Computers and Electrical Engineering 113 (2024) 109053

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Energy-efficient and fast IMPLY-based approximate full adder applying NAND gates for image processing

Samane Asgari, Mohammad Reza Reshadinezhad *, Seyed Erfan Fatemieh

Department of Computer Architecture, Faculty of Computer Engineering, University of Isfahan, Isfahan, 8174673441, Iran

ARTICLE INFO

Keywords:
Memristor
Approximate full adder
Approximate computing
In-memory computing
Image processing
IMPLY logic

ABSTRACT

Today's processing applications require frequent and energy-consuming data transfers between memory and processing elements. Reducing the dimensions of Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) brings essential problems. Therefore, applying emerging technologies should be considered as a solution. As a crossbar array compatible technology, memristors can play the computing and memory device roles. Therefore, it is suitable for In-Memory Computing (IMC). The issue of changing the computation method to achieve higher efficiency can also be considered. Today, approximate computing is of great interest for error-tolerant applications. These two concepts are applied, and an approximate serial full adder based on the Material Implication (IMPLY) method is introduced. The proposed Serial Approximate Full Adder using NAND gates (SAFAN) reduces the number of steps and energy consumption by up to 30% and 26% compared to approximate ones, 70% and 68% compared to exact ones, respectively. The error analysis results show that the accuracy decreases to a reasonable level by applying the SAFAN to the least significant bits (LSBs) of an approximate 8-bit adder. Image analysis criteria evaluated the functionality of the SAFAN in image processing applications, and the results showed the suitable performance of the SAFAN in error-tolerant applications.

1. Introduction

Today, lots of computer systems are built using Von-Neumann architecture. Separating the processing unit and the memory unit leads to a decrease in system efficiency [1]. Data must be exchanged between these two units when performing computations, which requires much time and energy consumption [1]. These problems are known as Von-Neumann's bottleneck or memory wall. This bottleneck is found more in data-intensive applications [2]. Among the research in this field, bringing the data and processing memory units closer physically is suggested as a solution [3]. It should be noted that this solution can only partially eliminate data movements, and there is still a need to move data between these two units.

Computing applications such as image processing, neural networks, speech recognition, and the Internet of Things (IoT) must process large amounts of data and require significant computation [4]. Reducing the size of Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) is also challenging due to the need to increase the transistor count on the same area chip. This dimension reduction has several problems, including increasing subthreshold leakage currents, reducing reliability, and gate control [5,6]. For the reasons mentioned, there is a need for new technologies that can overcome the problems and limitations of Complementary MOS (CMOS) technology [7]. A proposed approach to improve these problems is to consider emerging technologies. Carbon

* Corresponding author.
E-mail address: m.reshadinezhad@eng.ui.ac.ir (M.R. Reshadinezhad).

https://doi.org/10.1016/j.compeleceng.2023.109053
Received 25 September 2023; Received in revised form 19 November 2023; Accepted 22 November 2023
Available online 12 December 2023
0045-7906/

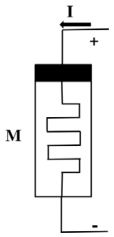

Fig. 1. Memristor symbol [15].

Nanotube Field Effect Transistors (CNFETs) [8], Quantum-dot Cellular Automata (QCA) [7], and memristors are considered emerging technologies. It should be remembered that due to the high volume of data processing and Von-Neumann's bottleneck, appropriate technologies attracted the researchers' attention. These technologies enable the implementation of both the roles of computing and memory elements in one place, in addition to overcoming the limitations of MOSFET technology. This results in avoiding the movements of data. A suitable technology for this purpose is memristors [9].

The characteristics of memristors include simple structure, high-density integration, non-volatile behavior, scalability, and compatibility with today's manufacturing technology. These characteristics of memristors attract the attention of researchers [1,10]. It should be emphasized that one of the most essential features of the memristor is its non-volatile behavior. It means that when no voltage or current is applied, the memristor can maintain its last resistance value until the voltage or current is applied to it again. These characteristics have made memristors a promising candidate to replace today's widespread memories, such as flash memory and DRAM [1,11].

Memristors in stateful design methods, such as the Material Implication (IMPLY) design method, are consistent with the crossbar array. Thus, they can be used to implement In-Memory Computing (IMC) [12]. IMC means performing computations in memory [9]. Among the advantages of this type of operation are reducing the energy required for reading and writing data, transferring data to the processing unit, enabling extensive parallelization of operations, and improving the Von-Neumann bottleneck problem [9].

In addition to changing technology, higher efficiency can be achieved by changing how computations are done. For example, hardware complexity, energy consumption, and delay can be reduced by using approximate computing [13]. Approximate computing is a design method that can be applied in error-tolerant applications. The benefits of approximate computing are obtained at the cost of decreasing the accuracy of the computing. Therefore, approximate computing cannot be used in applications that require high accuracy. Image processing, signal processing, and pattern recognition are among the applications in which it is possible to use approximate computing [13]. To design approximate circuits, some elements of exact circuits can be removed or modified, or the truth table's values can be changed to have less hardware complexity and lead to an efficient design [6,13]. Approximate computing has been applied to design accelerators and low-power processors by IBM and ARM [14].

Along with circuit evaluation criteria in approximate computing, error evaluation measures such as Error Rate (ER), Error Distance (ED), Mean Error Distance (MED), and Normalized MED (NMED) are evaluated [13]. ED is the arithmetic value of subtraction of exact and inexact values [6]. The ER is the ratio of the number of wrong outputs to all possible output combinations [6]. MED is the mean ED of all inputs, and NMED is the normalized MED of all possible inputs [13].

Basic circuits such as adder and multiplier are the most critical parts of a processor. Due to the widespread use of these circuits in various computing applications and the increase in the amount of data required for processing, these circuits impact the efficiency of computing systems. Using memristors to implement approximate basic circuits for IMC has many advantages. In this paper, the authors have introduced an efficient approximate full adder for In-Memory Processing (IMP) of data-intensive error-tolerant applications. The IMPLY design method has been used in designing the full adder circuit due to its compatibility with the structure of crossbar arrays and IMC. Our contributions to this paper are as follows:

1. Proposing a Serial Approximate Full Adder using NAND gates (SAFAN) for IMC by applying the IMPLY method,
2. Evaluation of the applicability of the SAFAN in image processing applications,
3. Comparison of the SAFAN with exact and approximate IMPLY-based circuits,
4. Evaluate the SAFAN by Figure of Merit (FoM) to represent the design goals.

The rest of this paper is organized as follows. Section 2 gives a background of memristors, design methods, and memristor-based adder circuits. In Section 3, the implementation algorithm of the SAFAN is explained. Section 4 presents simulations and evaluations of the SAFAN, and finally, Section 5 concludes the paper.

2. Literature review

2.1. Memristors

In 1971, Leon Chua named the electrical element, defined based on the relationship between flux and electric charge, memristor [16]. The memristor is the fourth passive electrical element, an abbreviation of the memory resistor. In 2008, the first

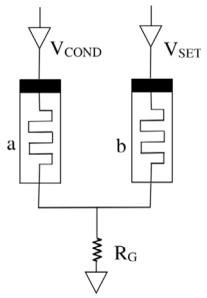

Fig. 2. IMPLY logic gate [13].

Table 1
IMPLY logic truth table [20].

Case	a	b	$a \rightarrow b$
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1

memristor was made in HP laboratories, and since then, lots of research has been done to build, design, model, and synthesize circuits based on memristors [17]. The first memristor was made by placing a layer of titanium dioxide (which includes two undoped and doped regions with positive oxygen ions (TiO_{2-x})) between two upper and lower platinum electrodes [15]. Fig. 1 demonstrates the memristor symbol [15]. Memristors can play both the role of memory and computing units. Memristors enable IMC due to their compatibility with crossbar array structure and can avoid massive data movement [11]. The resistance value of a memristor can change between two states, R_{ON} (logic '1') or R_{OFF} (logic '0') [2]. The memristor's resistance diminishes by applying the current from the direction of positive polarity (See Fig. 1). By applying the current from the opposite direction, negative polarity, the resistance increases. The resistance change of the memristor can also be assessed according to the applied voltage. If the input voltage of the memristor is higher than V_{SET}, the memristor will switch to the R_{ON} state. On the other hand, if the input voltage of the memristor is smaller than the V_{RESET}, the memristor switches to the R_{OFF} state [18].

2.2. Memristor-based design methods

Memristor logic families can be divided into stateful and non-stateful logic [19]. In the stateful logic, inputs, intermediate nodes, and outputs are measured as resistance. IMPLY [20], Memristor-Aided loGIC (MAGIC) [12], and reconfigurable NAND/AND gate [21] design methods are examples of this logic family. In the non-stateful logic family, input and output logics are determined by their voltage value [22]. The Memristor Ratioed Logic (MRL) design method [23] is included in this logic family. One of the criteria that affects the choice of circuit design method is the degree of compatibility of that method with the structure of crossbar arrays. The structure of crossbar arrays consists of two sets of nanowires that are perpendicular to each other. A memristor can be placed at each intersection of two wires [22]. Stateful design methods are well adapted to the structure of crossbar arrays. The SAFAN is designed and implemented based on the IMPLY design method, a stateful logic. This design method will be discussed in detail.

Only two operations, IMPLY and FALSE, are applied to implement various functions and circuits using IMPLY. Combining these two operations creates a complete logical set that can be used to design any function [13]. IMPLY operation is shown as $a \rightarrow b$, and its equivalent function is $b = \overline{a} + b$. As can be seen, the final value is stored in memristor b, and this causes memristor b to lose its initial value. The IMPLY logic gate, $a \rightarrow b$, is shown in Fig. 2, and its truth table is written in Table 1 [20]. Based on Table 1, the output value is '1' in all input cases, except for case 3, when memristor a is in low resistance state (logic '1' or R_{ON}) and memristor b is in high resistance state (logic '0' or R_{OFF}).

The IMPLY logic gate shown in Fig. 2 includes two memristors, a and b, and a resistor R_G. The two memristors' initial resistance value shows the input logic. The final result after applying two voltages of V_{SET} and V_{COND} to memristors b and a, respectively, is stored in memristor b. The following conditions must be considered to build the IMPLY gate:

- The resistance value of R_G must be between R_{ON} and R_{OFF} ($R_{ON} < R_G < R_{OFF}$)
- V_{SET} should be considered higher than V_{COND} ($|V_{COND}| < |V_{SET}|$) [20,24].

The following equations show the basic logic gates implementations based on IMPLY and FALSE functions [18,25].

$$NOT\ A_{in} : A_{in} \rightarrow 0 \equiv A_{in} \rightarrow FALSE \tag{1}$$

$$A_{in}\ OR\ B_{in} : (B_{in} \rightarrow 0) \rightarrow A_{in} \equiv \overline{A_{in}} \rightarrow B_{in} \equiv \overline{B_{in}} \rightarrow A_{in} \tag{2}$$

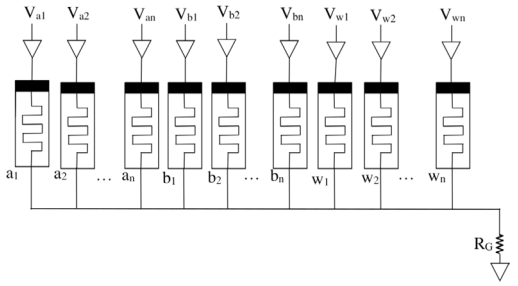

Fig. 3. Design of an adder based on IMPLY in serial architecture [9].

$$A_{in}\ NOR\ B_{in} : ((B_{in} \rightarrow 0) \rightarrow A_{in}) \rightarrow 0 \equiv \overline{\overline{A_{in}} \rightarrow B_{in}} \equiv \overline{\overline{B_{in}} \rightarrow A_{in}} \tag{3}$$

$$A_{in}\ AND\ B_{in} : (B_{in} \rightarrow (A_{in} \rightarrow 0)) \rightarrow 0 \equiv \overline{A_{in} \rightarrow \overline{B_{in}}} \equiv \overline{B_{in} \rightarrow \overline{A_{in}}} \tag{4}$$

$$A_{in}\ NAND\ B_{in} : (B_{in} \rightarrow (A_{in} \rightarrow 0)) \equiv A_{in} \rightarrow \overline{B_{in}} \equiv B_{in} \rightarrow \overline{A_{in}} \tag{5}$$

$$A_{in}\ XOR\ B_{in} : (A_{in} \rightarrow B_{in}) \rightarrow \overline{B_{in} \rightarrow A_{in}} \equiv (A_{in} \rightarrow B_{in}) \rightarrow \overline{\overline{A_{in}} \rightarrow \overline{B_{in}}} \equiv \overline{(\overline{A_{in}} \rightarrow B_{in}) \rightarrow (A_{in} \rightarrow \overline{B_{in}})} \tag{6}$$

$$A_{in}\ XNOR\ B_{in} : (A_{in} \rightarrow \overline{B_{in}}) \rightarrow \overline{\overline{B_{in}} \rightarrow A_{in}} \tag{7}$$

2.3. Memristive adders based on IMPLY logic

Memristive adders based on IMPLY Logic are designed in serial, parallel, semi-serial, and semi-parallel architectures [13]. The serial adder has advantages such as the need for fewer memristors than other architectures, minor hardware complexity, and no need for an external switch. However, it is the slowest architecture compared to other architectures [13]. A memristive serial adder is represented in Fig. 3 [9]. Adders implemented in parallel architecture are the fastest but occupy the largest array area regarding the number of memristors and use many switches [9]. Semi-serial and semi-parallel designs attain a trade-off between the number of memristors and the number of steps. These architectures require CMOS switches to connect different circuit parts [9,24]. From one approach, memristive arithmetic circuits are divided into exact and approximate circuits. Therefore, this section examines the adders designed by the IMPLY method in two groups.

2.3.1. Exact adders

In [25], the authors have designed a memristor-based serial full adder with the IMPLY method. The proposed algorithm in [25] requires only two work memristors in addition to three input memristors, and its execution time is 22 computational steps. The proposed algorithm stores the results of Sum and C_{out} on the input memristors, reducing the number of memristors. It should be noted that the initialization of work memristors is considered in the computational steps of this algorithm.

In [18], an IMPLY-based full adder applying serial architecture that requires five memristors is designed. This full adder's implementation algorithm takes 21 computational steps. Considering the initialization of work memristors, the number of computational steps increases to 23. The authors also presented two IMPLY-based serial and serial-parallel 8-bit adder architectures based on their proposed full adder [18].

A novel full adder based on semi-parallel architecture has been presented by applying the IMPLY design method [24]. The semi-parallel full adder proposed in [24] computes the outputs with fewer steps and more switches than IMPLY-based serial designs. This architecture uses fewer memristors and switches than IMPLY-based parallel designs. The proposed circuit in [24] requires five memristors and takes 17 computational steps considering the initialization of work memristors.

In [9], the authors first introduced a full adder in semi-serial architecture by applying the IMPLY method and then, in the next step, presented a multiplier using their proposed full adder. The semi-serial architecture proposed in [9] combines the features of serial and parallel architectures, intending to attain a design with better performance and uses fewer switches than the parallel design. The semi-serial full adder proposed in [9] requires eight memristors (five work memristors and three input memristors), and its implementation algorithm takes 12 steps.

Carry select and conditional carry adders have been proposed by applying the IMPLY method in [4]. The proposed multiplexers in this paper and the parallel full adder proposed in [18] are applied to implement the carry select adder. The n-bit carry select adder proposed in [4] requires $19n/2 + 6$ memristors, and its execution time is $3n + 27$ computational steps. The conditional carry adder proposed in [4] uses modified half adders and proposed multiplexers. The proposed 32-bit conditional carry adder requires 758 memristors, and its execution time is 90 computational steps. It should be noted that the authors in [4] did not consider the steps related to the initialization of work memristors in their designs.

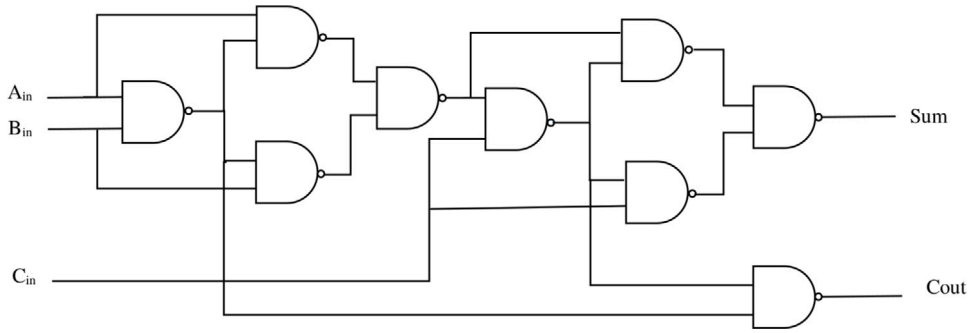

Fig. 4. Exact full adder designed with NAND gates [27].

2.3.2. Approximate adders

In [26], the authors implemented an approximate full adder using five memristors (two work memristors and three input memristors) and 8 computational steps. The ERs of approximate *Sum* and C_{out} are $\frac{3}{8}$ and $\frac{1}{8}$, respectively. The main drawback of this approximate full adder is that both *Sum* and C_{out} outputs are stored in work memristors. Thus, the number of memristors required in the n-bit adder increases by storing the output values in the work memristors. The approximate full adder implementation algorithm introduced in [26] was redesigned in [13] to store the *Sum* and C_{out} outputs in input memristors. The redesigned approximate full adder is called Serial IMPLY-based Approximate Full Adder 1 (SIAFA1) [13]. In [13], the authors named their four proposed approximate full adders SIAFA1 to SIAFA4. Designing these approximate circuits has been done using truth table modification method by specifying the input vectors for the proposed circuits. The criteria of the computational steps, the number of memristors, and the energy consumption were investigated as circuit evaluation criteria in [13]. The SIAFAs were evaluated in different schemes in Ripple Carry Adders (RCAs). The performance of the proposed approximate full adders was evaluated in four different image processing applications. SIAFA1 to SIAFA4 takes 8, 10, 8, and 8 steps, and the number of memristors is 4, 5, 4, and 4, respectively [13]. The ER of the *Sum* outputs of SIAFA1 to SIAFA4 is $\frac{3}{8}$, $\frac{2}{8}$, $\frac{3}{8}$, and $\frac{3}{8}$, respectively. The C_{out}'s ER in all proposed approximate full adders equals $\frac{1}{8}$ [13].

3. Proposed IMPLY-based approximate full adder

The NAND gate is functionally complete and is known as a universal gate. So, any function can be implemented with multiple NAND gates. The logic functions of *Sum* and C_{out} outputs of exact full adder are obtained according to its truth table. These functions are written in Eqs. (8) and (9) [27]. According to these equations and the implementation of the XOR gate using NAND gates, the exact full adder has been implemented using NAND gates [27]. This circuit is shown in Fig. 4 [27].

$$Sum = \overline{A_{in}} \cdot \overline{B_{in}} \cdot C_{in} + A_{in} \cdot \overline{B_{in}} \cdot \overline{C_{in}} + \overline{A_{in}} \cdot B_{in} \cdot \overline{C_{in}} + A_{in} \cdot B_{in} \cdot C_{in} = A_{in} \oplus B_{in} \oplus C_{in} \tag{8}$$

$$C_{out} = A_{in} \cdot B_{in} + A_{in} \cdot C_{in} + B_{in} \cdot C_{in} = A_{in} \cdot B_{in} + C_{in} \cdot (A_{in} \oplus B_{in}) \tag{9}$$

The SAFAN is designed by modifying the elements of the exact NAND-based full adder, changing the exact truth table, and determining the input vectors. In Fig. 5(a), the gate-level design of SAFAN is drawn to show the logic function of the outputs. The authors implement the SAFAN with only NAND gates. The circuit diagram and truth table of the SAFAN are shown in Fig. 5 and Table 2, respectively. Exact states are marked with ✓, and inexact states are marked with ×. The SAFAN's *Sum* and C_{out} ERs are $\frac{3}{8}$ and $\frac{1}{8}$, respectively. The SAFAN is implemented using the serial IMPLY design method. The vectors p_1, q_1, and p_2, q_2 acquire the *Sum* and C_{out} outputs of the SAFAN, respectively. These vectors are obtained by considering the truth table of the SAFAN and using the method introduced in [25]. The vectors p and q are selected according to the truth table of the IMPLY logic gate so that $p \rightarrow q$ reaches the desired logic. Remember that if p = '1' and q = '0', $p \rightarrow q$ is equal to '0' and otherwise equal to '1'. First, to design the *Sum* output, vectors p_1 and q_1 are considered equal to [11111110] and [10101010], respectively. The equation $p_1 \rightarrow q_1$ is written in Eq. (10). The seventh bit of the vector p_1 is changed to '0' to reduce the complexity of Eq. (10), considering that the result of '0' → '1' and '1' → '1' equals '1'. The vector q_1 remains unchanged. Therefore, modified p_1 (p_1') becomes equal to [11111100]. The rewritten equation of the $p_1' \rightarrow q_1$ is written in Eq. (11).

$$Sum = p_1 \rightarrow q_1 \equiv \overline{\overline{A_{in} \cdot B_{in}} \cdot C_{in}} \equiv [C_{in} \rightarrow (B_{in} \rightarrow \overline{A_{in}})] \rightarrow \overline{C_{in}} \tag{10}$$

$$Sum = p_1' \rightarrow q_1 \equiv (A_{in} \rightarrow \overline{B_{in}}) \rightarrow \overline{C_{in}} \tag{11}$$

The same method as the *Sum* output is applied to design C_{out} by computing $p_2 \rightarrow q_2$. The C_{out} ($p_2 \rightarrow q_2$) vector of the SAFAN is implemented by considering p_2 = [111X1X0X] and q_2 = [01010101]. In p_2, the do not care states that can be '0' or '1' are marked

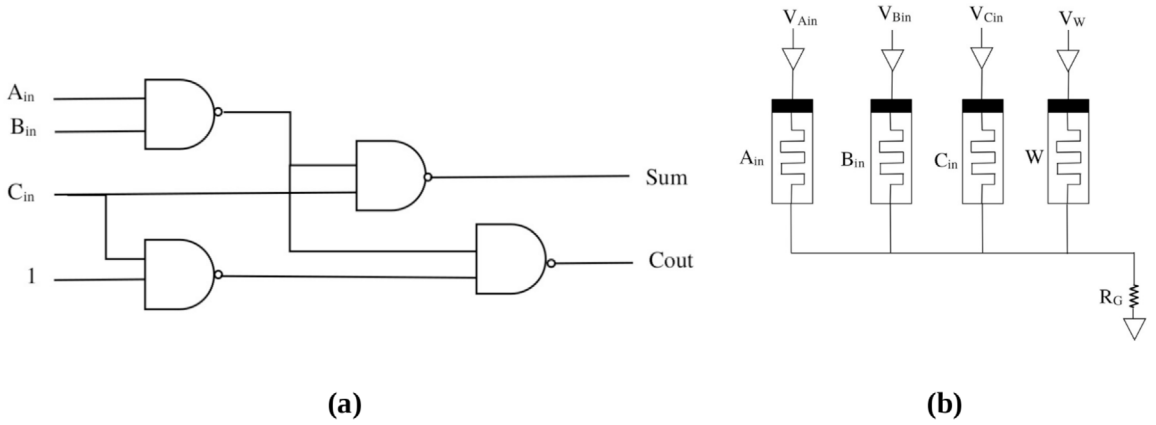

Fig. 5. The SAFAN: (a) gate-level design, and (b) the IMPLY-based circuit.

Table 2
The truth table of the SAFAN.

A_{in}	B_{in}	C_{in}	Exact Sum	Exact C_{out}	Proposed Sum	Proposed C_{out}
0	0	0	0	0	1 ×	0 ✓
0	0	1	1	0	0 ×	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	1 ×	1 ✓
1	1	1	1	1	1 ✓	1 ✓

Table 3
Computational steps of the SAFAN.

Step	Operation	Equivalent logic	Equivalent gate
1	$W = 0$	$FALSE(W)$	$RESET(W)$
2	$B_{in} \rightarrow W = W'$	$NOT(B_{in})$	$W = \overline{B_{in}}$
3	$A_{in} \rightarrow W' = W''$	$A_{in} \rightarrow \overline{B_{in}}$	$W = \overline{A_{in}} + \overline{B_{in}} = \overline{A_{in} \cdot B_{in}} = NAND(A_{in}, B_{in})$
4	$B_{in} = 0$	$FALSE(B_{in})$	$RESET(B_{in})$
5	$C_{in} \rightarrow B_{in} = B'_{in}$	$NOT(C_{in})$	$B_{in} = \overline{C_{in}}$
6	$W'' \rightarrow B'_{in} = B''_{in}$	$Sum = (A_{in} \rightarrow \overline{B_{in}}) \rightarrow \overline{C_{in}}$	$Sum = \overline{\overline{A_{in} \cdot B_{in}} \cdot C_{in}} = NAND(NAND(A_{in}, B_{in}), C_{in})$
7	$W'' \rightarrow C_{in} = C'_{in}$	$C_{out} = (A_{in} \rightarrow \overline{B_{in}}) \rightarrow C_{in}$	$C_{out} = \overline{\overline{A_{in} \cdot B_{in}} \cdot \overline{C_{in}}} = NAND(NAND(A_{in}, B_{in}), \overline{C_{in}})$

with 'X'. Overlapping the computational steps is possible by setting p_2 and p'_1 equal. The logic function of C_{out} that is equal to $p_2 \rightarrow q_2$ is written in Eq. (12).

$$C_{out} = p_2 \rightarrow q_2 \equiv \overline{\overline{A_{in} \cdot B_{in}} \cdot \overline{C_{in}}} \equiv (A_{in} \rightarrow \overline{B_{in}}) \rightarrow C_{in} \tag{12}$$

The SAFAN requires four memristors (three input memristors, A_{in}, B_{in}, and C_{in}, and one work memristor, W). Fig. 5(b) is drawn to understand the synchronization problem. The proposed approximate full adder can be implemented in a row or column of a memristive crossbar array for IMP. In this architecture, an IMPLY operation (between two memristors) or FALSE (one memristor) is performed in each computational cycle, and the data of other memristors does not change. Hence, the resistance of the memristors does not change if they are not the operands of IMPLY or FALSE operations. The SAFAN reuses the input memristors to store the results of the outputs, thus reducing the number of required memristors. The Sum result is stored in the B_{in} memristor, and the C_{out} result is stored in the C_{in} memristor. The algorithm of the SAFAN takes seven computational steps. By not considering the initialization of the work memristor, the step count is reduced to six. The computational steps for the SAFAN are shown in Table 3.

In the 1st step, the only work memristor (W) is reset. In the 2nd step, the value of the B_{in} memristor is inverted and stored in the W memristor. In the 3rd step, the value of $\overline{A_{in} \cdot B_{in}}$ is stored in the memristor W. In the 4th step, the B_{in} memristor is reset. In the 5th step, the inverse of the C_{in} memristor value is stored in the B_{in} memristor. In the 6th step, the value of Sum equals $\overline{\overline{A_{in} \cdot B_{in}} \cdot C_{in}}$ is obtained. Finally, in the 7th step, the value of C_{out}, which equals $\overline{\overline{A_{in} \cdot B_{in}} \cdot \overline{C_{in}}}$, is computed. The equivalent logic gate of each step is added to the rightmost column of Table 3.

Table 4
Setup parameters of the VTEAM model and IMPLY logic [13].

Parameter	Value	Parameter	Value	Parameter	Value
v_{off}	0.7 V	v_{on}	−10 mV	α_{off}	3
α_{on}	3	R_{off}	1 MΩ	R_{on}	10 kΩ
k_{on}	$-0.5\ \frac{nm}{s}$	k_{off}	$1\ \frac{cm}{s}$	w_{off}	0 nm
w_{on}	3 nm	w_C	107 pm	a_{off}	3 nm
a_{on}	0 nm	v_{set}	1 V	v_{reset}	1 V
v_{cond}	900 mV	R_G	40 KΩ	t_{pulse}	30 µs

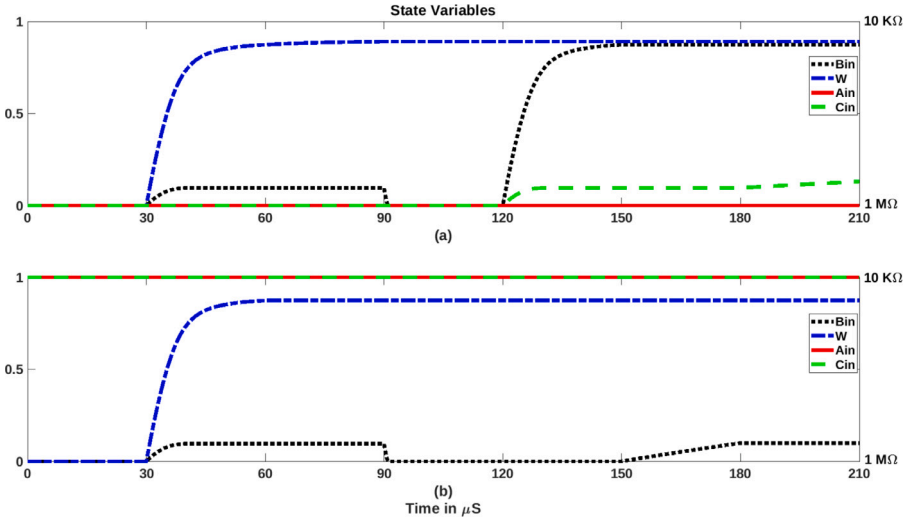

Fig. 6. The output waveform of the SAFAN in the states: (a) $A_{in}B_{in}C_{in}$ = "000", and (b) $A_{in}B_{in}C_{in}$ = "101".

4. Results of simulation and comparison

4.1. Circuit-level simulation and analysis

The SAFAN is simulated in the LTSPICE simulator utilizing the VTEAM model to ensure the correct process of the SAFAN. The parameters applied in the VTEAM model and IMPLY logic are written in Table 4 [13]. The correctness of the SAFAN was checked in all states of its truth table. For example, the memristors' normalized resistance value waveform of the SAFAN in the states $A_{in}B_{in}C_{in}$ = "000" and $A_{in}B_{in}C_{in}$ = "101" is drawn in Fig. 6(a) and (b). In Fig. 6, every 30 µs shows a step of the proposed algorithm. In Fig. 6(a), the value of all inputs is equal to zero ($A_{in}B_{in}C_{in}$ = "000"). During the 1st step (0 µs–30 µs), the work memristor is reset (W = '0'). During the 2nd step (30 µs–60 µs), the value of the B_{in} memristor is inverted and stored in the W memristor. Therefore, the memristor W has a value of '1' at the end of 60 µs. In the 3rd step (60 µs–90 µs), the value of $\overline{A_{in} \cdot B_{in}}$ is stored in the memristor W. So, at the end of 90 µs, the memristor W still has a value of '1'. The B_{in} memristor is reset during the 4th step (90 µs–120 µs). Therefore, the B_{in} memristor still equals '0'. During the 5th step (120 µs–150 µs), the C_{in} memristor value is inverted, and the result is stored in the B_{in} memristor. Therefore, the B_{in} memristor has a value of '1' at the end of this computational step. During the 6th step (150 µs–180 µs), the value of Sum, which equals '0', is stored in the B_{in} memristor. Finally, in the last step (180 µs–210 µs), the value of C_{out}, equals '0', is stored in the memristor C_{in}. The same explanation given in Fig. 6(a) holds for Fig. 6(b) except for their inputs.

The SAFAN is compared with previous works [13,18,25] regarding energy consumption, number of memristors, and steps in Table 5. In this paper, like [13], to calculate the full adder's energy consumption, first, energy consumption of a memristor's in each state of the truth table of a full adder block is computed. Then, the total energy consumption of the memristors involved in the implementation algorithm is calculated, and the average energy consumption is reported as an estimation of the energy consumption. The energy consumption of the SAFAN is reduced by up to 26% compared to SIAFA1-4 and up to 68% compared to the proposed circuits in [18,25]. The number of memristors to implement the SAFAN is equal to those required to implement SIAFA1, SIAFA3, and SIAFA4. According to the algorithm in Table 3, the SAFAN's computational steps have been reduced by at least 12.5% and at most 30% compared to the proposed circuits in [13] and at most 70% compared to the proposed circuits in [18,25].

Two schemes are considered to evaluate the SAFAN in the 8-bit approximate RCAs. In scheme 1, the SIAFA1-4 [13] and SAFAN are used for the 3 LSBs, and the exact full adders are used for the 5 Most Significant Bits (MSBs). In scheme 2, the 4 LSBs are implemented with the SIAFA1-4 [13] and SAFAN, and the 4 MSBs are implemented with the exact full adders. The computational structure drawn in Fig. 7 refers to the scheme 2's architecture. The exact full adder introduced in [25] has been used for circuit

Table 5
Comparison of the number of steps, energy consumption, and the number of memristors of exact and approximate full adders [13,18,25] in 2 schemes.

Serial full adder (exact and approximate)	Steps	Improvement percentage over [18]	Memristor count	Improvement percentage over [18]	Energy Consumption ($\times 10^{-9}$ J)	Improvement percentage over [18]
[25]	22	5%	5	0%	1.90859	5%
[18]	23	–	5	–	2.00727	–
SIAFA1 [13]	8	65%	4	20%	0.67221	67%
SIAFA2 [13]	10	57%	5	0%	0.86032	57%
SIAFA3 [13]	8	65%	4	20%	0.67221	67%
SIAFA4 [13]	8	65%	4	20%	0.67086	67%
SAFAN	7	70%	4	20%	0.64282	68%

Table 6
Comparison of step count, energy dissipation, and memristor count of exact and approximate full adders [13,18,25].

Serial adder	No. of steps		No. of memristors		Energy consumption ($\times 10^{-9}$ J)	
(exact and approximate)	n	n = 8-bit	n	n = 8-bit	n	n = 8-bit
[25]	22n	176	2n + 3	19	1.90859n	15.26872
[18]	23n	184	2n + 3	19	2.00727n	16.05816
Scheme 1						
SIAFA1 [13]	8(n−5) + 22(n−3)	134	2n + 3	19	0.67221(n−5) + 1.90859(n−3)	11.55958
SIAFA2 [13]	10(n−5) + 22(n−3)	140	2n + 3	19	0.86032(n−5) + 1.90859(n−3)	12.12391
SIAFA3 [13]	8(n−5) + 22(n−3)	134	2n + 3	19	0.67221(n−5) + 1.90859(n−3)	11.55958
SIAFA4 [13]	8(n−5) + 22(n−3)	134	2n + 3	19	0.67086(n−5) + 1.90859(n−3)	11.55553
SAFAN	7(n−5) + 22(n−3)	131	2n + 3	19	0.64282(n−5) + 1.90859(n−3)	15.26872
Scheme 2						
SIAFA1 [13]	8(n−4) + 22(n−4)	120	2n + 3	19	0.67221(n−4) + 1.90859(n−4)	10.3232
SIAFA2 [13]	10(n−4) + 22(n−4)	128	2n + 3	19	0.86032(n−4) + 1.90859(n−4)	11.07564
SIAFA3 [13]	8(n−4) + 22(n−4)	120	2n + 3	19	0.67221(n−4) + 1.90859(n−4)	10.3232
SIAFA4 [13]	8(n−4) + 22(n−4)	120	2n + 3	19	0.67086(n−4) + 1.90859(n−4)	10.3178
SAFAN	7(n−4) + 22(n−4)	116	2n + 3	19	0.64282(n−4) + 1.90859(n−4)	10.20564

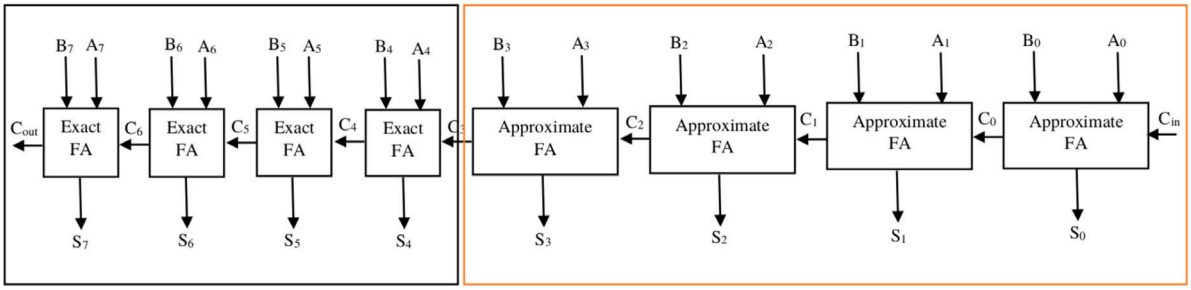

Fig. 7. Computational structure of the scheme 2.

simulations of these two schemes in MSBs. In Table 6, the number of memristors, the number of computational steps, and the energy dissipation of the 8-bit approximate adders, including the approximate full adders proposed in this paper, in [13], as well as the exact full adders proposed in [18,25], are written. The method applied to compute the number of computational steps and the energy consumption of the n-bit approximate adder that LSBs are computed approximately in schemes 1 and 2 is written in columns 2 and 6 of Table 6, respectively. The step count of the SAFAN is decreased by up to 37% compared to exact full adders and up to 10% compared to approximate full adders. All approximate 8-bit structures in Table 6 require 19 memristors.

4.2. Error analysis simulation and comparison

Error analysis criteria such as MED and NMED should be computed and checked in addition to circuit evaluation criteria in approximate computing. The reason for the importance of checking the error evaluation criteria is to measure the reduction of computation accuracy along with hardware complexity. Accordingly, the SAFAN and SIAFA1-4 [13] were placed in the 3 and 4 LSBs of the 8-bit approximate adder. All 65536 possible inputs' states were applied to these approximate adders. The simulation results of error analysis are reported in Table 7. The computation's accuracy decreases to a reasonable level by applying the SAFAN to the 3 and 4 LSBs of the 8-bit approximate adder by examining the results of Table 7.

Table 7
Simulation results of the error analysis criteria.

Approximate adder	MED	NMED
Scheme 1		
SIAFA1 [13]	2.062	0.004
SIAFA2 [13]	2.656	0.0052
SIAFA3 [13]	2.062	0.004
SIAFA4 [13]	2.625	0.0051
SAFAN	2.9375	0.0057
Scheme 2		
SIAFA1 [13]	4.351	0.0085
SIAFA2 [13]	6.1718	0.0121
SIAFA3 [13]	4.351	0.0085
SIAFA4 [13]	5.3125	0.0104
SAFAN	5.78125	0.0113

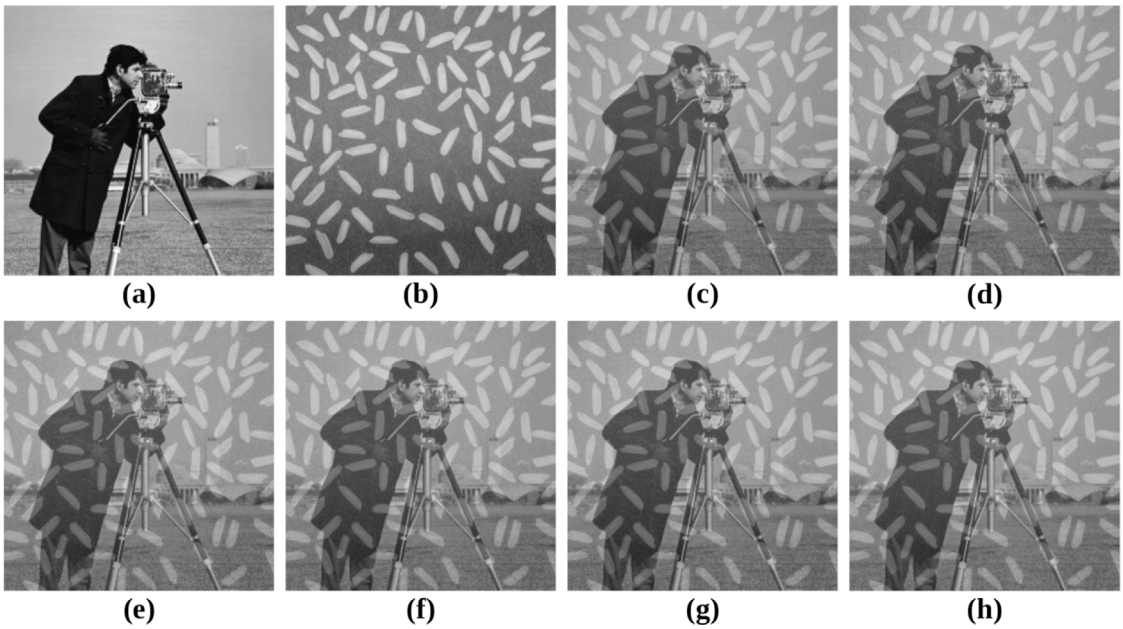

Fig. 8. The results of image addition in scheme 2. (a) Cameraman, (b) Rice, (c) Exact result, (d) SIAFA1 [13], (e) SIAFA2 [13], (f) SIAFA3 [13], (g) SIAFA4 [13], and (h) SAFAN.

4.3. Application-level simulation and comparison

The SAFAN was evaluated in error-tolerant image processing applications, i.e., image addition, image subtraction (motion detection), and grayscale filter. Ensuring the proper usefulness of the SAFAN in error-tolerant applications is of great importance in approximate computing. This proposed approximate full adder and applied RCAs in schemes 1 and 2 in their behavioral context are implemented in MATLAB.

In the image addition application simulation, two 8-bit 256 × 256 pixels grayscale images are added together, and the result is stored in the output image's corresponding pixel of the two input pixels. In image subtraction, the subtraction result of two image sequences showing motion is written in the output image's corresponding pixel of the two subtracted pixels. Two 512 × 512 pixels grayscale images [28] have been used to simulate grayscale filter application. The grayscale filter converts an 8-bit RGB image into a grayscale photo. For this purpose, the Red, Green, and Blue values of pixels are added together. Then, the final sum is divided by three and written in the output image's corresponding pixel of the input pixel [13]. It should be noted that the exact divider is used in this application.

The simulation results of image addition application, image subtraction, and grayscale filter are written in Tables 8–10, respectively. The quality of the output images is evaluated by Peak Signal to Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Mean SSIM (MSSIM) criteria. Calculating these three criteria is explained in detail in [5,14,29]. The scheme 2's simulation results of image addition, image subtraction, and grayscale filter are shown in Figs. 8–10, respectively.

When the SAFAN is placed in 3 and 4 LSBs of the 8-bit approximate adder, the PSNR was greater than 30 dB in all cases. The quality of images with PSNR greater than 30 dB is acceptable [30]. The results illustrated that applying the SAFAN in the

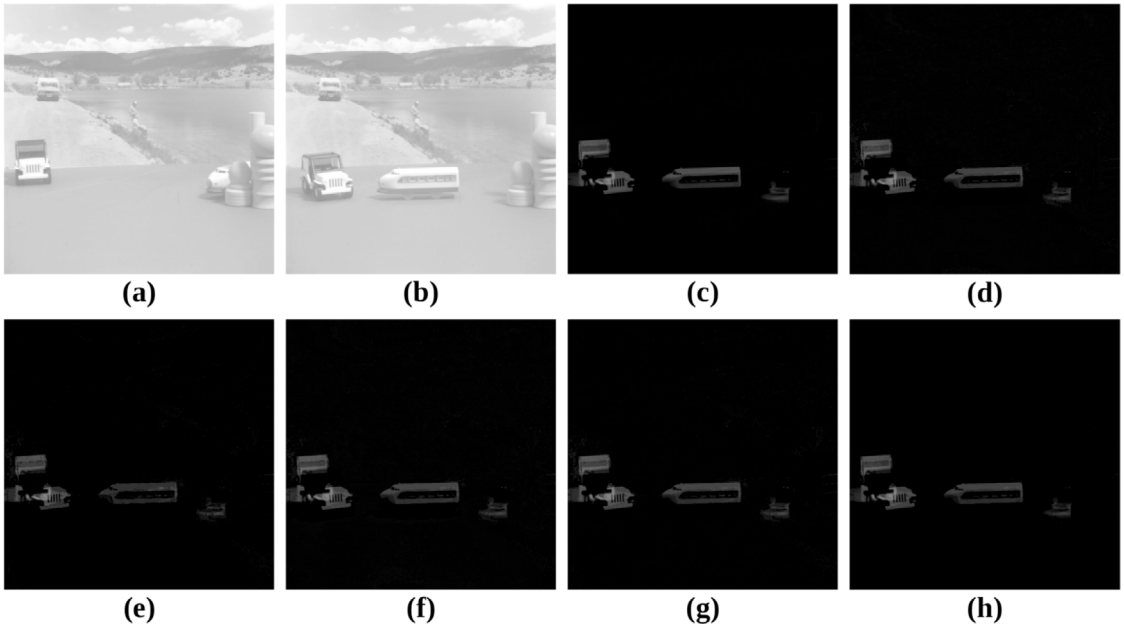

Fig. 9. The results of image subtraction in scheme 2. (a) Image 1 [28], (b) Image 2 [28], (c) Exact result, (d) SIAFA1 [13], (e) SIAFA2 [13], (f) SIAFA3 [13], (g) SIAFA4 [13], and (h) SAFAN.

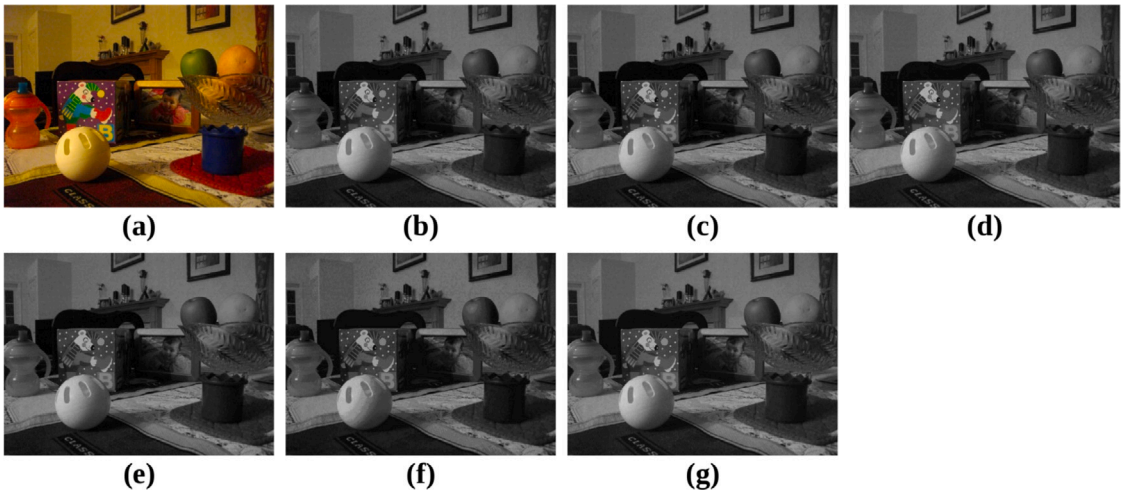

Fig. 10. The results of the grayscale filter in scheme 2. (a) RGB image, (b) Exact result, (c) SIAFA1 [13], (d) SIAFA2 [13], (e) SIAFA3 [13], (f) SIAFA4 [13], and (g) SAFAN.

approximate structures of scheme 1 and scheme 2 is accompanied by an acceptable reduction in computational accuracy, hardware complexity, and an increase in computational efficiency.

Each designer can design a FoM according to their limitations to better represent their goals. In [13], a FoM is introduced based on Energy Delay Product (EDP) and NMED. The purpose of using this FoM was to create a compromise between the circuit-level evaluation criteria and the accuracy evaluation criteria. Any design with a smaller FoM is better regarding the desired criteria. The FoM's equation is written in Eq. (13). FoM results are written in Table 11. Based on the FoM, when the SAFAN is placed in the approximate computational structure of scheme 2, it provides the best compromise between the circuit-level evaluation criteria and the accuracy.

$$FoM = \frac{EDP}{(1 - NMED)} \tag{13}$$

Table 8
Image quality criteria in the application of image addition.

Approximate adder	PSNR (dB)	SSIM	MSSIM
Scheme 1			
SIAFA1 [13]	44.5148	0.9899	0.99
SIAFA2 [13]	41.9674	0.9858	0.9861
SIAFA3 [13]	44.5222	0.9898	0.99
SIAFA4 [13]	43.7483	0.9878	0.988
SAFAN	41.8917	0.994	0.994
Scheme 2			
SIAFA1 [13]	38.67	0.9644	0.9649
SIAFA2 [13]	35.4576	0.9425	0.9436
SIAFA3 [13]	38.8399	0.9638	0.9644
SIAFA4 [13]	37.8083	0.959	0.9597
SAFAN	36.6395	0.9793	0.9796

Table 9
Image quality criteria in the application of image subtraction.

Approximate adder	PSNR (dB)	SSIM	MSSIM
Scheme 1			
SIAFA1 [13]	41.5919	0.7711	0.7901
SIAFA2 [13]	43.5407	0.9678	0.9874
SIAFA3 [13]	41.8705	0.7905	0.8103
SIAFA4 [13]	45.5121	0.9534	0.9693
SAFAN	49.7504	0.9863	0.9905
Scheme 2			
SIAFA1 [13]	37.4131	0.6405	0.6705
SIAFA2 [13]	37.5605	0.9338	0.9652
SIAFA3 [13]	36.8613	0.5993	0.6302
SIAFA4 [13]	40.3861	0.9131	0.9423
SAFAN	44.0205	0.9798	0.9857

Table 10
Image quality criteria in grayscale filter application.

Approximate adder	PSNR (dB)	SSIM	MSSIM
Scheme 1			
SIAFA1 [13]	47.1982	0.9911	0.999
SIAFA2 [13]	43.1339	0.9833	0.9977
SIAFA3 [13]	47.2496	0.9914	0.999
SIAFA4 [13]	43.0565	0.9841	0.997
SAFAN	42.2449	0.9905	0.9974
Scheme 2			
SIAFA1 [13]	41.4201	0.9693	0.9957
SIAFA2 [13]	35.9998	0.9263	0.9874
SIAFA3 [13]	41.2315	0.9684	0.9956
SIAFA4 [13]	36.9634	0.9451	0.989
SAFAN	36.0101	0.9625	0.9886

Table 11
FoM results of SIAFA1-4 [13] and SAFAN in the scheme 2 of the RCA structure.

Approximate adder	FoM
SIAFA1 [13]	1249.4434
SIAFA2 [13]	1435.0482
SIAFA3 [13]	1249.4434
SIAFA4 [13]	1251.169
SAFAN	1197.4279

5. Conclusion

This paper presents an inexact memristive serial full adder based on the IMPLY design method. Due to the advantages of approximate computing, the computational steps required to compute outputs and the total energy consumption of the SAFAN has been reduced by a maximum of 70% and 68% compared to the exact cells, respectively. The SAFAN has also improved the number of steps and energy dissipation by up to 30% and 26% compared to the existing inexact counterparts, respectively. Error analysis and image quality criteria, including MED, NMED, PSNR, SSIM, and MSSIM, are considered in two different schemes for evaluation. In the scheme 1, 3 LSBs, and in the scheme 2, 4 LSBs of an 8-bit RCA are implemented with the SAFAN. The evaluation results show that in both schemes, the adders implemented with the SAFAN have the lowest step count compared to the previous works. The error analysis criteria show that the accuracy is reduced to a reasonable level by applying the SAFAN to the structures of scheme 1 and scheme 2. Hence, the correct operation of the SAFAN has been confirmed. Also, in this paper, an existing FoM is used, which measures the product of energy dissipation, the step count, and NMED simultaneously. The SAFAN has obtained the best FoM compared to the previous works.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Mohammad Reza Reshadinezhad reports a relationship with University of Isfahan that includes: employment. No Conflict of Interest.

Data availability

No data was used for the research described in the article.

References

[1] Haj-Ali A, Ronen R, Ben-Hur R, Wald N, Kvatinsky S. Memristor-based in-memory logic and its application in image processing. In: Memristive devices for brain-inspired computing. Elsevier; 2020, p. 175–94.
[2] Eliahu A, Ben-Hur R, Haj-Ali A, Kvatinsky S, Andrade L, Rousseau F. mMPU: Building a memristor-based general-purpose in-memory computation architecture. Multi-Pr Syst-on-Chip 1 Archit 2021;6:119–32.
[3] Dlugosch P, Brown D, Glendenning P, Leventhal M, Noyes H. An efficient and scalable semiconductor architecture for parallel automata processing. IEEE Trans Parallel Distrib Syst 2014;25(12):3088–98.
[4] Kaushik N, Srinivasu B. IMPLY-based high-speed conditional carry and carry select adders for in-memory computing. IEEE Trans Nanotechnol 2023.
[5] Farahani SS, Reshadinezhad MR, Fatemieh SE. New design for error-resilient approximate multipliers used in image processing in CNTFET technology. J Supercomput 2023;1–19.
[6] Fatemieh SE, Farahani SS, Reshadinezhad MR. LAHAF: Low-power, area-efficient, and high-performance approximate full adder based on static CMOS. Sustain Comput: Inform Syst 2021;30:100529.
[7] Rashidi H, Rezai A, Soltany S. High-performance multiplexer architecture for quantum-dot cellular automata. J Comput Electr 2016;15:968–81.
[8] Fatemieh SE, Reshadinezhad MR. Power-efficient, high-PSNR approximate full adder applied in error-resilient computations based on CNTFETs. In: 2020 20th International symposium on computer architecture and digital systems (CADS). IEEE; 2020, p. 1–5.
[9] Radakovits D, TaheriNejad N, Cai M, Delaroche T, Mirabbasi S. A memristive multiplier using semi-serial imply-based adder. IEEE Trans Circuits Syst I Regul Pap 2020;67(5):1495–506.
[10] Teimoory M, Amirsoleimani A, Shamsi J, Ahmadi A, Alirezaee S, Ahmadi M. Optimized implementation of memristor-based full adder by material implication logic. In: 2014 21st IEEE International conference on electronics, circuits and systems (ICECS). IEEE; 2014, p. 562–5.
[11] Haj-Ali A, Ben-Hur R, Wald N, Ronen R, Kvatinsky S. Imaging: In-memory algorithms for image processing. IEEE Trans Circuits Syst I Regul Pap 2018;65(12):4258–71.
[12] Kvatinsky S, Belousov D, Liman S, Satat G, Wald N, Friedman EG, Kolodny A, Weiser UC. MAGIC—Memristor-aided logic. IEEE Trans Circuits Syst II 2014;61(11):895–9.
[13] Fatemieh SE, Reshadinezhad MR, TaheriNejad N. Fast and compact serial IMPLY-based approximate full adders applied in image processing. IEEE J Emerg Sel Top Circuits Syst 2023;13(1):175–88.
[14] Karimi F, Faghih Mirzaee R, Fakeri-Tabrizi A, Roohi A. Design and evaluation of ultra-fast 8-bit approximate multipliers using novel multicolumn inexact compressors. Int J Circuit Theory Appl 2023.
[15] Pal S, Bose S, Islam A. Design of memristor based low power and highly reliable ReRAM cell. Microsyst Technol 2019;1–15.
[16] Islam MM. Design and implementation of memristor (Ph.D. thesis), GREEN UNIVERSITY OF BANGLADESH; 2018.
[17] Strukov DB, Snider GS, Stewart DR, Williams RS. The missing memristor found. Nature 2008;453(7191):80–3.
[18] Karimi A, Rezai A. Novel design for a memristor-based full adder using a new IMPLY logic approach. J Comput Electr 2018;17(3):1303–14.
[19] Padberg H, Regev A, Piccolboni G, Bricalli A, Molas G, Nodin JF, Kvatinsky S. Experimental demonstration of non-stateful in-memory logic with 1T1R OxRAM valence change mechanism memristors. IEEE Trans Circuits Syst II 2023.
[20] Amirsoleimani A, Ahmadi M, Teimoory M, Ahmadi A. Memristor-based 4: 2 compressor cells design. In: 2016 IEEE International symposium on circuits and systems (ISCAS). IEEE; 2016, p. 1242–5.
[21] Huang P, Kang J, Zhao Y, Chen S, Han R, Zhou Z, Chen Z, Ma W, Li M, Liu L, et al. Reconfigurable nonvolatile logic operations in resistance switching crossbar array for large-scale circuits. Adv Mater 2016;28(44):9758–64.
[22] Ali KA. New design approaches for flexible architectures and in-memory computing based on memristor technologies (Ph.D. thesis), Ecole nationale supérieure Mines-Télécom Atlantique; 2020.
[23] Kvatinsky S, Wald N, Satat G, Kolodny A, Weiser UC, Friedman EG. MRL—Memristor ratioed logic. In: 2012 13th International workshop on cellular nanoscale networks and their applications. IEEE; 2012, p. 1–6.
[24] Rohani SG, Taherinejad N, Radakovits D. A semiparallel full-adder in IMPLY logic. IEEE Trans Very Large Scale Integr (VLSI) Syst 2019;28(1):297–301.
[25] Rohani SG, TaheriNejad N. An improved algorithm for IMPLY logic based memristive full-adder. In: 2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE). IEEE; 2017, p. 1–4.

[26] Fatemieh SE, Reshadinezhad MR, TaheriNejad N. 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, p. 3115–9.
[27] Kumar AP, Aditya B, Sony G, Prasanna C, Satish A. Estimation of power and delay in CMOS circuits using LCT. Indones J Electr Eng Comput Sci 2019;14:990–8.
[28] Weber AG. The USC-SIPI image database: Version 5. 2006, http://sipi.usc.edu/database/.
[29] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 2004;13(4):600–12.
[30] Mittal S. A survey of techniques for approximate computing. ACM Comput Surv 2016;48(4):1–33.

Samane Asgari was born in Qazvin, Iran, in 1993. She received her B.Sc degree in computer hardware engineering from the University of Guilan, Gilan, and her M.Sc degree in computer architecture from the University of Mohaghegh Ardabili, Ardabil, in 2016 and 2019, respectively. She is now a Ph.D. candidate at the University of Isfahan, Iran. Her research interests are in-memory computing and approximate and accurate digital arithmetics.

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.

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, Digital VLSI, Computer Arithmetic, and Quantum Computing and Reversible Circuits.