Fast and Compact Serial IMPLY-Based Approximate Full Adders applied in Image Processing

Seyed Erfan Fatemieh, Mohammad Reza Reshadinezhad, and Nima TaheriNejad, *Member, IEEE.*

***Abstract*—The barriers to improving computers' performance have led to the emergence of new computing paradigms and technologies. Among these, the memristors are of great concern. In addition to storing data, memristors can perform logical operations and are proper for In-Memory Computation (IMC). Furthermore, approximate computing is an emerging paradigm introduced to improve performance by reducing the accuracy of calculations in error-resistant applications. These two concepts are combined and presented in four serial Material Implication (IMPLY)-based approximate full adders. In addition, to the positive features of the serial method, the proposed circuits reduce the number of calculation steps by 7%-43%, and the energy consumption improves by 56%-68% compared to the existing exact full adders. The accuracy loss of proposed circuits in different simulated scenarios combining exact and approximate adders are analyzed. Four different image processing applications are applied to ensure the proper functionality of the proposed circuits. The results indicate that in most scenarios, the quality of the images is acceptable, and the Peak Signal-to-Noise Ratio (PSNR) criterion is more than 30 dB.**

***Index Terms*—Approximate Computing, Approximate Full adder, Memristors, IMPLY, Image Processing, Error Analysis.**

I. Introduction

ARITHMETIC circuits are among the most significant parts of a processor. Addition and multiplication are the basic instructions of processors. Digital Signal Processing (DSP) processors are one of the most widely used processors, and a high percentage of the instructions executed in these processors need adders and multipliers [1]–[3]. Hence, improving them can significantly affect the overall performance improvement of computing systems. Recently, this has gained additional importance given the challenges that other methods of improving computer performance face (e.g., the end of Dennard's scaling, the slowdown of Moore's law, and the von Neumann bottleneck). In addition, this has led to several emerging technologies, such as memristive technology, and computing paradigms, such as approximate computing.

Approximate computing is among the most promising solutions to overcome the power wall problem [4]–[9]. In approximate computing, the accuracy of the calculations is reasonably decreased, resulting in a significant reduction of the hardware complexity, power consumption, and delay [2], [4], [6]–[12]. Approximate computing can be applied in error-resilient applications such as image and video processing, pattern recognition, machine learning, communication, robotics, and data mining [1], [4], [10], [11], [13]. In image processing, considerable accuracy reduction can be tolerated since human visual perception is limited [1], [6], [8]. Pattern recognition and classification are among the applications that require statistical and probabilistic computations, and the errors in calculations do not significantly decrease their performance [10]. One of the reasons that make DSP applications error-resilient is noisy inputs, due to which the accuracy of calculations is limited [10], [11].

Data transfer from memory to processor and vice-versa is associated with considerable power consumption and cost [14]–[18]. Nevertheless, IMC solutions can tackle von Neumann's bottleneck, especially using new technologies, such as memristors, that can process and store data [14]–[20]. Several studies have been conducted on IMC and its different architectures for various applications [14], [15], [17]–[19], [21]–[27]. Among several memristive IMC logics, stateful logics stand out as they do not need to read and write data to perform calculations. Thus, they reduce the associated power consumption and delay [14]. The IMPLY logic is a stateful logic, compatible with the crossbar array [16]–[18]. The design of arithmetic circuits using this logic for IMC is well-studied [16], [17], [20], [23], [26], [28]. IMPLY-based adders can be divided into three main architectures: serial, parallel, and hybrid [16]–[18], [20], [23], [28], [29]. In this article, we expand and improve our initial adder [16] and present four serial IMPLY-based approximate full adder (SIAFA) units. Their number of memristors is as minimal as possible, and they reduce power consumption and hardware complexity. The proposed circuits also improve the number of steps required to calculate the outputs compared to the exact serial full adders.

This article expands and improves the state-of-the-art (SoA) thanks to the following key contributions:

- Improving the number of memristors required for the n-

Manuscript received October 16, 2022; revised December 19, 2022, and January 11, 2023; accepted January 11, 2023.

S. E. Fatemieh and M. R. Reshadinezhad are with the Department of Computer Architecture, Faculty of Computer Engineering, University of Isfahan, Isfahan 8174673441, Iran (email: erfanfatemieh@eng.ui.ac.ir; m.reshadinezhad@eng.ui.ac.ir) (*Corresponding author: Mohammad Reza Reshadinezhad).*

N. TaheriNejad is currently with the Faculty of Engineering, Institute of Computer Engineering at Heidelberg University, Heidelberg, Germany, and was with the Faculty of Electrical Engineering and Information Technology, Institute of Computer Technology, TU Wien, Vienna, Austria, for a portion of this work (email: nima.taherinejad@ziti.uni-heidelberg.de).

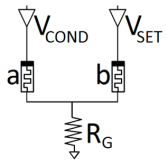

Fig. 1. Circuit-level memristor-based IMPLY logic gate [17]

bit adder by providing a new algorithm;
- Proposing three new fast and compact approximate full adders;
- Thorough evaluation of error metrics in different scenarios;
- Introducing a Figure of Merit (FOM) for evaluating both hardware performance metrics and error metrics at once;
- Extending application-level evaluations for a larger number of approximate adders using different combinations of approximate and exact full adders;
- Adding three image processing applications to the previously used image addition application.

The rest of this paper is organized into four sections. The backgrounds of this study and the SoA are reviewed in Section II. SIAFAs and their implementation algorithms are presented in Section III. In Section IV, SPICE simulation results are provided to validate the functionality of the proposed circuits. Various error metrics are assessed in different adder scenarios and four image processing applications, presented in Section IV and Section V, respectively. The results of image quality analyses are reported in Section V. Finally, in Section VI, we conclude and discuss future works.

II. Background

A. Memristors and IMPLY logic

The memristor is a two-terminal element, along with the resistor, inductor, and capacitor are the four main circuit elements. It is a non-volatile memory that stores values as its resistance and boasts of features such as very low power consumption, writing time, and small dimensions [14], [26], [27], [30]. Two maximum (R_{off}) and minimum (R_{on}) resistance values of memristors are reached based on the memristor's current direction and input voltage. In digital design, the maximum resistance value can be considered equivalent to logic zero, and the minimum resistance value is equivalent to logic one [16]–[18], [20]. Therefore, we apply this convention here.

In some digital memristive circuits, the input and output values of the logical operation are represented using the memristor's resistance. Such logic is stateful logic [14], [16]. Furthermore, these methods reduce the number of reading and writing on memristors since no reading and writing operations are needed to perform logical operations and calculations [14]. Although there exist other non-stateful logics, such as Memristor Ratioed Logic (MRL) [26], [31], due to the favorable properties of stateful logics, we choose them for our work.

HP introduced the IMPLY logic, which has been heavily used as the first memristive stateful logic [16]. The basic

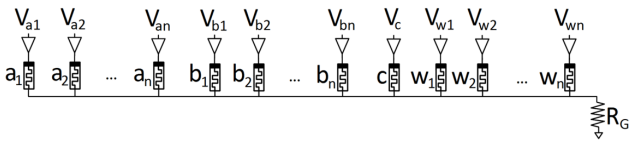

Fig. 2. The IMPLY-based serial n-bit adder architecture [17]

structure of this method can be seen in Fig. 1. In IMPLY logic, two memristors, two constant voltages, V_{Cond} and V_{Set}, and resistance R_G are needed to implement IMPLY operation. The following conditions, $R_{On} << R_G << R_{Off}$ and $V_{Cond} < V_c < V_{Set}$, where V_c is the threshold voltage of the memristor, must be satisfied for correct operations [16]–[18], [20], [23]. The placement of memristors in the IMPLY logic is compatible with the crossbar arrays, and this feature makes the IMPLY logic a suitable method for processing in memory. IMPLY (IMP), and FALSE functions constitute a complete logic set and can be used to implement all logical functions.

B. IMPLY-based adders

IMPLY-based adders can be divided into serial and parallel architectures, and hybrid ones (semi-serial and semi-parallel). In the serial design method, memristors are placed in the same column or row of the crossbar array, which is the standard design architecture on crossbar arrays [17], [18]. The serial memristor-based adder is demonstrated in Fig. 2 [17], [18]. The serial adder performs an IMPLY or FALSE operation in each clock cycle [18]. The number of computational steps in the serial method is more than in other methods. However, the required number of memristors, hardware complexity, and area usage is less than the other methods. They do not need any external switches either. The serial full adder proposed in [18] calculates the outputs in 22 steps using 5 memristors and has the best performance compared to the previous serial designs.

In the design of IMPLY-based parallel adders, input memristors, work memristors, and a work resistor are placed in one row (column) to calculate each bit. Fig. 3 depicts the IMPLY-based parallel adder architecture. In the parallel architecture, parts of the calculation algorithm that do not have dependencies between memristors are calculated simultaneously [18], [20]. Thus, the number of steps in this method reduces significantly, but the area and hardware complexity increase [18], [20]. However, all the computational steps cannot be parallelized as some steps depend on each other and must be calculated sequentially [18]. The parallel adder proposed in [28] calculates the final result using $4n+1$ memristors and $5n+16$ steps. In addition, it requires n external switches for an n-bit addition [28].

In Fig. 4, the semi-parallel structure is shown [17], [20]. The purpose of designing a semi-parallel adder is to simultaneously improve the number of computational steps compared to the serial design and reduce the area compared to the parallel design [20]. In the top row, the first input and a work memristor are located [20]. The second input memristor, the C_{in} memristor, and a work memristor are placed in the bottom row [20]. If it is possible to parallelize commands in two rows (columns) and there is no dependency between the steps, each

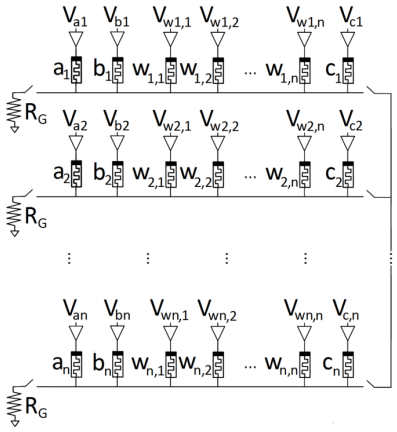

Fig. 3. The IMPLY-based parallel n-bit adder architecture [17]

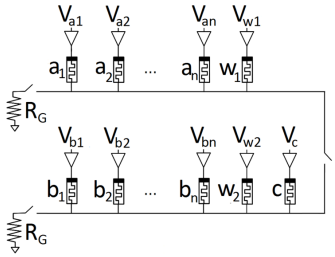

Fig. 4. The IMPLY-based semi-parallel n-bit adder architecture [17]

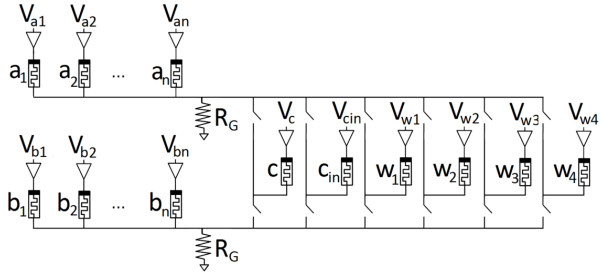

Fig. 5. The IMPLY-based semi-serial n-bit adder architecture [17]

row/column executes one command in each cycle. A switch can connect the upper and lower rows, and the calculations can be performed between the memristors of the two rows [20]. The proposed algorithm in the semi-parallel structure calculates the outputs of a full adder in 17 cycles [20]. The semi-parallel adder requires three external Complementary Metal Oxide Semiconductor (CMOS) switches [20].

The structure of the semi-serial adder can be seen in Fig. 5. As a hybrid architecture, like semi-parallel, this architecture also tries to strike a more balanced trade-off between area and speed compared to serial and parallel architectures. This structure consists of two parallel rows for the first and second input, and each row is connected to a single work resistor. The five work memristors and the C_{in} memristor are connected to the rows of the first and second inputs by two separate switches [17]. This structure requires $2n+6$ memristors, two work resistors, and 12 switches [17]. The semi-serial adder requires $10n+2$ computational steps for n-bit addition [17].

C. Approximate computing

1) Metrics: In approximate computing, power consumption, delay, and area are reduced significantly by reducing the complexity of the hardware at the cost of accuracy. Hence, the accuracy reduction should be properly evaluated. To this end, several error analysis criteria have been introduced in SoA [1], [10], [11], [32]–[34]. The most important error analysis criteria are [1], [10], [11], [32]–[34]:

- Error distance (ED): the absolute value of the difference between the exact and approximate values.
- Error rate (ER): The ratio of the number of erroneous outputs to the total possible outputs.
- Relative ED (RED): The ratio of ED to the exact value.
- Mean ED (MED): The average ED of all possible inputs.
- The normalized MED (NMED): The normalized value of MED.
- The Mean RED (MRED): The average RED of all possible inputs.

Error-resistant applications such as image processing are among the applications where approximate computing is often applied [1], [6], [8]. PSNR is the widely used image quality criterion to evaluate the quality of the images obtained from approximate computing. The image quality is acceptable if the PSNR exceeds 30 dB [12], [35]. In addition to the PSNR, Structural Similarity Index (SSIM) and Mean SSIM (MSSIM) are the other image quality analysis metrics that are also examined to assess the quality of the output images of approximate circuits [1], [36], [37]. For more information about these two criteria, please refer to [36], [37].

2) Approximate full adders: Different approximate full adders in various technologies, CMOS and otherwise, have been introduced [1], [6]–[11], [32]–[34], [38]–[40]. The primary methods for designing approximate arithmetic circuits are redefining approximate logic from exact logic by removing transistors of exact circuits or/and applying different input voltages to the gate of transistors and/or altering the truth table of the circuits [1].

3) Memristor-based approximate full adders: Approximate full adders based on memristor technology and by applying the MRL method are introduced in [38], [39]. These inexact full adders are designed based on redefining the approximate logic from the exact logic by removing the memristors from the exact structure and altering the exact full adder's truth table. The exact full adder designed based on the MRL method requires 33 memristors [38]. In MRL-based full adders, there is a need for buffers to maintain the voltage level of the circuit nodes [38]. The transmission of signals between memristors and transistors (buffers) at the device level also affects the delay and power consumption. In [38], the authors have proposed an approximate full adder based on the MRL method, which requires ten memristors and some CMOS inverters. The proposed approximate full adder in [38] has been evaluated in the error-tolerant application of image addition, and the PSNR was used to evaluate the quality of the output images.

In another research, Prabaharan et al. proposed another approximate full adder based on the MRL method by redefinition the approximate logic from the exact logic and removing the

memristors from the exact structure [39]. The full adder proposed in [39] consists of 14 memristors and some CMOS inverters. The functionality of this approximate full adder has been evaluated in the image addition application [39]. PSNR is the image quality metric that evaluates the approximate outputs [39].

To the best of our knowledge, we are the first to introduce approximate computing to stateful memristive IMC [16]. In this work, we expand on and extend the concept presented therein.

III. Proposed IMPLY-based Approximate Full Adders

A. Method

One of the main methods for designing approximate circuits is redefining approximate logic from exact logic [1], [11]. It is common to design approximate circuits by removing or modifying exact circuits' elements or changing the exact truth table to an inexact one and designing hardware-efficient circuits [1], [10], [11].

The truth table of the IMPLY function can be seen in Table I. This paper designs the proposed circuits based on the IMPLY function's first (P='0', Q='0') and third (P='1', Q='0') states, similar to the method introduced in [18]. In these states of the IMPLY function, if the first and second inputs are zero, the output is the inverted value of the first input, and it is one. If the first input is one and the second input is zero, the output is zero, which is the inverted value of the first input. Four proposed serial IMPLY-based approximate full adders (SIAFA) are introduced here based on the IMPLY function.

In this article, not only has the truth table been simplified to get the approximate circuit outputs logic, but also the input vectors of the IMPLY function have been specified. Furthermore, this is done in such a way that the logical functions of the outputs of the proposed IMPLY-based approximate full adders can be implemented with the minimum number of calculation steps based on the IMPLY and FALSE functions. In other words, the main goal of assessing the truth table here was to reduce the number of calculation steps based on the proposed IMPLY-based output equations. In addition, the possibility of fast implementation of output functions using the minimum possible number of memristors was desired.

The main limitation of the proposed method is that the input vectors must be determined so that the output error analysis metrics are acceptable. The output logic created based on the input vectors can be implemented with the appropriate number of steps and memristors to achieve the main goals in approximate calculations, i.e., a limited reduction in accuracy versus a reduction in hardware complexity.

B. SIAFA1 and its implementation algorithm

In the first SIAFA, to produce the Sum output, the first input (P) is "01011111," and the second input (Q) is "11001100". If P $\rightarrow$ Q is done, the output is "11101100". If this value is assigned to the output (Sum) of the first proposed approximate full adder and compared to the exact full adder's Sum, its ER is $\frac{3}{8}$. If the C_{out} of the SIAFA1 is equal to $\overline{Sum}$, the C_{out} is equivalent to "00010011", which has an ER of $\frac{1}{8}$ compared to the exact full adder's C_{out}. To generate the first SIAFA's C_{out}, the first input of the IMPLY function (P) is equal to the value of the SIAFA1's Sum, and the second input (Q) is equivalent to "0000000". So P $\rightarrow$ Q (SIAFA1's C_{out}) is "00010011".

TABLE I
The truth table of IMPLY logic

P	Q	$P\ IMP\ Q \equiv P \rightarrow Q$
0	0	1
0	1	1
1	0	0
1	1	1

TABLE II
The truth table of SIAFA1 [16]

A_{in}	B_{in}	C_{in}	Exact Sum	Exact C_{out}	Approximate Sum	Approximate C_{out}
0	0	0	0	0	1 ✗	0 ✓
0	0	1	1	0	1 ✓	0 ✓
0	1	0	1	0	1 ✓	0 ✓
0	1	1	0	1	0 ✓	1 ✓
1	0	0	1	0	1 ✓	0 ✓
1	0	1	0	1	1 ✗	0 ✗
1	1	0	0	1	0 ✓	1 ✓
1	1	1	1	1	0 ✗	1 ✓

$$Sum_{SIAFA1} = (\overline{A_{in}} \rightarrow C_{in}) \rightarrow \overline{B_{in}} \equiv \overline{B_{in}} + \overline{A_{in}}.\overline{C_{in}} \quad (1)$$

$$C_{out_{SIAFA1}} = \overline{Sum_{SIAFA1}} \quad (2)$$

The SIAFA1's truth table is shown in Table II, and the logical functions of its outputs were written in (1) and (2). The exact and inexact states are marked with check and cross marks. Finally, the eight-step algorithm shown in Table III was applied to implement the first SIAFA in a crossbar array serially.

C. SIAFA2 and its implementation algorithm

The approximate C_{out}'s column in Table IV is the same as the exact C_{out}'s column in seven rows (states). The second row's value of the seventh column of Table IV was assigned by inverting the value of the second row ($A_{in}B_{in}C_{in}$="001") of the exact C_{out}'s column. This column shows the C_{out} bits of the second SIAFA. Accordingly, the ER of this full

TABLE III
SIAFA1's implementation algorithm based on IMPLY and FALSE operations

Step	Operation	Equivalent logic
1	$S_1 = 0$	$FALSE(S_1)$
2	$A_{in} \rightarrow S_1 = {S_1}'$	$NOT(A_{in})$
3	$A_{in} = 0$	$FALSE(A_{in})$
4	$B_{in} \rightarrow A_{in} = {A_{in}}'$	$NOT(B_{in})$
5	${S_1}' \rightarrow C_{in} = {C_{in}}'$	$NOT(A_{in}) \rightarrow C_{in}$
6	${C_{in}}' \rightarrow {A_{in}}' = {A_{in}}''$	$Sum = (\overline{A_{in}} \rightarrow C_{in}) \rightarrow \overline{B_{in}}$
7	$C_{in} = 0$	$FALSE(C_{in})$
8	${A_{in}}'' \rightarrow C_{in} = {C_{in}}'$	$C_{out} = \overline{Sum}$

TABLE IV
THE TRUTH TABLE OF SIAFA2

A_{in}	B_{in}	C_{in}	Exact Sum	Exact C_{out}	Approximate Sum	Approximate C_{out}
0	0	0	0	0	1 ✗	0 ✓
0	0	1	1	0	1 ✓	1 ✗
0	1	0	1	0	1 ✓	0 ✓
0	1	1	0	1	0 ✓	1 ✓
1	0	0	1	0	1 ✓	0 ✓
1	0	1	0	1	0 ✓	1 ✓
1	1	0	0	1	0 ✓	1 ✓
1	1	1	1	1	0 ✗	1 ✓

TABLE V
SIAFA2'S IMPLEMENTATION ALGORITHM BASED ON IMPLY AND FALSE OPERATIONS

Step	Operation	Equivalent logic
1	$S_1 = 0$	$FALSE(S_1)$
2	$S_2 = 0$	$FALSE(S_2)$
3	$B_{in} \rightarrow S_1 = {S_1}'$	$NOT(B_{in})$
4	$B_{in} \rightarrow S_2 = {S_2}'$	$NOT(B_{in})$
5	$A_{in} \rightarrow {S_1}' = {S_1}''$	$A_{in} \rightarrow NOT(B_{in})$
6	${S_1}'' \rightarrow C_{in} = {C_{in}}'$	$C_{out} = (A_{in} \rightarrow \overline{B_{in}}) \rightarrow C_{in}$
7	${S_2}' \rightarrow A_{in} = {A_{in}}'$	$NOT(B_{in}) \rightarrow A_{in}$
8	$B_{in} = 0$	$FALSE(B_{in})$
9	${A_{in}}' \rightarrow B_{in} = {B_{in}}'$	$Not(NOT(B_{in}) \rightarrow A_{in})$
10	${C_{in}}' \rightarrow {B_{in}}' = {B_{in}}''$	$Sum = ((A_{in} \rightarrow \overline{B_{in}}) \rightarrow C_{in}) \rightarrow \overline{(B_{in} \rightarrow A_{in})}$

adder's C_{out} is $\frac{1}{8}$. We aim to reach this output in this approximate full adder by applying IMPLY and FALSE functions. The first input (P) is equal to "11111100", and the second input (Q) is equal to "01010101". The output of P $\rightarrow$ Q equals "01010111", which is the C_{out} of the second proposed approximate adder (SIAFA2). Six computational steps are needed to implement this output according to the proposed algorithm in Table V. Now, if the first input in P $\rightarrow$ Q is equal to "01010111" (C_{out}) and the second input is equal to "11000000", the result of P $\rightarrow$ Q is equal to "11101000". By comparing this output with the Sum of an exact full adder, it can be said that the approximate Sum is inexact in $A_{in}B_{in}C_{in}$="000" and "111" states. Therefore, the ER of the Sum of the second SIAFA equals $\frac{2}{8}$. In the proposed algorithm, see Table V, the Sum of this proposed approximate full adder is calculated in the tenth step. The logical equations of the Sum and C_{out} of the second SIAFA are:

$$C_{out_{SIAFA2}} = (A_{in} \rightarrow \overline{B_{in}}) \rightarrow C_{in} \equiv \overline{A_{in}}.\overline{B_{in}} + \overline{A_{in}}.\overline{C_{in}} + \overline{B_{in}}.\overline{C_{in}} \quad (3)$$

$$Sum_{SIAFA2} = ((A_{in} \rightarrow \overline{B_{in}}) \rightarrow C_{in}) \rightarrow \overline{(\overline{B_{in}} \rightarrow A_{in})} \equiv C_{in} + A_{in}.B_{in} \quad (4)$$

D. SIAFA3 and its implementation algorithm

In SIAFA3, some of the outputs' values of the exact full adder's truth table are inverted to redefine and construct the approximate full adder's truth table. The truth table of the SIAFA3 is shown in Table VI. The output of IMPLY function (P $\rightarrow$ Q) is "11111000" when the first input of IMPLY function (P) is "01110111" and its second input (Q) is "11110000". This output is assigned to the third SIAFA's Sum. If it is compared to the exact full adder's Sum, its ER is $\frac{3}{8}$.

TABLE VI
THE TRUTH TABLE OF SIAFA3

A_{in}	B_{in}	C_{in}	Exact Sum	Exact C_{out}	Approximate Sum	Approximate C_{out}
0	0	0	0	0	1 ✗	0 ✓
0	0	1	1	0	1 ✓	0 ✓
0	1	0	1	0	1 ✓	0 ✓
0	1	1	0	1	1 ✗	0 ✗
1	0	0	1	0	1 ✓	0 ✓
1	0	1	0	1	0 ✓	1 ✓
1	1	0	0	1	0 ✓	1 ✓
1	1	1	1	1	0 ✗	1 ✓

TABLE VII
SIAFA3'S IMPLEMENTATION ALGORITHM BASED ON IMPLY AND FALSE OPERATIONS

Step	Operation	Equivalent logic
1	$S_1 = 0$	$FALSE(S_1)$
2	$B_{in} \rightarrow S_1 = {S_1}'$	$NOT(B_{in})$
3	$B_{in} = 0$	$FALSE(B_{in})$
4	$A_{in} \rightarrow B_{in} = {B_{in}}'$	$NOT(A_{in})$
5	${S_1}' \rightarrow C_{in} = {C_{in}}'$	$NOT(B_{in}) \rightarrow C_{in}$
6	${C_{in}}' \rightarrow {B_{in}}' = {B_{in}}''$	$Sum = (\overline{B_{in}} \rightarrow C_{in}) \rightarrow \overline{A_{in}}$
7	$C_{in} = 0$	$FALSE(C_{in})$
8	${B_{in}}'' \rightarrow C_{in} = {C_{in}}'$	$C_{out} = \overline{Sum}$

The second output of this proposed approximate full adder, C_{out}, is made based on applying IMPLY function and inverting the Sum output. So, the first operand of IMPLY function (P) is equal to the proposed Sum, and the second operand (Q) is "00000000". As a result, P $\rightarrow$ Q, the second approximate full adder's C_{out}, is equal to "00000111," and its ER is $\frac{1}{8}$. The logical equations of the outputs of SIAFA3 are:

$$Sum_{SIAFA3} = (\overline{B_{in}} \rightarrow C_{in}) \rightarrow \overline{A_{in}} \equiv \overline{A_{in}} + \overline{B_{in}}.\overline{C_{in}} \quad (5)$$

$$C_{out_{SIAFA3}} = \overline{Sum_{SIAFA3}} \quad (6)$$

The proposed circuit can be implemented in a memristive crossbar array serially using eight steps algorithm of Table VII.

E. SIAFA4 and its implementation algorithm

The truth table of SIAFA4 is presented in Table VIII. In the first stage, for designing the approximate Sum, the first input of the IMPLY function (P) is "00111111", and the second input (Q) is "10101010". P $\rightarrow$ Q is equal to "11101010", which is assigned to the Sum output. The ER of Sum equals $\frac{3}{8}$. Now, in the second stage, if the Sum output is the first operand (P) of IMPLY function and the second one (Q) is "00000000", the result of P $\rightarrow$ Q is "00010101", which is the C_{out} of the fourth SIAFA. The ER of this full adder's C_{out} is $\frac{1}{8}$. In the two columns to the right of Table VIII, the truth table of the SIAFA4 is shown. Exact values in these columns, like the three previous proposed approximate full

TABLE VIII
THE TRUTH TABLE OF SIAFA4

A_{in}	B_{in}	C_{in}	Exact Sum	Exact C_{out}	Approximate Sum	Approximate C_{out}
0	0	0	0	0	1 ✗	0 ✓
0	0	1	1	0	1 ✓	0 ✓
0	1	0	1	0	1 ✓	0 ✓
0	1	1	0	1	0 ✓	1 ✓
1	0	0	1	0	1 ✓	0 ✓
1	0	1	0	1	0 ✓	1 ✓
1	1	0	0	1	1 ✗	0 ✗
1	1	1	1	1	0 ✗	1 ✓

TABLE IX
SIAFA4'S IMPLEMENTATION ALGORITHM BASED ON IMPLY AND FALSE OPERATIONS

Step	Operation	Equivalent logic
1	$S_1 = 0$	$FALSE(S_1)$
2	$A_{in} \rightarrow S_1 = S_1{}'$	$NOT(A_{in})$
3	$A_{in} = 0$	$FALSE(A_{in})$
4	$C_{in} \rightarrow A_{in} = A_{in}{}'$	$NOT(C_{in})$
5	$S_1{}' \rightarrow B_{in} = B_{in}{}'$	$NOT(A_{in}) \rightarrow B_{in}$
6	$B_{in}{}' \rightarrow A_{in}{}' = A_{in}{}''$	$Sum = (\overline{A_{in}} \rightarrow B_{in}) \rightarrow \overline{C_{in}}$
7	$C_{in} = 0$	$FALSE(C_{in})$
8	$A_{in}{}'' \rightarrow C_{in} = C_{in}{}'$	$C_{out} = \overline{Sum}$

adders, are presented with check marks, and the inexact ones are represented with cross marks. The logical equations of the fourth SIAFA's outputs are:

$$Sum_{SIAFA4} = (\overline{A_{in}} \rightarrow B_{in}) \rightarrow \overline{C_{in}} \equiv \overline{C_{in}} + \overline{A_{in}}.\overline{B_{in}} \quad (7)$$

$$C_{out_{SIAFA4}} = \overline{Sum_{SIAFA4}} \quad (8)$$

The proposed approximate full adder's output values can be calculated in eight steps using the algorithm shown in Table IX.

F. Summary

Table X shows an overview of the proposed adders and their properties. Even though they look similar in various metrics, as we will see in the upcoming sections, they lead to different properties and performance when used in more complex circuits and applications.

IV. CIRCUIT SIMULATION RESULTS AND COMPARISON

A. Simulation setup

The proposed approximate full adders were simulated in LTSPICE to validate the functionality of their algorithms. The Voltage ThrEshold Adaptive Memristor (VTEAM) model was

TABLE X
SUMMARY OF THE PROPOSED APPROXIMATE FULL ADDERS AND THEIR ERRORS

Approximate full adder	No. of steps	No. of memristors	ED	MED	NMED
SIAFA1	8	4	3	0.375	0.125
SIAFA2	10	5	4	0.5	0.166
SIAFA3	8	4	3	0.375	0.125
SIAFA4	8	4	3	0.375	0.125

TABLE XI
VTEAM MODEL AND SETUP VALUES [17]

Parameter	v_{off}	v_{on}	α_{off}	α_{on}	R_{off}	R_{on}
Value	0.7 V	-10 mV	3	3	1 MΩ	10 kΩ
k_{on}	k_{off}	w_{off}	w_{on}	w_C	a_{off}	a_{on}
-0.5 nm/s	1 cm/s	0 nm	3 nm	107 pm	3 nm	0 nm

TABLE XII
IMPLY LOGIC PARAMETER VALUES [17]

Parameter	Value
V_{SET}	1 V
V_{RESET}	1 V
V_{COND}	900 mv
R_G	40 $K\Omega$
t_{pulse}	30 μs

applied, and the values of its parameters are shown in Table XI [17]. As mentioned in Section II, IMPLY logic has special parameters, and the used value of these parameters are inserted in Table XII [17]. Since [18] and [28] are the most relevant references for comparison, we have simulated them using the same setup.

B. Circuit-level simulation and comparison

1) Simulation results: Section III describes the logic equations and algorithms of the SIAFAs in detail. Circuit-level simulation ensures that the proposed algorithms and circuits operate correctly with the mentioned parameters in all the input cases of a full adder. Each step of the simulation is 30μs. The Sum of the first, third, and fourth proposed approximate full adders (SIAFA1, 3, and 4) is calculated in the sixth step between 150μs-180μs and the C_{out} of these proposed circuits is calculated in the last step between 210μs-240μs. In SIAFA1 and SIAFA4, Sum has stored in memristor A_{in}, and in SIAFA2 and SIAFA3, Sum output is stored in memristor B_{in}. The C_{out} of SIAFA1-4 is stored in the C_{in} memristor.

All eight states of the truth table of the proposed circuits (SIAFA1-4) were simulated and led to correct results. The output waveforms of the SIAFA1, 3, and 4 for two different inputs with exact and inexact outputs, as showcased, are shown in Fig. 6, Fig. 7, and Fig. 8. The output waveforms of the second proposed approximate full adder for two different inputs ($A_{in}B_{in}C_{in}$="001" and "110") as two samples are shown in Fig. 9. The C_{out} of this proposed circuit is calculated in the sixth step (150μs-180μs), and the Sum is calculated in the last step between 270μs-300μs.

To ensure the proper circuit-level functionality of the proposed approximate full adders in 8-bit Ripple Carry Adder (RCA), we did SPICE simulations according to the architecture of Fig. 10. For each proposed approximate full adder, two random 8-bit numbers were added, and the outputs were calculated correctly.

2) Circuit-level comparison: We compare the proposed approximate full adders and the exact full adders presented in [18] and [28] as two of the fastest, low-power, and compact IMPLY-based exact serial full adders, using different circuit-

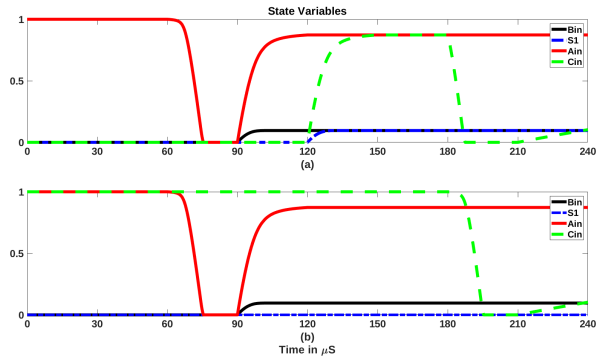

Fig. 6. SIAFA1's simulation results: (a) $A_{in}B_{in}C_{in}$="100", and (b) $A_{in}B_{in}C_{in}$="101"

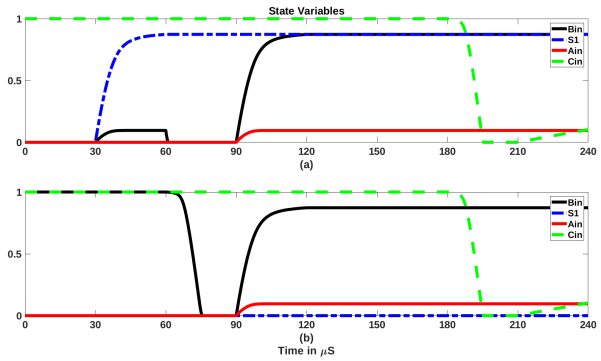

Fig. 7. SIAFA3's simulation results: (a) $A_{in}B_{in}C_{in}$="001", and (b) $A_{in}B_{in}C_{in}$="011"

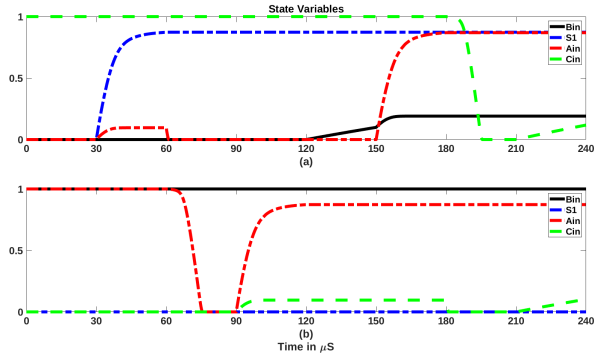

Fig. 8. SIAFA4's simulation results: (a) $A_{in}B_{in}C_{in}$="001", and (b) $A_{in}B_{in}C_{in}$="110"

level criteria. Energy consumption is one of the crucial circuit-level criteria, and the proposed approximate full adders' energy consumption was calculated using the LTSPICE energy calculation tool. First, each memristor's energy consumption for each input combination of full adder is calculated. Then, for each state of the full adder, the energy consumption values of all involved memristors were added together. Finally, the average value of all states was considered as the energy consumption of a full adder. The energy consumption values of SIAFAs and the exact full adders [18], [28] obtained from the simulation, along with the energy consumption improvement percentage of approximate circuits compared to the exact full adders [18], [28], are shown in Table XIII.

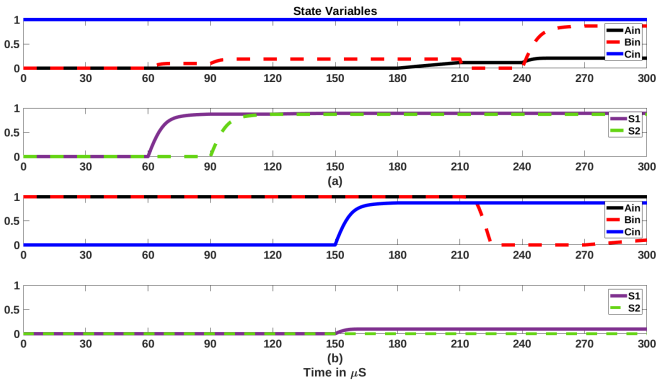

Fig. 9. SIAFA2's simulation results: (a) $A_{in}B_{in}C_{in}$="001", and (b) $A_{in}B_{in}C_{in}$="110"

TABLE XIII
ENERGY CONSUMPTION RESULTS OF SINGLE-BIT SIAFAS AND THE EXACT FULL ADDER CELLS IN [18], [28]

Serial full adder	Energy consumption ($\times 10^{-9} J$)	An improvement in comparison over [18]	An improvement in comparison over [28]
Exact 1 [18]	1.8531	-	-
Exact 2 [28]	1.9518	-	-
SIAFA1	0.6444	65%	67%
SIAFA2	0.8049	56%	59%
SIAFA3	0.6444	65%	67%
SIAFA4	0.6431	66%	68%

The second proposed circuit consumes 160 pJ more energy than the other three proposed circuits. The energy consumption of the first, third, and fourth approximate full adders is almost equal. The fourth circuit consumes one pJ less energy than the other two circuits. Even though this is a small value, it corresponds to only one-bit addition. In multi-bit additions and once thousands and mi1lions of addition (which is very common in many algorithms nowadays) are performed, this number will make a more substantial difference. The proposed approximate full adders have improved energy consumption compared to the exact full adder from [18] by 56% to 66% and compared to the exact full adder from [28] by 59% to 68%.

The number of steps and memristors needed to calculate the final results are other circuit-level criteria that should be analyzed. The number of steps and memristors required to implement the proposed circuits can be calculated based on the algorithms introduced in Section III. The first, third, and fourth proposed approximate full adders calculate the final result in eight steps. In the first version of our research (the first version of SIAFA1), presented in [16], two work memristors are needed, and Sum and C_{out}'s results were written in the work memristors. Based on the proposed algorithms of these three circuits (SIAFA1, 3, and 4), only one work memristor is needed for the correct functionality of the proposed approximate full adders by reusing the input memristors.

The number of memristors and calculation steps for the proposed adders and the most competitive previous works are inserted in Table XIV. They were compared in five different scenarios. In these scenarios, the least significant exact full

TABLE XIV
COMPARISON OF THE NUMBER OF STEPS AND MEMRISTORS OF SINGLE-BIT SIAFAS AND THE EXACT FULL ADDER CELLS IN [18], [28]

Serial full adder	No. of steps		No. of memristors	
	n	n=8-bit	n	n=8-bit
Exact 1 [18]	22n	176	2n+3	19
Exact 2 [28]	23n	184	2n+3	19
SIAFA 1, 3, and 4 (Five different scenarios)	8(n-7)+22(n-1)	162	2n+3	19
	8(n-6)+22(n-2)	148	2n+3	19
	8(n-5)+22(n-3)	134	2n+3	19
	8(n-4)+22(n-4)	120	2n+3	19
	8(n-3)+22(n-5)	106	2n+3	19
SIAFA 2 (Five different scenarios)	10(n-7)+22(n-1)	164	2n+3	19
	10(n-6)+22(n-2)	152	2n+3	19
	10(n-5)+22(n-3)	140	2n+3	19
	10(n-4)+22(n-4)	128	2n+3	19
	10(n-3)+22(n-5)	116	2n+3	19

adders are approximated one by one from the first bit to the fifth bit (Scenarios 1-5) applying the proposed adders. We have chosen this number of approximated bits because the experiments showed that higher numbers lead to undesirable degradation in results.

The number of required memristors was reduced by applying a new algorithm for the design [16]. In the most significant bits of the RCA structure, the exact full adder [18] was utilized, so two work memristors are needed in the 8-bit approximate structures. Therefore, the required memristors in all structures include $2n$ memristors for input bits and three memristors, including the C_{in} memristor and two work memristors. In [16], the number of memristors required increases as the number of the approximate full adders increases in the least significant bits. By changing the implementation algorithm in this article, this concern is addressed.

By applying the proposed approximate full adders in the 8-bit RCA architecture, the number of calculation steps in the mentioned approximate structures (five different scenarios) was reduced from 7% to 43% compared to the exact structures.

The number of computational steps can be expanded for larger approximate RCAs. For example, the number of computational steps for an n-bit structure, including (m1) exact full adders in the most significant bits and (m2) approximate full adders in the least significant bits, can be calculated based on (9) for the first, third, and fourth proposed cells, and (10) for the second proposed cell.

$$No.\ of\ steps = 8(n - m1) + 22(n - m2) \quad (9)$$

$$No.\ of\ steps = 10(n - m1) + 22(n - m2) \quad (10)$$

C. Error analysis

1) 8-bit error analysis: Common error evaluation criteria were introduced in Section II of the article. In approximate computing, error evaluation criteria should also be assessed in addition to circuit evaluation criteria like power consumption.

Behavioral MATLAB simulations were applied to evaluate the error metrics. All the 65536 possible inputs have been applied to several 8-bit approximate adder structures based on the mentioned scenarios in Section IV-B. The approximate adder structures are evaluated so that the error created and propagated in the 8-bit RCA structure evaluated first has the minimum value, and the last one has the maximum value. That is, the first least significant exact full adder was replaced by the proposed approximate full adders and evaluated, then the second exact full adder, and so on. This process was continued by replacing the exact full adders with the approximate full adders until the fifth significant ones (See Fig. 10). By applying this method, the total ED changed from the lowest to the highest value. Conventional error evaluation criteria were calculated, and the results can be seen in Table XV.

TABLE XV
ERROR METRICS IN 8-BIT RCA

Approximate full adder	MED	NMED	MRED
8-bit Error analysis			
Scenario 1: seven most significant full adders are exact.			
SIAFA1, and 3	0.25	0.0004	0.0013
SIAFA2	0.25	0.0004	0.0013
SIAFA4	0.5	0.0009	0.0027
Scenario 2: six most significant full adders are exact.			
SIAFA1, and 3	0.875	0.0017	0.0048
SIAFA2	1	0.0019	0.0055
SIAFA4	1.25	0.0024	0.0068
Scenario 3: five most significant full adders are exact.			
SIAFA1, and 3	2.062	0.004	0.0115
SIAFA2	2.656	0.0052	0.015
SIAFA4	2.625	0.0051	0.0145
Scenario 4: four most significant full adders are exact.			
SIAFA1, and 3	4.351	0.0085	0.0248
SIAFA2	6.1718	0.0121	0.0359
SIAFA4	5.3125	0.0104	0.0299
Scenario 5: three most significant full adders are exact.			
SIAFA1, and 3	8.8554	0.0173	0.0522
SIAFA2	13.498	0.0264	0.0822
SIAFA4	10.6562	0.0208	0.0616

TABLE XVI
ERROR METRICS IN 16-BIT RCA

Approximate full adder	MED	NMED	MRED
16-bit Error analysis			
Scenario 1: Fourteen most significant full adders are exact.			
SIAFA1, and 3	0.892	0.0000068	0.0000187
SIAFA2	1.0063	0.0000077	0.0000207
SIAFA4	1.2591	0.0000096	0.0000269
Scenario 2: Twelve most significant full adders are exact.			
SIAFA1, and 3	4.3506	0.000032	0.0000928
SIAFA2	6.0371	0.000046	0.0001285
SIAFA4	5.3103	0.00004	0.0001136
Scenario 3: Ten most significant full adders are exact.			
SIAFA1, and 3	17.8939	0.00014	0.0003812
SIAFA2	29.2887	0.00022	0.0006318
SIAFA4	21.0881	0.00016	0.0004521
Scenario 4: Eight most significant full adders are exact.			
SIAFA1, and 3	72.0983	0.00055	0.0016
SIAFA2	122.1056	0.00093	0.0026
SIAFA4	85.42	0.00065	0.0019
Scenario 5: Six most significant full adders are exact.			
SIAFA1, and 3	285.1531	0.0022	0.0063
SIAFA2	495.2381	0.0038	0.0109
SIAFA4	364.2438	0.0026	0.0075

Based on Table XV, the error of the first and third SIAFAs are equal. Table X shows that SIAFA1 and SIAFA3 have the same single-bit error evaluation criteria (ER, ED, and MED). To understand the main reason for the same error evaluation criteria of the SIAFA1 and SIAFA3 in Table X, we should focus on the truth table of these two full adders (SIAFA1 and

TABLE XVII
ERROR METRICS IN 32-BIT RCA

Approximate full adder	MED	NMED	MRED
32-bit Error analysis			
Scenario 1: Twenty four most significant full adders are exact.			
SIAFA1, and 3	71.3096	≈ 0	≈ 0
SIAFA2	122.065	≈ 0	≈ 0
SIAFA4	85.1628	≈ 0	≈ 0
Scenario 2: Sixteen most significant full adders are exact.			
SIAFA1, and 3	18297	0.0000021	0.0000059
SIAFA2	35041	0.000004	0.0000115
SIAFA4	21862	0.0000025	0.000007
Scenario 3: Eight most significant full adders are exact.			
SIAFA1, and 3	4773200	0.00055	0.0015
SIAFA2	8897700	0.001	0.0029
SIAFA4	5630000	0.00067	0.0018

SIAFA3). According to the truth table of SIAFA1 and SIAFA3, if states 0 ($A_{in}B_{in}C_{in}$="000") and 7 ($A_{in}B_{in}C_{in}$="111") occur, only the Sum outputs of SIAFA1 and SIAFA3 are incorrect, while the C_{out} of both circuits are correct. If we concentrate on state 5 ($A_{in}B_{in}C_{in}$="101") in SIAFA1 and state 3 ($A_{in}B_{in}C_{in}$="011") in SIAFA3, we understand that C_{out} is calculated incorrectly. If these two conditions occur in $(n-1)^{th}$ bit of RCA, the Sum of this full adder is calculated incorrectly. By propagating the wrong C_{out} in both circuits to the n^{th} full adder of RCA (the applied approximate full adder is the same as the previous bit), two states occur: First, states 2 ($A_{in}B_{in}C_{in}$="010"), 4 ($A_{in}B_{in}C_{in}$="100"), or 6 ($A_{in}B_{in}C_{in}$="110"), and both outputs of the full adder are exact and, Second, state 0 ($A_{in}B_{in}C_{in}$="000"), where the Sum output is inexact, while the C_{out} is calculated exactly. So propagation of error and creation of error are the same in both proposed circuits. Because the probability distribution of the inputs in this simulation is equal, the errors created and propagated in the approximate 8-bit RCA structure, which is made of SIAFA1 and SIAFA3, are the same based on the error analysis of the previous paragraph. All we have said here is also valid for evaluating 16 and 32-bit error analysis metrics. Also, the error analysis metrics of these two proposed circuits are less than the other two proposed circuits. The error values of the fourth proposed circuit are less than the second proposed circuit, rendering SIAFA2 the least precise adder among the proposed adders. We know that error and energy consumption are often inversely proportional [12]. Therefore, evaluating a metric such as NMED can help to make a trade-off between energy consumption and accuracy [12]. According to Table XV, the first and third proposed circuits have the lowest NMED compared to the other approximate designs. The second approximate full adder has the lowest computation accuracy in the mentioned architectures. This circuit's ER is less than the other three proposed circuits, but the errors (total ED) created in the 8-bit architecture are more than in the other circuits. Therefore, only single-bit ER evaluation is insufficient to evaluate and predict the errors created in larger computing structures. We plan to investigate the closed-form analyses of the impact of incorrect carry bits on the most significant bits, and overall adder and the prediction of the total ED in our future research.

In order to evaluate the accuracy, energy consumption, and the number of calculation steps (delay) together, we use a FOM defined as

$$FOM = \frac{Energy \times Number\ of\ steps}{1 - NMED}, \tag{11}$$

which gives equal importance to energy consumption, speed (number of steps), and error (1-NMED). The structure shown in Fig. 10 was used to calculate FOM. The results of FOM are seen in Table XVIII. The first and third proposed circuits have 17% and 1% smaller FOM than the second and fourth circuits. Therefore, the first and third proposed circuits in the structure of Fig. 10 created a better compromise between the error and circuit evaluation criteria. The total energy consumption estimation for calculating the FOM in n-bit RCA includes (m1) exact full adders, and (m2) approximate full adders are calculated based on

$$Total\ Energy\ Consumption = \ Energy\ consumption\ of\ each\ SIAFA \times (n-m1)nJ + 1.8531 \times (n-m2)nJ. \tag{12}$$

TABLE XVIII
THE FOM RESULTS OF THE SIAFAS IN THE STRUCTURE OF FIG. 10

Approximate applied full adder	FOM
SIAFA1	947.204
SIAFA2	1141.866
SIAFA3	947.204
SIAFA4	949.886

2) 16 and 32-bit error analysis: One million random numbers with uniform input distribution have been considered to calculate the error metrics in the 16 and 32-bit RCA. In the first scenario, approximate full adders are applied in the two least significant digits of the 16-bit RCA architecture, and the rest are exact full adders. In the second scenario, four approximate full adders are applied in the 16-bit RCA architecture, and the remaining bits are exact full adders. In the third scenario, six approximate full adders are in the least significant bits, and the remaining are exact full adders. In the fourth and fifth scenarios, eight and ten approximate full adders are applied in the least significant bits, respectively, and the rest are exact full adders. In the 32-bit RCA, 8, 16, and 24 least significant full adders were considered approximate cells, and the remaining cells are exact full adders (scenarios 1-3). Error evaluation criteria were calculated based on these 16 and 32-bit scenarios. The results are reported in Tables XVI and XVII.
With the increase in the number of approximate full adders, all the error analysis metrics increase by examining the reported results. Based on the results reported in Tables XV, XVI and XVII and with the uniform input distribution, it can be estimated that by replacing one more approximate full adder with the exact one in the approximate RCA structure, the ED, MED, NMED, and MRED approximately doubles. Based on this fact and the results reported in Tables XV, XVI and XVII, it is possible for the researchers to apply the proposed SIAFAs in the 8, 16, and 32-bit adder structures and design

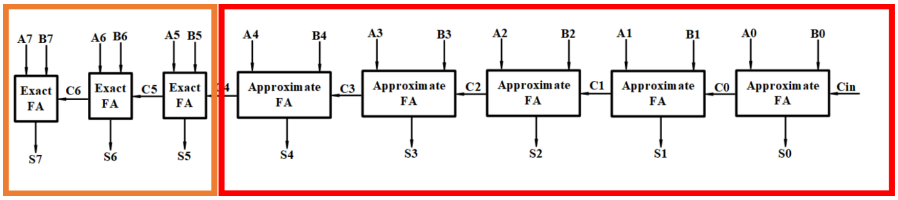

Fig. 10. The structure applied to calculate error metrics, FOM, and image quality metrics [1]

their desired systems based on the amount of error that they can tolerate.

V. Application-level Simulation and Comparison

Image processing is a pervasive application that plays a vital role in daily human lives, e.g., it is applied in daily used personal machine vision software, robotics, industrial systems, and media [41]. Here, we investigated the proposed approximate circuits in image processing to ensure that the output result quality is acceptable. The proposed circuits were evaluated in image addition, image subtraction, grayscale filter application, and image multiplication (blending). In addition, a behavioral description of the proposed adders in MATLAB was used to simulate image processing applications. The applied structures to simulate the image processing applications (image addition, motion detection, and grayscale filter) are the approximate 8-bit RCA architectures (scenarios 1-5) introduced in subsection IV-B2.

1) Image addition: Image addition is one of the basic image processing applications used for masking and enhancing images [33]. In image addition, the corresponding pixels of each main images are added together pixel by pixel. For example, two 256 × 256 grayscale images were added together by the approximate structures described in subsection IV-B2. The results of image quality analysis metrics can be seen in Table XIX for all scenarios and Fig. 11 for the scenario according to the architecture of Fig. 10.

Based on Table XIX, by applying the first, third, and fourth proposed circuits in the approximate structures according to scenarios 1-5, the output images of the image addition application have acceptable quality. Their PSNR is higher than 30 dB, commonly considered an acceptable image quality in the literature [12], [35]. By applying the second proposed circuit in the approximate structures according to scenarios 1-4, the quality of the output images is acceptable, and the PSNR is higher than 30 dB. By applying this circuit in the structure of Fig. 10 (scenario 5), PSNR equals 28.2504 dB. According to the PSNR criterion, the image quality is less than 30 dB. However, the loss of quality compared to the images produced by other proposed circuits (SIAFA1, 3, and 4) is not significant considering human visual perception. Consider that in another scenario, the sixth least significant full adder is approximated, and the exact cell is replaced by the proposed cells (only the two most significant full adders are exact). In this scenario, the PSNR criterion for the SIAFA1-4 is 27.49 dB, 23.1 dB, 27.14 dB, and 25.88 dB, respectively. Therefore, the output images achieved from MATLAB simulation in this scenario are not of good quality, and this approximate structure is unsuitable for image addition application.

TABLE XIX
Image Quality Metrics in Different Scenarios (Image Addition)

Approximate full adder	PSNR (dB)	SSIM	MSSIM
Scenario 1: seven most significant full adders are exact.			
SIAFA1	54.0958	0.9991	0.9991
SIAFA2	54.0958	0.9991	0.9991
SIAFA3	54.0958	0.9991	0.9991
SIAFA4	54.0958	0.9991	0.9991
Scenario 2: six most significant full adders are exact.			
SIAFA1	49.78	0.9972	0.9972
SIAFA2	48.6645	0.997	0.997
SIAFA3	49.7593	0.9972	0.9972
SIAFA4	49.3735	0.9967	0.9967
Scenario 3: five most significant full adders are exact.			
SIAFA1	44.5148	0.9899	0.99
SIAFA2	41.9674	0.9858	0.9861
SIAFA3	44.5222	0.9898	0.99
SIAFA4	43.7483	0.9878	0.988
Scenario 4: four most significant full adders are exact.			
SIAFA1	38.67	0.9644	0.9649
SIAFA2	35.4576	0.9425	0.9436
SIAFA3	38.8399	0.9638	0.9644
SIAFA4	37.8083	0.959	0.9597
Scenario 5: three most significant full adders are exact.			
SIAFA1	32.9823	0.8974	0.8996
SIAFA2	28.2504	0.8156	0.8166
SIAFA3	32.6497	0.8905	0.8915
SIAFA4	32.0442	0.8931	0.8956

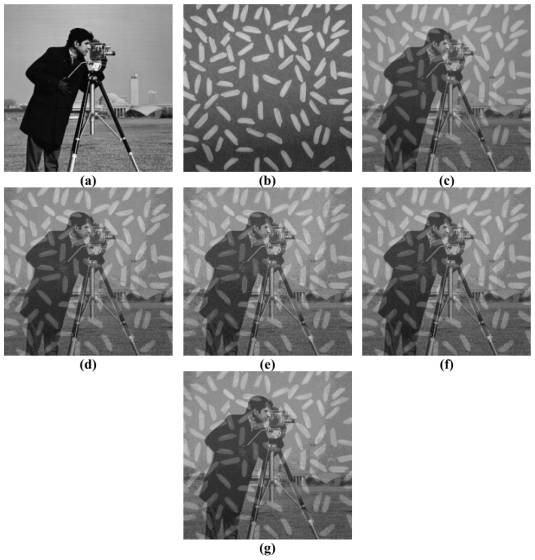

Fig. 11. The results of image addition simulation based on the architecture of Fig. 10 (scenario 5): (a) Cameraman, (b) Rice, (c) Exact output, (d) SIAFA1, (e) SIAFA2, (f) SIAFA3, and (g) SIAFA4

TABLE XX
IMAGE QUALITY METRICS IN DIFFERENT SCENARIOS (IMAGE SUBTRACTION)

Approximate full adder	PSNR (dB)	SSIM	MSSIM
Scenario 1: seven most significant full adders are exact.			
SIAFA1	53.1973	0.9819	0.9851
SIAFA2	54.7249	0.9962	0.999
SIAFA3	53.2099	0.9815	0.9845
SIAFA4	58.8859	0.9974	0.9985
Scenario 2: six most significant full adders are exact.			
SIAFA1	46.7004	0.9122	0.9227
SIAFA2	48.8606	0.9860	0.9953
SIAFA3	46.8256	0.9159	0.9255
SIAFA4	50.7763	0.9827	0.9888
Scenario 3: five most significant full adders are exact.			
SIAFA1	41.5919	0.7711	0.7901
SIAFA2	43.5407	0.9678	0.9874
SIAFA3	41.8705	0.7905	0.8103
SIAFA4	45.5121	0.9534	0.9693
Scenario 4: four most significant full adders are exact.			
SIAFA1	37.4131	0.6405	0.6705
SIAFA2	37.5605	0.9338	0.9652
SIAFA3	36.8613	0.5993	0.6302
SIAFA4	40.3861	0.9131	0.9423
Scenario 5: three most significant full adders are exact.			
SIAFA1	32.6121	0.508	0.5404
SIAFA2	31.6441	0.8991	0.9265
SIAFA3	32.4096	0.4747	0.5094
SIAFA4	35.0436	0.8664	0.902

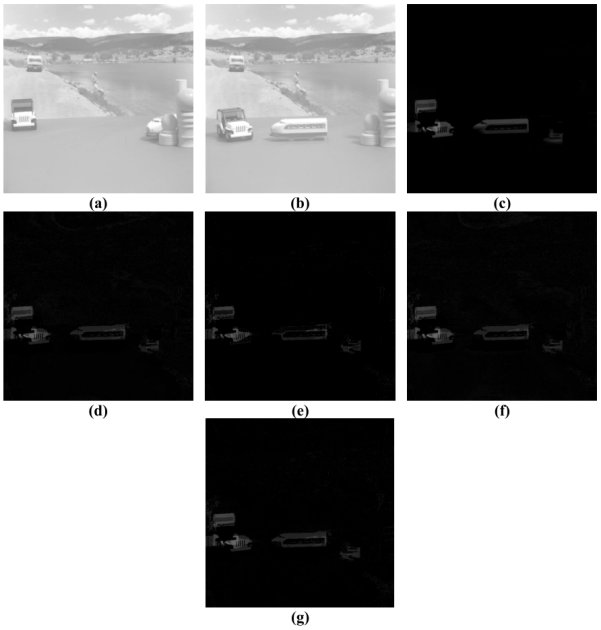

Fig. 12. The results of image subtraction simulation based on the architecture of Fig. 10 (scenario 5): (a) First image [42], (b) Second image [42], (c) Exact output, (d) SIAFA1, (e) SIAFA2, (f) SIAFA3, and (g) SIAFA4

2) Image subtraction: Subtractors can be designed using two's complement addition. The subtraction result of two images is created from the subtraction of the corresponding pixels of the two images, just like the addition of images. Image subtraction is frequently used for motion detection [7]. Fig. 12 illustrates the image subtraction results based on the architecture of Fig. 10.

The results of image subtraction simulation using the proposed approximate circuits for five scenarios of the 8-bit approximate RCA can be seen in Table XX. All four proposed approximate circuits in this application and these five scenarios have produced acceptable outputs, and their PSNR is higher than 30 dB in all cases. In another 8-bit approximate RCA scenario, the two most significant full adders are exact, and the proposed cells are placed in the other six least significant ones. In that scenario, the image quality criterion in all four structures is higher than 27.5 dB. However, the output images lack enough quality, and the "detected" motion is not visible. So, the structures applied in scenarios 1-5 are the acceptable architectures for this application.

TABLE XXI
IMAGE QUALITY METRICS IN DIFFERENT SCENARIOS (GRAYSCALE FILTER)

Approximate full adder	PSNR (dB)	SSIM	MSSIM
Scenario 1: seven most significant full adders are exact.			
SIAFA1	57.4443	0.9993	0.9999
SIAFA2	57.4443	0.9993	0.9999
SIAFA3	57.4443	0.9993	0.9999
SIAFA4	54.1607	0.9984	0.9998
Scenario 2: six most significant full adders are exact.			
SIAFA1	52.4811	0.9974	0.9997
SIAFA2	50.3565	0.9966	0.9995
SIAFA3	52.7460	0.9976	0.9997
SIAFA4	49.0616	0.9955	0.9993
Scenario 3: five most significant full adders are exact.			
SIAFA1	47.1982	0.9911	0.999
SIAFA2	43.1339	09833	0.9977
SIAFA3	47.2496	0.9914	0.999
SIAFA4	43.0565	0.9841	0.997
Scenario 4: four most significant full adders are exact.			
SIAFA1	41.4201	0.9693	0.9957
SIAFA2	35.9998	0.9263	0.9874
SIAFA3	41.2315	0.9684	0.9956
SIAFA4	36.9634	0.9451	0.989
Scenario 5: three most significant full adders are exact.			
SIAFA1	35.5671	0.9019	0.9778
SIAFA2	28.4883	0.7576	0.9317
SIAFA3	35.3588	0.8916	0.9794
SIAFA4	31.5146	0.8525	0.9589

3) Grayscale filter: The grayscale filter is a filter to convert RGB images to grayscale. Each pixel of the RGB image consists of three values (Red, Green, and Blue). By adding the R, G, and B values of each pixel and dividing it by 3, a value between 0 and 255 is obtained, which is the way to convert RGB pixels to grayscale ones. Table XXI and Fig. 13 present the simulation results of this application.

According to the simulation results of the second proposed circuit in scenarios 1-4, the quality of output images in all four scenarios is acceptable since the respective PSNR is more than 30 dB. When the second proposed circuit is placed in the fifth least significant full adder of the 8-bit RCA instead of the exact full adder (See Fig. 10), the quality of the output image decreases from 30 dB to 28.48 dB. By comparing this output image with other output images in this scenario, it can be concluded that it is acceptable. The first, third, and fourth SIAFAs have acceptable outputs in all scenarios. It should be noted that the divider applied to calculate the results is exact.

The proposed circuits (SIAFAs 1-4) were evaluated in a grayscale filter application on a 16-bit RGB image based on the scenarios (1-5) mentioned in subsection IV-C2. The image quality metrics and output results are reported in Table XXII and Fig. 14. Based on the results of simulations, when the

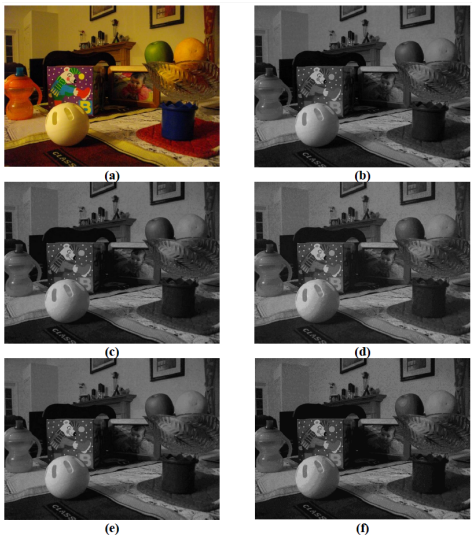

Fig. 13. The results of grayscale filter simulation based on the architecture of Fig. 10 (scenario 5): (a) RGB image (standard MATLAB image), (b) Exact output, (c) SIAFA1, (d) SIAFA2, (e) SIAFA3, and (f) SIAFA4

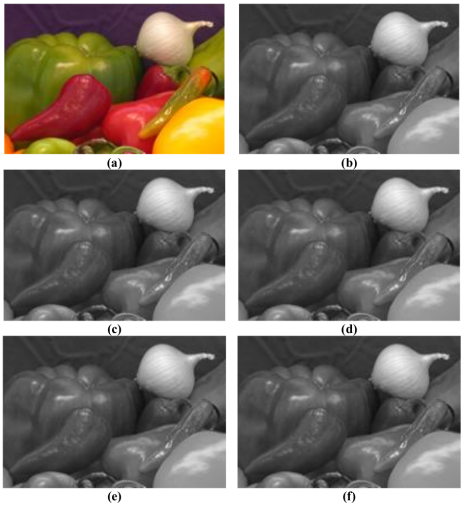

Fig. 14. The results of 16-bit grayscale filter simulation. The SIAFAs are placed in the 10 least significant bits of the RCA, and the other 6 bits are made up of exact full adders: (a) 16-bit RGB image (standard MATLAB image), (b) Exact output, (c) SIAFA1, (d) SIAFA2, (e) SIAFA3, and (f) SIAFA4

TABLE XXII
IMAGE QUALITY METRICS IN DIFFERENT SCENARIOS (16-BIT GRAYSCALE FILTER)

Approximate full adder	PSNR (dB)	SSIM	MSSIM
Scenario 1: fourteen most significant full adders are exact.			
SIAFA1	100.6735	1	1
SIAFA2	98.6093	1	1
SIAFA3	100.6704	1	1
SIAFA4	97.1389	1	1
Scenario 2: twelve most significant full adders are exact.			
SIAFA1	89.6305	1	1
SIAFA2	84.2986	1	1
SIAFA3	89.2601	1	1
SIAFA4	85.1007	1	1
Scenario 3: ten most significant full adders are exact.			
SIAFA1	77.4907	1	1
SIAFA2	70.3889	1	0.9999
SIAFA3	77.7544	1	1
SIAFA4	73.0714	1	1
Scenario 4: eight most significant full adders are exact.			
SIAFA1	66.8283	1	1
SIAFA2	55.7789	0.9999	1
SIAFA3	67.2283	1	1
SIAFA4	63.4844	1	1
Scenario 5: six most significant full adders are exact.			
SIAFA1	53.8573	0.9983	0.9918
SIAFA2	45.9466	0.995	0.9788
SIAFA3	54.0663	0.9984	0.9923
SIAFA4	49.3622	0.9971	0.9868

TABLE XXIII
THE AVERAGE VALUES OF IMAGE QUALITY METRICS FOR 8-BIT IMAGE PROCESSING APPLICATIONS IN DIFFERENT SCENARIOS

Approximate full adder	PSNR (dB)	SSIM	MSSIM
Scenario 1: seven most significant full adders are exact.			
SIAFA1	54.9124	0.9934	0.9947
SIAFA2	55.4216	0.9982	0.9993
SIAFA3	54.9166	0.9933	0.9945
SIAFA4	55.7141	0.9983	0.9991
Scenario 2: six most significant full adders are exact.			
SIAFA1	49.6538	0.9689	0.9732
SIAFA2	49.2938	0.9932	0.9972
SIAFA3	49.7769	0.9702	0.9741
SIAFA4	49.7371	0.9916	0.9949
Scenario 3: five most significant full adders are exact.			
SIAFA1	44.4349	0.9173	0.9263
SIAFA2	42.8806	0.9789	0.9904
SIAFA3	44.5474	0.9239	0.9331
SIAFA4	44.1056	0.9751	0.9847
Scenario 4: four most significant full adders are exact.			
SIAFA1	39.1677	0.858	0.877
SIAFA2	36.3393	0.9342	0.9654
SIAFA3	38.9775	0.8438	0.8634
SIAFA4	38.3859	0.939	0.9636
Scenario 5: three most significant full adders are exact.			
SIAFA1	33.7205	0.7691	0.8059
SIAFA2	29.4609	0.8241	0.8916
SIAFA3	33.4727	0.7522	0.7934
SIAFA4	32.8674	0.8706	0.9188

SIAFAs are placed in the ten least significant bits of the RCA, and the other six bits are made up of exact full adders (scenario 5), the output images have good image quality metrics. This structure can reduce the hardware complexity compared to the exact RCA structure, increase efficiency, and reduce the image quality to an acceptable level. If the proposed approximate full adders are applied to the twelve least significant full adders of the 16-bit RCA structure, the PSNR criterion is greater than 30 dB, but the shade is formed on the output images.

The quality of the output images obtained from the approximate adder structures depends on input data distribution. So, the average values of the image quality metrics of 8-bit image processing applications (image addition, image subtraction, and grayscale filter) are written in Table XXIII. This table can give a better overview of the application of the proposed approximate circuits in the 8-bit RCA in image processing applications.

4) Image multiplication (blending): Image multiplication is one of the important image processing operations [12]. In image multiplication, an 8-bit grayscale image is multiplied by the corresponding pixel of another 8-bit grayscale image. The final result is an 8-bit grayscale image obtained by scaling back the multiplication result to 8-bit [43]. An unsigned 8-bit array multiplier was chosen to simulate image multiplication. The architecture of the array multiplier is explained in [44].

TABLE XXIV
IMAGE QUALITY METRICS IN DIFFERENT SCENARIOS (IMAGE MULTIPLICATION)

Approximate full adder	PSNR (dB)	SSIM	MSSIM
Structure 8			
SIAFA1	45.3825	0.9827	0.9826
SIAFA2	38.2294	0.9509	0.9498
SIAFA3	39.2544	0.9578	0.9568
SIAFA4	42.7596	0.983	0.9829
Structure 9			
SIAFA1	39.7229	0.9495	0.9494
SIAFA2	32.28	0.8877	0.8855
SIAFA3	34.7659	0.903	0.902
SIAFA4	37.9907	0.9561	0.9558
Structure 10			
SIAFA1	34.2596	0.8868	0.8876
SIAFA2	26.9145	0.7768	0.7742
SIAFA3	28.9314	0.787	0.7849
SIAFA4	33.4718	0.8978	0.8977
Structure 11			
SIAFA1	29.5667	0.8004	0.7998
SIAFA2	21.4033	0.6337	0.6311
SIAFA3	25.2185	0.708	0.7076
SIAFA4	28.5698	0.795	0.7932

In the 8×8 unsigned array multiplier structure, consisting of 16 columns, the primary computing, the And Partial Product (APP) cells, are placed in columns 2-15 because in the first column, there is only an AND gate, and in the last column the C_{out} is transferred to column 16. Beginning from the second column, first, the exact full adder of the APP is replaced by the proposed approximate full adders, where the APP cells in the remaining columns are exact, which is called structure 1. Following this, image multiplication is performed, and image quality metrics are computed for this structure. Next, the exact full adders of the APP cells' second and third columns are replaced by approximate full adders, which is called structure 2, and the rest APP cells remain exact. Following this, the image quality metrics are computed. This process is repeated to construct 14 structures. In this process, the quality of the output images became unacceptable for all four proposed circuits when structures 12-14 were applied. The details of approximate image multiplication for columns 2-9 (structure 8), 2-10 (structure 9), 2-11 (structure 10), and 2-12 (structure 11) are tabulated in Table XXIV.

Based on Table XXIV, when the first and fourth proposed approximate full adders were applied in structure 10, the image multiplication outputs had a PSNR above 30 dB. When the second and third approximate full adders were applied in this structure, the PSNR was less than 30 dB. The second and third approximate full adders can be applied to structure 9, and the image multiplication outputs have a PSNR higher than 30 dB. Another structure that can be evaluated for image multiplication is the 8-bit approximate Dadda multiplier [12]. In this multiplier, it is possible to truncate the least significant columns. The partial product in the middle columns of the multiplication tree can be reduced by approximate cells and, in the most significant columns, by exact ones. The comparison of image multiplication simulations based on the approximate array multiplier and the approximate Dadda multiplier can be evaluated in future research.

Fig. 15. The results of image multiplication simulation: (a) Cameraman, (b) Second image [42], (c) Exact output, (d) SIAFA1, (e) SIAFA2, (f) SIAFA3, and (g) SIAFA4

VI. CONCLUSION

This article introduces four memristor-based approximate full adders based on the IMPLY logic. The proposed circuits are designed through the fully serial architecture to minimize the required number of memristors and energy consumption. By applying approximate computing, the number of calculation steps is reduced significantly. The number of memristors and computational steps of these proposed circuits are evaluated in 5 different scenarios. The SIAFAs examined in these scenarios indicate a reduction of 7%-43% in the number of computational steps compared to the exact serial structures. In addition, the proposed cells' energy consumption is improved by up to 68% compared to the exact IMPLY-based fully serial full adders. In multiple scenarios, the SIAFAs are evaluated in RCA by several error evaluation criteria, like the ED, MED, NMED, and MRED in MATLAB. A FOM combining energy, speed, and accuracy is considered to evaluate both circuit evaluation criteria and the accuracy of proposed circuits. These proposed circuits are evaluated in different scenarios and applications of image addition, image subtraction, grayscale filter, and image multiplication. Based on the simulation results, the quality of the output images remains acceptable if SIAFAs 1-4 are applied in the five least significant bits of an 8-bit RCA architecture. In addition, if the ten least significant bits of 16-bit RCA are approximated, the quality of the images is entirely acceptable. To estimate the errors created in the RCA and predict the error and image quality created using RCA approximate adders, extracting and providing a probabilistic model is of concern in future works.

REFERENCES

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

[2] I. Alouani, H. Ahangari, O. Ozturk, and S. Niar, "A novel heterogeneous approximate multiplier for low power and high performance," *IEEE Embedded Systems Letters*, vol. 10, no. 2, pp. 45–48, 2017.

[3] M. Taheri, A. Arasteh, S. Mohammadyan, A. Panahi, and K. Navi, "A novel majority based imprecise 4: 2 compressor with respect to the current and future vlsi industry," *Microprocessors and Microsystems*, vol. 73, p. 102962, 2020.

[4] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing [point of view," *Proceedings of the IEEE*, vol. 108, no. 3, pp. 394–399, 2020.

[5] H. A. D. Nguyen, J. Yu, M. A. Lebdeh, M. Taouil, S. Hamdioui, and F. Catthoor, "A classification of memory-centric computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 16, no. 2, pp. 1–26, 2020.

[6] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2012.

[7] Y. S. Mehrabani, R. F. Mirzaee, Z. Zareei, and S. M. Daryabari, "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*, vol. 26, no. 05, p. 1750082, 2017.

[8] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate xor/xnor-based adders for inexact computing," in *2013 13th IEEE International Conference on Nanotechnology (IEEE-NANO 2013)*, pp. 690–693, IEEE, 2013.

[9] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pp. 343–348, 2015.

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

[11] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[12] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "A majority-based imprecise multiplier for ultra-efficient approximate image multiplication," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 11, pp. 4200–4208, 2019.

[13] C. Ossimitz and N. TaheriNejad, "A fast line segment detector using approximate computing," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2021.

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

[15] M. R. Alam, M. H. Najafi, and N. TaheriNejad, "Sorting in memristive memory," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2022.

[16] S. E. Fatemieh, M. R. Reshadinezhad, and N. TaheriNejad, "Approximate in-memory computing using memristive imply logic and its application to image processing," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2022.

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

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

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

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

[21] M. R. Alam, M. H. Najafi, N. TaheriNejad, and R. Gottumukkala, "Stochastic computing in beyond von-neumann era: Processing bitstreams in memristive memory," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2022.

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

[23] 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, 2013.

[24] 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.

[25] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "The desired memristor for circuit designers," *IEEE Circuits and Systems Magazine*, vol. 13, no. 2, pp. 17–22, 2013.

[26] S. Kvatinsky, *Memristor-based circuits and architectures*. PhD thesis, Technion-Israel Institute of Technology, Faculty of Electrical Engineering, 2014.

[27] K. Alhaj Ali, *New design approaches for flexible architectures and in-memory computing based on memristor technologies*. PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique Bretagne Pays de la Loire, 2020.

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

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

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

[31] X.-Y. Wang, P.-F. Zhou, J. K. Eshraghian, C.-Y. Lin, H. H.-C. Iu, T.-C. Chang, and S.-M. Kang, "High-density memristor-cmos ternary logic family," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 264–274, 2020.

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

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

[34] Z. Yang, J. Han, and F. Lombardi, "Transmission gate-based approximate adders for inexact computing," in *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH´ 15)*, pp. 145–150, IEEE, 2015.

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

[36] F. Karimi, R. F. Mirzaee, A. Fakeri-Tabrizi, and A. Roohi, "Ultra-fast, high-performance 8x8 approximate multipliers by a new multi-column 3, 3: 2 inexact compressor and its derivatives," *arXiv preprint arXiv:2107.11881*, 2021.

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

[38] S. Muthulakshmi, C. S. Dash, and S. Prabaharan, "Memristor augmented approximate adders and subtractors for image processing applications: An approach," *AEU-International Journal of Electronics and Communications*, vol. 91, pp. 91–102, 2018.

[39] S. Muthulakshmi, C. S. Dash, and S. Prabaharan, "Memristor-based approximate adders for error resilient applications," in *Nanoelectronic Materials and Devices*, pp. 51–59, Springer, 2018.

[40] R. Ataie, A. A. Emrani Zarandi, and Y. Safaei Mehrabani, "An efficient inexact full adder cell design in cnfet technology with high-psnr for image processing," *International Journal of Electronics*, vol. 106, no. 6, pp. 928–944, 2019.

[41] M. Khaleqi Qaleh Jooq, M. Ahmadinejad, and M. H. Moaiyeri, "Ultraefficient imprecise multipliers based on innovative 4: 2 approximate compressors," *International Journal of Circuit Theory and Applications*, vol. 49, no. 1, pp. 169–184, 2021.

[42] University of Southern California (USC) Signal and Image Processing Institute (SIPI), *The USC-SIPI Image Database*. https://sipi.usc.edu/database/database.php?volume=sequences.

[43] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. Di Meo, "Comparison and extension of approximate 4-2 compressors for low-power approximate multipliers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 9, pp. 3021–3034, 2020.

[44] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolić, *Digital integrated circuits: a design perspective*, vol. 7. Pearson Education Upper Saddle River, NJ, 2003.

Seyed Erfan Fatemieh was born in Isfahan, Iran, in 1996. He received his B.Sc. degree in Computer Engineering and M.Sc. degree in Computer Architecture from the University of Isfahan, Isfahan, Iran, in 2018 and 2020. He is currently a Ph.D. candidate in Computer Architecture at the University of Isfahan, Isfahan, Iran. His research interests include In-memory Computing, Computer Arithmetic, Digital VLSI, and Quantum Computing and Reversible Circuits.

Mohammad Reza Reshadinezhad was born in Isfahan, Iran, in 1959. He received his B.S. and M.S. degrees from the Electrical Engineering Department of the University of Wisconsin, Milwaukee, USA, in 1982 and 1985, respectively. He has been in the position of lecturer as faculty of computer engineering at the University of Isfahan since 1991. He also received a Ph.D. Degree in computer architecture from Shahid Beheshti University, Tehran, Iran, in 2012. He is currently an Associate Professor in the Faculty of Computer Engineering at the University of Isfahan. His research interests are Digital Arithmetic, Nanotechnology concerning CNTFET, VLSI Implementation, and Cryptography.

Nima TaheriNejad (S'08-M'15) 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, 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 80 articles. Dr. Taherinejad has served as a reviewer and an editor of various journals and conferences. He has also been an organizer and a chair of various conferences and workshops. He has received several awards and scholarships from universities, conferences, and competitions he has attended. This includes the Best University Booth award at DATE 2021, First prize in the 15th Digilent Design Contest (2019) and in the Open-Source Hardware Competition at Eurolab4HPC (2019) as well as Best Teacher and Best Course awards at TU Wien (2020).