presence of the R_g resistor in IMPLY topologies.

Compared to Approximate Adder Designs: Fig. 16 shows the energy consumption of the IMPLY-based approximate adders with the proposed MAFA's under different Outlines. Here, we have selected the best IMPLY-based energy-efficient approximate designs for comparison. Consequently, the proposed MAFA designs demonstrate better energy efficiency than the IMPLY-based approximation. In particular, the proposed 8-bit RCA under Outlines 1, 2, and 3 for different MAFA designs results in energy savings in the range of 93.5% to 97.5% compared to representative IMPLY-based designs from [13], [27], [28].

Error Metrics: The error metrics introduced in Section II.B were computed in Python by evaluating all 65,536 input combinations of the approximated 8-bit RCA. The results for Outlines 1, 2, and 3 are summarized in Table IX.

We highlight the performance trade-off between accumulated error and the ECP of the proposed designs compared to

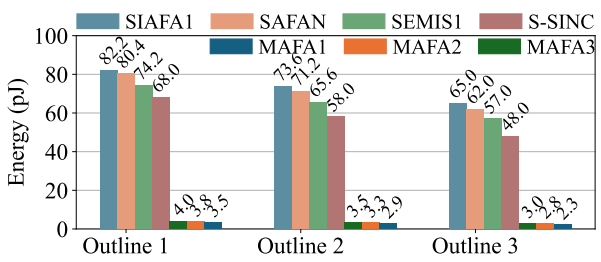


Fig. 16. Energy comparison of MAFA designs to representative SoA approximate adder [11], [13], [27], [28].

TABLE IX
ERROR METRICS FOR PROPOSED APPROXIMATE ADDER IN 8-BIT RCA

Design	MAFA-1		MAFA-2		MAFA-3	
Outline	MED	MRED (%)	MED	MRED (%)	MED	MRED (%)
1	2.625	1.45	2.25	1.25	1.718	0.97
2	5.312	2.98	4.468	2.25	3.617	2.09
3	10.656	6.09	8.912	5.13	7.376	4.43

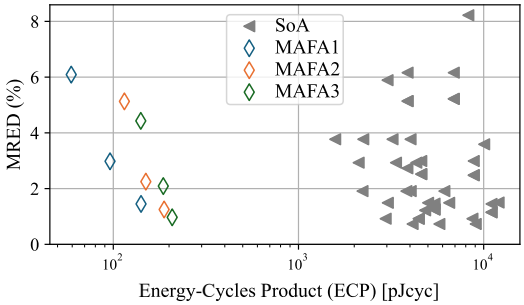


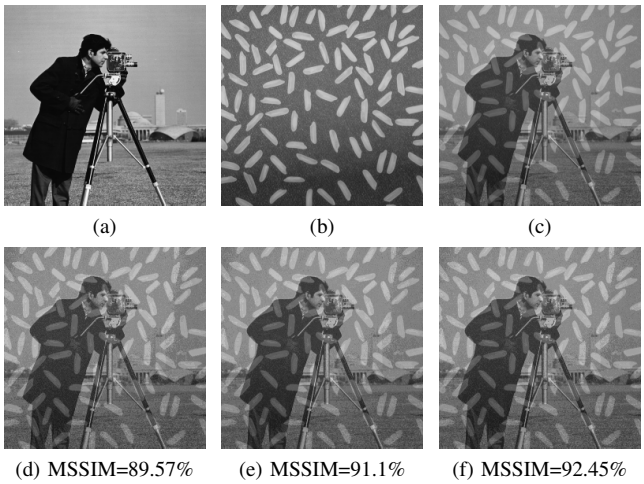
Fig. 17. Comparison to the SoA in terms of MRED over ECP.

SoA approximations in Fig. 17. The results indicate that the proposed designs achieve significantly lower energy consumption and latency while maintaining a similar MRED distribution to existing approaches. This improvement is enabled by the parallel architecture adopted in this work, which reduces latency by up to 76% and energy by up to 97.5% than IMPLY-based designs, at the cost of increased memristor usage. Such a trade-off is well suited for latency and energy-sensitive IMC applications discussed in Section V.

V. APPLICATION IN IMAGE PROCESSING

In this section we analyze the impact of the proposed approximations using four commonly used image processing applications. For evaluating the designs in an image processing context, this work uses the Kodak dataset [46] and the No Carry (NC) dataset [11]. For image subtraction we used the dataset from [47]. The quality of the resulting images are evaluated through PSNR and MSSIM, following the example of [11], [13], [27]–[29]. An overview of the average quality metrics for all applications within the NC dataset [11] is shown in Table X. As we evaluated multiple images for all tasks on two different dataset, we additionally illustrate the distribution of the resulting MSSIM values for all outlines in Fig. 22.

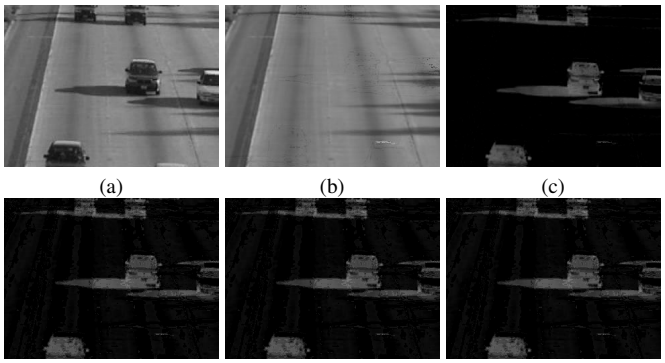
1) Image Addition: Image addition is a common application that is often used for image masking. The addition of two images is achieved by adding the respective pixels of each image. The individual additions are performed by adding the pixel values with the carry-in set to zero and halving the result to limit the output to 8 bits. We calculated 100 additions with random parings for the NC [11] and Kodak [46] datasets and noted the average quality and the corresponding standard deviation. Additions are first performed through an exact adder and then through the various approximation Outlines. Standard grayscale images of size 256×256 [11], shown in Fig. 18, are used to illustrate exact and approximate addition for Outline 3. It is noticeable that as the ED changes from 4 to 2 in MAFA-1 to MAFA-3, the MSSIM values increases. All MAFA designs



(a) (b) (c)

(d) MSSIM=89.57% (e) MSSIM=91.1% (f) MSSIM=92.45%

Fig. 18. Addition of two 256×256 images; (a) Image 1 (b) Image 2 (c) Exact addition; Outline 3 results for (d) MAFA-1 (e) MAFA-2 (f) MAFA-3.



(a) (b) (c)

(d) MSSIM=46.36% (e) MSSIM=47.95% (f) MSSIM=90.23%

Fig. 19. Subtraction of two 320×240 images [47] for motion detection; (a) Image 1 (b) Image 2 (c) Exact addition; Outline 3 approximation result for (d) MAFA-1 (e) MAFA-2 (f) MAFA-3.

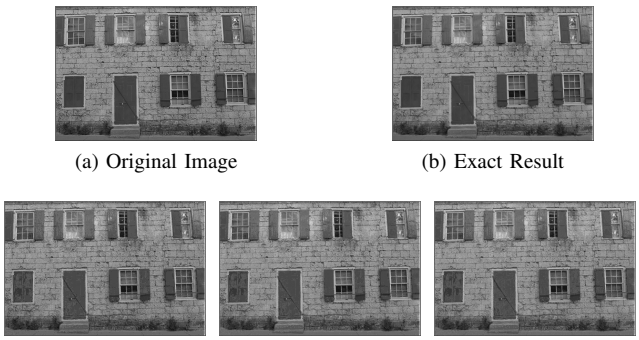
consistently achieve an average MSSIM above 85% and PSNR exceeding 30 dB, indicating acceptable image quality.

2) Image Subtraction: Motion detection is another application commonly used for surveillance [48]. This is accomplished by performing a subtraction on two images in a sequence, which can be achieved through an addition using 2's complement. Similarly to addition this calculation is performed for each pixel pair. We evaluated each outline on the "highway" dataset as proposed in [11]. Some examples are shown in Fig. 19, where it is noticeable that MAFA-3 drastically improves over MAFA-1,2 in terms of MSSIM.

3) RGB to Greyscale: Converting a color image to its greyscale version is a vital task often used for pre-processing. In practice, this conversion is performed by applying perceptual weights to the individual color channels to account for the human visual system's differing sensitivity to red, green, and blue, yielding $Y = 0.299R + 0.587G + 0.114B$. Each weighted channel value is truncated to its integer representation before the grayscale value is obtained by summing the weighted integer components using two addition operations per pixel. An example image from the Kodak dataset [46] that is converted using our MAFA designs is illustrated in Fig. 20. For Outline 3, the PSNR drops below the 30 dB threshold, but

(a) Kodim 19 (b) Exact (c) 76.63% (d) 82.21% (e) 84.25%

Fig. 20. RGB to Greyscale Conversion of Image 19 in Koadak Dataset:(a) Color Image (b) Exact greyscale Image; Approximate outputs through Outline-3 (c) MAFA-1 (d) MAFA-2 (e) MAFA-3; MSSIM is given with images.



(a) Original Image (b) Exact Result

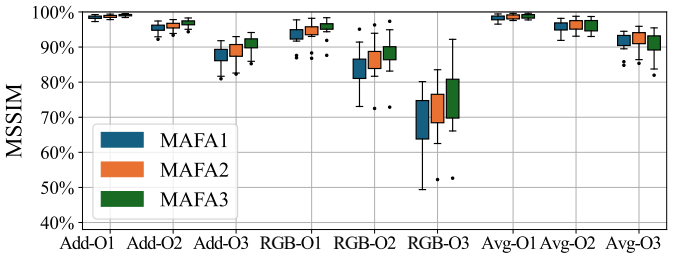
(c) MSSIM=90.2% (d) MSSIM=92.2% (e) MSSIM=91.8%

Fig. 21. Average Pooling operation performed on a Kodim1 image [46]. Results of Outline 3 for (c) MAFA-1 (d) MAFA-2 (e) MAFA-3.

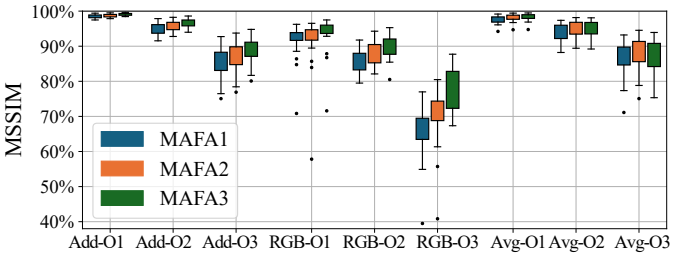
an MSSIM above 70% is achieved for every design besides MAFA-1. Experiments show that the grayscaling application exhibits input dependence. Reversing the operand order (e.g., $B+R$ instead of $R+B$) improves MAFA-1 by approximately 3 dB in PSNR and up to 12% in MSSIM. However, changing the application methodology would disadvantage SoA designs. To ensure a fair and consistent comparison, we therefore report results obtained using our original experimental setup. The other evaluated applications show no significant sensitivity to input ordering, with variations below 1%.

4) Average Pooling: Pooling is a widely utilized operation within modern CNNs, that reduces dimensions while preserving critical features for tasks such as object detection and image classification. Average pooling is performed using a sliding kernel that spans the input image and produces a reduced representation, where each output value corresponds to the mean of the pixels within the specified filter region. We utilized a 2×2 kernel with a stride of 1, which is implemented using three additions described in Section V-1. An example image of size 768×512 is shown in Fig. 21(a), which is reduced to size 384×256 via the aforementioned kernel, while preserving the essential features of the image (Fig. 21(b-e)). All evaluated outlines demonstrate the acceptable preservation of the image features through the average pooling, maintaining an MSSIM over 90%, and affirming its applicability for such operation.

Comparative Analysis: The proposed MAFA designs are evaluated for various image-processing applications. Fig. 22 provides a comparative analysis between the proposed MAFA designs, focusing on MSSIM as the performance metric for the NC and Kodak datasets. MAFA designs achieve an average MSSIM higher than 70% (besides MAFA-1 at greyscaling with a mean MSSIM of 69%) across all applications, demonstrat-



(a) MSSIM for Different Applications on NC Dataset [11]



(b) MSSIM for Different Applications on Kodak Dataset [46]

Fig. 22. MSSIM values for different image applications (Addition, Grayscaling, and Average pooling) conducted on NC and Kodak Dataset [11], [46] with proposed MAFA's. O-1, 2, and 3 represent Outline 1, 2, and 3, respectively.

ing its applicability for such tasks. In most applications the MSSIM values for MAFA-1,2 are lower than MAFA-3, but the images remain visually indistinguishable as highlighted by our previous examples. It is noticeable that the spread/deviation of the resulting MSSIM values significantly increases with a higher approximation degree. Further comparison is conducted with other SoA approximate designs for the image addition application using the NC and Kodak datasets. The average MSSIM values are plotted in Fig. 23, showing that MAFA designs generally outperform most SoA approximate adders, except for No Carry and No Carry+ [11], which exhibit comparable quality to MAFA-3.

The average pooling application is considered for the application-level comparison with the SoA, as a representative task. Fig. 24 presents a comparative analysis of MAFA and SoA designs in terms of energy, latency, and image quality. The energy and latency values are estimated by counting the total number of 8-bit additions required to downsample the

TABLE X
PERFORMANCE METRICS FOR PROPOSED APPROXIMATE DESIGNS FOR DIFFERENT IMAGE APPLICATIONS ON NC [11] DATASET

Image Addition						
Adder	MAFA-1		MAFA-2		MAFA-3	
Outline	PSNR	MSSIM %	PSNR	MSSIM %	PSNR	MSSIM %
1	43.44	98.49	44.04	98.72	45.22	99.07
2	37.80	95.36	38.60	96.02	39.49	96.80
3	31.78	87.38	32.70	88.79	33.67	90.75
Image Subtraction						
1	41.29	93.77	41.83	94.46	42.84	95.77
2	35.30	76.92	36.20	81.76	37.18	88.27
3	29.04	50.79	30.17	58.46	31.58	78.86
RGB to Grey						
1	33.46	93.40	33.19	94.25	35.95	95.44
2	29.90	84.29	30.54	86.50	32.62	88.44
3	22.84	68.80	23.35	72.13	26.25	74.95
Average Pooling						
1	41.76	98.31	41.33	98.71	40.77	98.82
2	37.01	95.79	36.70	96.48	35.83	96.24
3	32.25	91.59	32.05	92.40	31.02	90.80

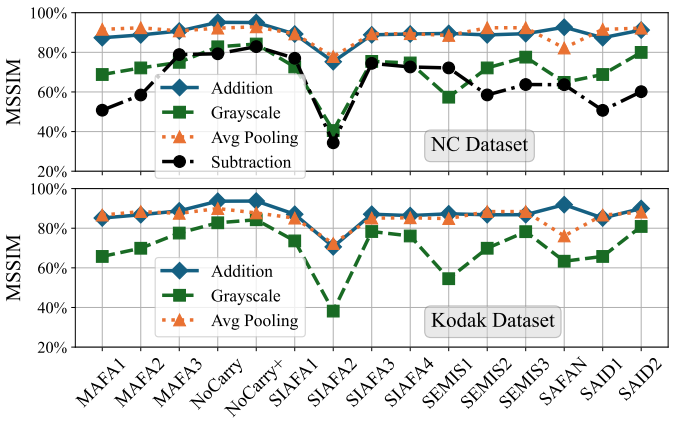


Fig. 23. Average Image Processing Results over the NC Dataset [11] and Kodak Dataset [46] compared to SoA approximate adders for $k = 5$.

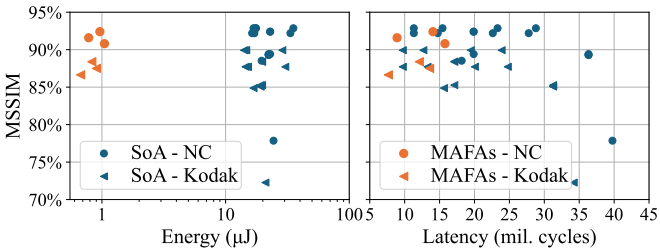


Fig. 24. Application-level comparison of MAFA to SoA at $k = 5$. Results of Average Pooling on the NC & Kodak Dataset.

exemplary 768×512 image (Fig. 21) to 384×256 using 2×2 average pooling and scaling this by the average per-operation energy/latency obtained from the circuit-level evaluation. The results show that MAFA designs achieve significantly higher energy efficiency (15–45×) while maintaining comparable high MSSIM values, and require fewer computational steps than most SoA designs, resulting in reduced latency without compromising image quality. Overall, this demonstrates the efficiency of MAFA designs at the application level in terms of energy and latency while preserving acceptable image quality.

VI. APPLICATION IN MACHINE LEARNING

This section evaluates the effectiveness of the proposed designs in Convolutional Neural Networks (CNNs) to demonstrate their applicability for ML algorithms. CNNs, widely used for image classification, segmentation, and object detection [49], [50], provide an ideal platform for evaluation due to their layered structure and data-processing capabilities. In a conventional exact 8-bit signed multiplier, multiplication proceeds in two steps. The Partial Products (PPs) are generated first, which is followed by seven stages of PP summation to produce final result [51]–[53]. Fig. 25 shows the structure of an 8-bit signed multiplier that forms the basis of our approximate designs. For PP summation, we use seven 8-bit adders in this structure and introduce approximation by replacing some or all of them with the proposed approximate MAFA adders. We designed a total of 15 approximate 8-bit signed integer multipliers with different accuracy levels using the proposed MAFAs, as summarized in Table XI. Each multiplier is labeled MULx_y, where x denotes the MAFA type (MAFA-1, MAFA-2, or MAFA-3) and y indicates that output bits 0 through y of the multiplication result (i.e., the first $y + 1$ output bits)

TABLE XI
ADDER CONFIGURATIONS AND ACCURACY CRITERIA OF THE PROPOSED MULTIPLIERS

Multipliers	Adder Configurations							Accuracy Criteria	
	k_1	k_2	k_3	k_4	k_5	k_6	k_7	MED	MRED
MAFA-1 Family									
MUL1_4	4	3	2	1	*Exact*	*Exact*	*Exact*	23.4	0.03
MUL1_5	5	4	3	2	1	*Exact*	*Exact*	48.7	0.08
MUL1_6	6	5	4	3	2	1	*Exact*	99.7	0.16
MUL1_7	7	6	5	4	3	2	1	147.2	0.26
MUL1_8	8	7	6	5	4	3	2	212.3	0.34
MAFA-2 Family									
MUL2_4	4	3	2	1	*Exact*	*Exact*	*Exact*	30.3	0.05
MUL2_5	5	4	3	2	1	*Exact*	*Exact*	70.6	0.12
MUL2_6	6	5	4	3	2	1	*Exact*	160.7	0.28
MUL2_7	7	6	5	4	3	2	1	311.8	0.53
MUL2_8	8	7	6	5	4	3	2	467.6	0.81
MAFA-3 Family									
MUL3_4	4	3	2	1	*Exact*	*Exact*	*Exact*	23.0	0.04
MUL3_5	5	4	3	2	1	*Exact*	*Exact*	52.9	0.09
MUL3_6	6	5	4	3	2	1	*Exact*	118.5	0.22
MUL3_7	7	6	5	4	3	2	1	216.8	0.42
MUL3_8	8	7	6	5	4	3	2	356.4	0.68

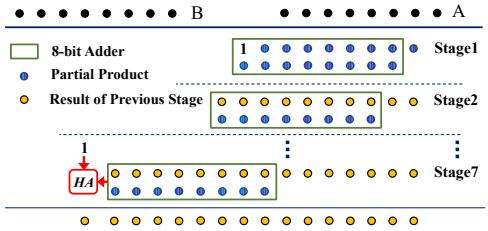


Fig. 25. Structure of an 8-bit signed multiplier using seven 8-bit adders.

are approximated. For instance, in MUL1_5, since $y = 5$, approximation is applied only to output bits 0 to 5. This is achieved by replacing the five 8-bit adders in stages 1 to 5 (Fig. 25) with MAFA-1 (because $x = 1$). Notably, as shown in Table XI for MUL1_5, these five MAFA-1 adders use different levels of approximation (i.e., different k values). This ensures that approximation affects only bits 0 to 5 of the multiplication result. In fact, the Stage-1 8-bit adder approximates 5 LSBs ($k_1 = 5$), the Stage-2 adder approximates 4 LSBs ($k_2 = 4$), and so on, with the Stage-5 adder approximating 1 LSB ($k_5 = 1$), thereby limiting approximation to bits 0 through 5 of the multiplication output. In each family, the degree of approximation is maximal for the configuration MULx_8, where all the 8-bit adders in the seven stages (Fig. 25) are approximated.

We selected nine CNN models, for CNN-based image classification on the CIFAR-10 dataset [54]. These networks serve as our case study, as they contain multiple convolutional and dense layers, building blocks that remain widely used in modern vision models. Initially, these models were trained in FP32 format (without approximation). Then, Post-Training Quantization (PTQ) was applied to convert all weights and activations to INT8 format, allowing the integration of our approximate integer multipliers. After that, we generated Look-Up Tables (LUTs) for all 15 approximate multipliers and replaced all exact multiplications with approximate ones to evaluate the performance of each MAFA-based multiplier in these networks using AdaPT framework [55]. Table XII shows the Top-1 accuracy of inference on the CIFAR-10 test set. For the exact pre-trained models, the inference accuracy of all networks dropped significantly to roughly 10% when using

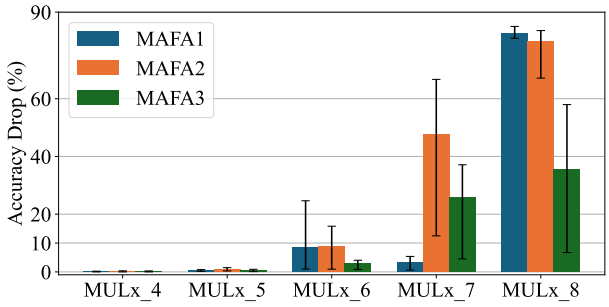


Fig. 26. Average accuracy degradation of the proposed multipliers over all CNNs. The error bars indicate the maximal deviation from the mean.

approximate multipliers. To address this, approximation-aware retraining was applied using the proposed multipliers on the training set, leading to a significant improvement in accuracy compared to the initial results. Table XII presents the accuracy results after approximate retraining. In the literature, a 10% accuracy drop is considered the threshold for acceptability [51] for CNN applications.

The proposed multiplier MULx_4 and MULx_5 met this criterion for all MAFA designs when evaluated on the CNN architectures from Table XII. The mean accuracy drop for MAFA-1, MAFA-2, and MAFA-3 for these configurations is only 0.38 percentage points, leading to an average Top-1 accuracy of 93.27% over all networks. Although the average accuracy for all MAFA-based multipliers of type MULx_6 is within the acceptable threshold, the variation between different network architectures is increasing. This leads to an accuracy drop below the threshold for some networks, such as the VGG approaches for MUL1_6 and ResNet34&50 for MUL2_6. Only MUL3_6 achieves acceptable results for all CNN architectures evaluated, with an average accuracy of 90.79%. To better highlight these results, we illustrated the mean accuracy drop compared to the exact networks for all multiplier configurations in Fig. 26. The error bars indicate the minimum and maximum results for all architectures. Although MAFA-1 introduces the highest error and does not meet the threshold for MUL1_6, the configuration MUL1_7 leads to an average accuracy degradation of only 3.27% (with the largest loss of 5.36% for ResNet34), and is well within the acceptable threshold. Multipliers MUL2_7 and MUL3_7 are not able to achieve results within the given threshold for all networks.

To analyze the trade-off between accuracy degradation and circuit-level benefits, such as energy efficiency and speed, we next examine the application-level gains in the exemplary CNN workloads. The gains are calculated by summing up the steps/energy of the individual adder with their respective approximation degree k_i and comparing this value to the total requirements of the exact MFA-based multiplier. When we compare MUL1_5, MUL2_5, and MUL3_5 to a multiplier consisting only of exact adders (i.e., the proposed MFA), the approximation approach is able to reduce latency by 10% to 24%, and energy consumption by 19% to 25%, while roughly maintaining accuracy. With MUL3_6 and MUL1_7, latency is lowered by 14% & 45% and energy-efficiency improved by 26% & 46%, while only reducing the accuracy by 2.87 and 3.27 percentage points, respectively.

TABLE XII
EVALUATION OF CNNS ACCURACY WITH VARIOUS APPROXIMATE MULTIPLIERS

Designs	DenseNet121	DenseNet161	DenseNet169	ResNet18	ResNet34	ResNet50	VGG13_bn	VGG16_bn	VGG19_bn
Exact	**93.98%**	**93.54%**	**93.87%**	**92.97%**	**93.30%**	**93.56%**	**94.15%**	**93.88%**	**93.78%**
MUL1_4	93.74%	93.54%	93.87%	92.74%	93.03%	93.38%	94.14%	93.69%	93.68%
MUL1_5	93.61%	93.05%	93.51%	92.47%	93.05%	93.16%	93.74%	93.08%	93.20%
MUL1_6	92.99%	91.06%	91.85%	89.85%	92.15%	91.54%	77.63%	71.72%	69.13%
MUL1_7	93.34%	89.14%	89.79%	89.09%	92.82%	88.20%	92.70%	91.46%	91.55%
MUL1_8	13.02%	11.00%	12.78%	11.05%	14.75%	10.63%	9.11%	10.59%	09.65%
MUL2_4	93.71%	93.43%	93.86%	92.60%	93.05%	93.20%	94.07%	93.60%	93.58%
MUL2_5	93.64%	92.82%	92.96%	91.69%	93.00%	92.09%	93.35%	92.87%	92.65%
MUL2_6	93.03%	81.66%	85.89%	85.12%	92.15%	77.73%	88.71%	86.98%	87.38%
MUL2_7	81.50%	34.85%	34.76%	54.50%	61.78%	26.86%	57.16%	50.76%	45.91%
MUL2_8	26.85%	12.83%	12.39%	16.15%	12.63%	10.94%	11.24%	11.76%	10.16%
MUL3_4	93.70%	93.48%	93.83%	92.63%	93.09%	93.37%	94.13%	93.56%	93.52%
MUL3_5	93.60%	93.10%	93.63%	92.07%	92.94%	92.97%	93.76%	93.24%	93.02%
MUL3_6	93.13%	90.64%	90.96%	89.58%	92.46%	89.53%	92.22%	90.90%	90.60%
MUL3_7	89.47%	74.32%	63.60%	70.92%	87.60%	56.44%	72.21%	69.00%	58.22%
MUL3_8	87.27%	64.19%	56.69%	61.55%	84.36%	35.59%	68.07%	58.79%	54.67%

VII. CONCLUSION

This work proposes one exact and three approximate MAGIC-based adder designs implemented using a parallel architecture for in-memory computing. The proposed MFA achieves a latency improvement of 27% to 65% over state-of-the-art MAGIC-based exact adders for 8-bit addition and attains the highest figure of merit among comparable designs. When deployed in an 8-bit RCA structure, the approximate adders MAFA-1, MAFA-2, and MAFA-3 achieve speedups of 37.3% to 85% with energy reductions of 17% to 73% compared to MAGIC-based exact adders. Relative to IMPLY-based counterparts, the proposed MAFA designs offer up to 97% and 77.6% improvements in energy and latency, respectively. The proposed approximate designs are evaluated on various image processing tasks, yielding acceptable PSNR and SSIM values, and further evaluated on multiple CNN-based image classification tasks, demonstrating acceptable accuracy. This work focuses on MAGIC-only implementations to maintain a uniform execution model and enable consistent architectural comparison. Hybrid stateful-logic designs combining MAGIC with IMPLY, majority logic, or other primitives are promising but require dedicated synthesis and benchmarking frameworks and are therefore left for future investigation.

ACKNOWLEDGEMENT

Authors at Heidelberg University would like to acknowledge and thank Hector Stiftung for their support.

REFERENCES

[1] H. Jiang *et al.* Approximate arithmetic circuits: A survey, characterization, and recent applications. *Proceedings of the IEEE*, 108(12):2108–2135, 2020.

[2] W. Liu *et al.* A retrospective and prospective view of approximate computing. *Proceedings of the IEEE*, 108(3):394–399, 2020.

[3] R. H. Dennard *et al.* A perspective on today's scaling challenges and possible future directions. *Solid-State Electronics*, 51(4):518–525, 2007. Special Issue: Papers selected from the 2006 ULIS Conference.

[4] N. TaheriNejad. In-memory computing: Global energy consumption, carbon footprint, technology, and products status quo. In *2024 IEEE 24th International Conference on Nanotechnology (NANO)*, pp. 495–500, 2024.

[5] L. Chua. Memristor-the missing circuit element. *IEEE Transactions on circuit theory*, 18(5):507–519, 1971.

[6] D. B. Strukov *et al.* The missing memristor found. *Nature*, 453:80–83, 2008.

[7] Y. Li *et al.* A survey of mram-centric computing: From near memory to in memory. *IEEE Transactions on Emerging Topics in Computing*, 2022.

[8] J. Borghetti *et al.* 'memristive' switches enable 'stateful' logic operations via material implication. *Nature*, 464(7290):873–876, 2010.

[9] S. Kvatinsky *et al.* Memristor-based material implication (imply) logic: Design principles and methodologies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(10):2054–2066, 2013.

[10] S. Kvatinsky *et al.* Magic—memristor-aided logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(11):895–899, 2014.

[11] F. Seiler and N. TaheriNejad. Accelerated image processing through imply-based nocarry approximated adders. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(11):5141–5154, 2024.

[12] N. Amirafshar *et al.* Carry disregard approximate multipliers. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 70(12):4840–4853, 2023.

[13] S. E. Fatemieh *et al.* Fast and compact serial imply-based approximate full adders applied in image processing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 13(1):175–188, 2023.

[14] N. Talati *et al.* Logic design within memristive memories using memristor-aided logic (magic). *IEEE Transactions on Nanotechnology*, 15(4):635–650, 2016.

[15] E. Lehtonen and M. Laiho. Stateful implication logic with memristors. In *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 33–36. IEEE, 2009.

[16] P. Huang *et al.* Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits. *Advanced Materials*, 28(44):9758–9764, 2016.

[17] S. Gupta *et al.* Felix: Fast and energy-efficient logic in memory. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, 2018.

[18] N. TaheriNejad. Sixor: Single-cycle in-memristor xor. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(5):925–935, 2021.

[19] P. Thangkhiew *et al.* Efficient implementation of adder circuits in memristive crossbar array. In *TENCON 2017-2017 IEEE Region 10 Conference*, pp. 207–212. IEEE, 2017.

[20] Ben-Hur et al. Simpler magic: Synthesis and mapping of in-memory logic executed in a single row to improve throughput. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2434–2447, 2020.

[21] C. K. Jha *et al.* Investigating various adder architectures for digital in-memory computing using magic-based memristor design style. In *2022 IEEE International Conference on Emerging Electronics (ICEE)*, pp. 1–4, 2022.

[22] P. L. Thangkhiew *et al.* Area efficient implementation of ripple carry adder using memristor crossbar arrays. In *2016 11th International Design & Test Symposium (IDT)*, pp. 142–147. IEEE, 2016.

[23] R. Gharpinde *et al.* A scalable in-memory logic synthesis approach using memristor crossbar. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(2):355–366, 2017.

[24] J. Liang *et al.* New metrics for the reliability of approximate and probabilistic adders. *IEEE Transactions on Computers*, 62(9):1760–1771, 2013.

[25] W. Liu *et al.* Design and analysis of majority logic-based approximate adders and multipliers. *IEEE Transactions on Emerging Topics in Computing*, 9(3):1609–1624, 2019.

[26] Z. Wang *et al.* Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[27] F. Seiler and N. TaheriNejad. Efficient image processing via memristive-based approximate in-memory computing. *IEEE Transactions on*

Computer-Aided Design of Integrated Circuits and Systems, pp. 1–12, 2024.

[28] S. Asgari *et al.* Energy-efficient and fast imply-based approximate full adder applying nand gates for image processing. *Computers and Electrical Engineering*, 113:109053, 2024.

[29] N. Kaushik *et al.* High-speed serial and semi-parallel imply-based approximate adders through memristors for in-memory computing. In *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2024.

[30] F. Seiler and N. TaheriNejad. An imply-based semi-serial approximate in-memristor adder. In *2023 IEEE Nordic Circuits and Systems Conference (NorCAS)*, pp. 1–7, 2023.

[31] D. Ochs *et al.* Approchs: A memristor-based in-memory adaptive approximate adder. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 15(1):105–119, 2025.

[32] C. Simonides *et al.* Approximated 2-bit adders for parallel in-memristor computing with a novel sum-of-product architecture. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 10:135–143, 2024.

[33] C. K. Jha *et al.* Imagin: Library of imply and magic nor-based approximate adders for in-memory computing. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 8(2):68–76, 2022.

[34] C. K. Jha *et al.* Input distribution aware library of approximate adders based on memristor-aided logic. In *2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID)*, pp. 577–582, 2024.

[35] V. Gupta *et al.* Impact: Imprecise adders for low-power approximate computing. In *IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 409–414, 2011.

[36] C. Labrado *et al.* Design of majority logic based approximate arithmetic circuits. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, 2017.

[37] S. Kvatinsky et al. Vteam: A general model for voltage-controlled memristors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(8):786–790, 2015.

[38] S. G. Rohani et al. An improved algorithm for imply logic based memristive full-adder. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4. IEEE, 2017.

[39] F. Seiler and N. TaheriNejad. An improved serial imply adder algorithm for efficient neural network applications. In *2025 IEEE 16th Latin America Symposium on Circuits and Systems (LASCAS)*, volume 1, pp. 1–5, 2025.

[40] N. Taherinejad et al. A semi-serial topology for compact and fast imply-based memristive full adders. In *2019 17th IEEE International New Circuits and Systems Conference (NEWCAS)*, pp. 1–4. IEEE, 2019.

[41] S. G. Rohani et al. A semiparallel full-adder in imply logic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(1):297–301, 2019.

[42] A. Karimi and A. Rezai. Novel design for a memristor-based full adder using a new imply logic approach. *Journal of Computational Electronics*, 17(3):1303–1314, 2018.

[43] D. Radakovits *et al.* A memristive multiplier using semi-serial imply-based adder. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(5):1495–1506, 2020.

[44] S. Singh *et al.* Should we even optimize for execution energy? rethinking mapping for magic design style. *IEEE Embedded Systems Letters*, 15(4):230–233, 2023.

[45] D. Ielmini and H.-S. P. Wong. In-memory computing with resistive switching devices. *Nature electronics*, 1(6):333–343, 2018.

[46] Kodak Lossless True Color Image Suite . Available: https://r0k.us/graphics/kodak/. [Online; accessed 6th-March-2024].

[47] A. Prati *et al.* Detecting moving shadows: algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):918–923, 2003.

[48] Y. S. Mehrabani *et al.* A novel high-speed, low-power cntfet-based inexact full adder cell for image processing application of motion detector. *Journal of Circuits, Systems and computers*, 26(05):1750082, 2017.

[49] S. Shakibhamedan *et al.* An analytical approach to enhancing DNN efficiency and accuracy using approximate multiplication. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*, 2024.

[50] D. Katare *et al.* Approximation strategies for vision models on edge devices: An accuracy-efficiency trade-off. *TechRxiv*, December 2024.

[51] S. Shakibhamedan *et al.* Ace-cnn: Approximate carry disregard multipliers for energy-efficient cnn-based image classification. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(5):2280–2293, 2024.

[52] N. Amirafshar *et al.* Prim: Hybrid array-compressor multipliers with carry disregard and or-based approximation. In *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2025.

[53] N. Amirafshar *et al.* An approximate carry disregard multiplier with improved mean relative error distance and probability of correctness. In *2022 25th Euromicro Conference on Digital System Design (DSD)*, pp. 46–52, 2022.

[54] CIFAR-10 (Canadian Institute for Advanced Research) . Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Online; accessed 15th-Feb-2025].

[55] D. Danopoulos *et al.* Adapt: Fast emulation of approximate dnn accelerators in pytorch. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2022.

Dr. Nandit Kaushik (Graduate Student Member, IEEE) received the M.Tech degree in ECE from The LNM Institute of Information & Technology (LNMIIT), Jaipur, India, in 2019 and Ph.D from Indian Institute of Technology (IIT) Mandi, India in 2025. He was a visiting researcher at the Institute of Computer Engineering (ZITI), Heidelberg University, Germany from Dec 2024 to May 2025. Currently, he is working as an Assistant Professor at Rashtriya Raksha University, Gandhinagar, Gujarat, India. His current research focuses on developing memristor-based arithmetic architectures for In-Memory Computing and their efficient implementation inside a memristive crossbar.

Fabian Seiler received the B.Sc. degree in electrical engineering and information technology in 2023 and the M.Sc. degree in embedded systems in 2025, both from TU Wien, Vienna, Austria. He is currently pursuing his Ph.D. at Heidelberg University, Germany. His research interests include memristive circuits and systems, in-memory and approximate computing, and machine learning. He has received multiple merit scholarships from TU Wien, the Palfinger Elevate Scholarship, and is the winner of the IEEE SMACD 2025 EDA Competition.

Nima Amirafshar received his B.Sc. degree in Electrical Engineering from Ferdowsi University of Mashhad (FUM), Mashhad, Iran, in 2019, and his M.Sc. degree in Electrical Engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2023. He is currently pursuing a Ph.D. at Heidelberg University, Heidelberg, Germany. His research interests include computer architecture, approximate computing, hardware acceleration for machine learning, and data-centric circuits and systems.

Nima TaheriNejad received his Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, Canada, in 2015. He is currently a Full Professor at Heidelberg University, Heidelberg, Germany. His areas of work include in-memory computing, cyber-physical and embedded systems, systems on chip, memristor-based circuit and systems, self-* systems, and health-care. He has published three books, three patents, and more than 100 articles.

Dr. Srinivasu Bodapati (Member, IEEE) received the Ph.D. degree in VLSI from the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai, India, in 2017. From 2017 to 2019, he was a Research Fellow at Nanyang Technological University, Singapore. Since 2019, he has been an Assistant Professor at the School of Computing and Electrical Engineering, Indian Institute of Technology Mandi, Mandi, India. His research interests include digital design with emerging device technologies, field-programmable gate array (FPGA)-based designs, and FPGA implementation of cryptographic primitives.