Journal of Computational Electronics (2018) 17:1303–1314
https://doi.org/10.1007/s10825-018-1198-5

Novel design for a memristor-based full adder using a new IMPLY logic approach

Ahmad Karimi[1] · **Abdalhossein Rezai**[1]

Published online: 20 June 2018

Abstract
This paper presents and evaluates a novel approach based on material implementation to improve the performance of a 1-bit memristor-based full adder. In addition, two new efficient architectures are presented and evaluated for an 8-bit memristor-based full adder, serial memristor-based full adder and parallel-serial memristor-based full adder. Using the proposed approach, the number of required steps to accumulate the sum and carry-out in the proposed serial and parallel-serial 8-bit memristor-based full adder architectures is 168 and 56, respectively. Moreover, the number of required memristors in the proposed serial and parallel-serial memristor-based full adder architectures is 19 and 33, respectively. The evaluation results show that both delay and density are improved in the proposed architectures compared to other memristor-based full adder architectures.

Keywords Memristors · IMPLY logics · Crossbar arrays · Nano-scale logic operation

1 Introduction

Moore's law anticipates that the number of transistors in a specific size doubles every 2 years [1,2]. However, the limitations of CMOS technology require looking for other technologies that can help to overcome this issue [2,3]. Some of these approaches are as follows: silicon-on-insulator field effect transistors (SOI-FETs) [4], carbon nanotube field effect transistors (CNTFETs) [2,3], graphene nanoribbon field effect transistors (GNTFETs), hybrid nano/CMOS technologies [5–8] and quantum-dot cellular automata (QCA) [2]. These new technologies are creating new opportunities to scale transistors into smaller sizes [1–8].

One of the new trending hybrid nano/CMOS devices is the memristor. A memristor is a two-terminal device that was first introduced as the missing circuit element by Leon Chua in 1971 [6,7]. By establishing a relation between electric flux (charge) and the magnetic flux, the memristor has both memory and resistive properties [8]. Memristors are capable of building high-density resistive memory arrays [9,10], which have been integrated on a CMOS subsystem [11,12]. As the current flowing through these memories is changed, the resistive value of these memories is changed as well [13,14]. The memristor's I–V curve is shown in Fig. 1 [14].

If the applied voltage (Vm) to the memristor is larger than the memristor's threshold voltage (V_{SET}), its memristance switches to the lowest possible value (R_{ON}). On the other hand, when Vm is smaller than the negative threshold voltage V_{RESET}, the memristance reaches its highest possible value (R_{OFF}) [14,15].

One of the approaches that helps improved logic operations based on memristive systems is called material implication (IMP). The IMP logic has been the basic block for arithmetic circuits such as multipliers [16], full adders [15–23] and shift registers [24]. Many researchers have designed different logic gates such as NOR–OR [15,25,26], AND–NAND [27,28] and XNOR–XOR [29] with different architectures. Some of these ideas are based only on IMP logic and memristors [16,19,30]; others have used memristors with a CMOS subsystem [18–20]. It should be mentioned that pipeline architecture has been utilized to map logic designs into hybrid nano/CMOS architectures [29]. Each of these designs has its own advantages and disadvantages. When using IMP logic due to the usage of just memristors, these devices can be scaled to smaller sizes in comparison with Nano/CMOS architectures [31].

✉ Ahmad Karimi
ahmad.karimi@jdeihe.ac.ir

✉ Abdalhossein Rezai
rezaie@acecr.ac.ir

[1] ACECR Institute of Higher Education, Isfahan Branch, Isfahan 84175-443, Iran

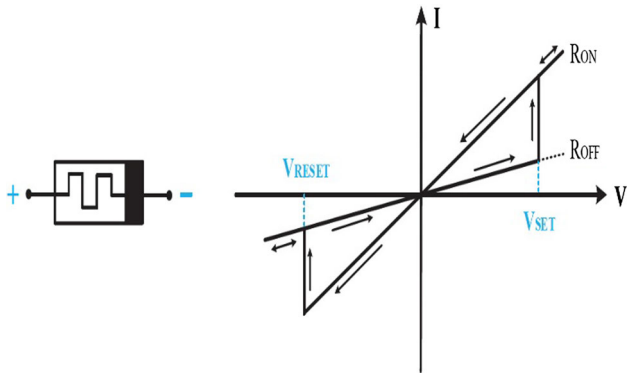
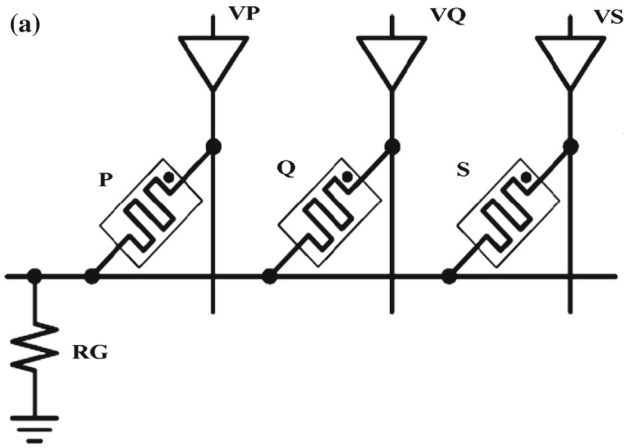


Fig. 1 The memristor's I–V curve (threshold-type) [14]



(b)

Switches	P	Q	S
State of switches	p	q	0

Fig. 2 The memristor-based NAND gate **a** cross bar circuit, **b** initial states after the FALSE operation [14]

Recently, researchers at Hewlett–Packard [14] demonstrated that a NAND operation could be executed [14] by utilizing only three memristors, one resistor and a three-state voltage driver (V_{Set}, V_{Clear} and V_{Cond}). The NAND gate operand can be used to construct other Boolean logic gates. Therefore, it is possible to build different logic gates by a network of NAND gates [14]. The advantages of this method are flexibility (implementable onto the nano layer), universality, and general-purpose application, applicable to logical and arithmetic units [31].

This paper proposes a novel 1-bit memristor-based full adder architecture, which is based on the IMP logic operations. This architecture is utilized to design two novel 8-bit full adder architectures, a serial memristor-based full adder and a parallel-serial memristor-based full adder. The proposed memristor-based full adders' architectures are formed by memristors and crossbar nanowires. The evaluation results show that the proposed full adder architectures provide an improvement compared to other full adder architectures in terms of layout area and speed.

The remainder of this paper is organized as follows. Section 2 provides a background about memristors and memristor-based full adder architectures. Section 3 presents the rules of IMP logic operations. The novel full adder designs and their architectures are presented in Sect. 4. Section 5 compares the proposed memristor-based full adder architectures with other full adder architectures. Finally, Sect. 6 concludes this paper.

2 Background

2.1 Memristor-based logic and hybrid architecture

A simple memristor-based NAND gate is shown in Fig. 2 [14]. Recall that a NAND gate is a basic gate to implement logic circuits.

Three memristors are enough to perform a NAND logic operation. The values of p and q are stored in switches (memristors) P and Q, respectively. The output of this NAND operation is stored in memristor S with the logic value of s [14]. Before initiating the IMP logic operation, it is necessary to perform a FALSE operation that clears the value of memristor S. This operation is done to make sure that the new information is successfully stored in this memristor [14]. The steps for this FALSE operation are as follows [14]:

$$\begin{aligned} &\text{Step1} && s' == p \rightarrow s \ (p \text{ IMP } s) \\ &\text{Step2} && s'' == q \rightarrow s' \ (q \text{ IMP } s') \\ &\text{Overall} && s'' == q \rightarrow (p \rightarrow s) == q \text{ IMP } (p \text{ IMP } s) \end{aligned}$$

In the second step s'' denotes the output value of memristor S. This output (s'') value is the same as the output of the NAND operation on logic states p and q. Based on this fully functional universal NAND logic gate, it is possible to construct any Boolean logic gates [14,32].

The conditions and the value of these circuits' elements depicted in Fig. 2 must be as follows [30]:

$$R_{ON}\frac{V_{SET}-V_{ON}}{V_{ON}-[V_{SET}-V_{COND}]} < RG < R_{OFF}\frac{V_{SET}-V_{ON}}{2V_{ON}-[V_{SET}-V_{COND}]} \quad (1)$$

$$\frac{V_{SET}}{V_{COND}} < \frac{R_{OFF}}{R_{ON}} \quad (2)$$

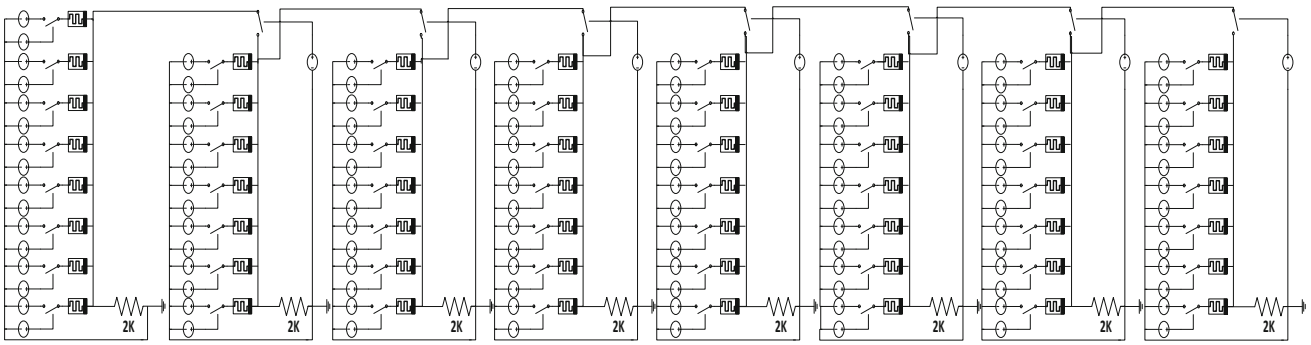


Fig. 3 The utilized 8-bit memristor-based full adder in [20]

Based on these conditions, the chosen value for RG that is considered for the circuits in this paper is as follows:

$$\mathrm{RG} = \sqrt{R_{\mathrm{ON}} \cdot R_{\mathrm{OFF}}} \tag{3}$$

As the basics of many different arithmetic operations are full adders, it is a necessity to improve the performance of these designs. There are two main characteristics of full adders, which require improving: (a) the required layout area, and (b) the speed of this device. In the following, some of the previous memristor-based full adder architectures are considered.

2.2 Previous memristor-based full adder architectures

The basic formulas to calculate the sum and the carry-out of the full adder are as follows [19]:

$$S = c_{\mathrm{in}} \oplus (p \oplus q) \tag{4}$$

$$c_{\mathrm{out}} = p \cdot q + c_{\mathrm{in}} (p \oplus q) \tag{5}$$

here, p and q denote two input logic values. C_{in} is the input carry. The outputs are stored in S, which is the sum and C_{out}, which denotes the output carry.

Recently, many researchers have focused on the topic of memristor-based arithmetic computations, especially full adders [16–23]. Some of these methods and their advantages and disadvantages are discussed in the following.

The recent research on IMP-based full adders are explained in [16,20,22]. It should be noted that this research is the most recent and has the best performance amongst the rest.

In [16], the approach of conjunctive normal form (CNF) is utilized. The sum and carry in this method is computed as follows:

$$S = (c_{\mathrm{in}} + a + b) \cdot \left(\overline{c_{\mathrm{in}}} + a + \bar{b}\right) \cdot (\overline{c_{\mathrm{in}}} + \bar{a} + b) \cdot \left(c_{\mathrm{in}} + \bar{a} + \bar{b}\right) \tag{6}$$

$$C_{\mathrm{out}} = (c_{\mathrm{in}} + a) \cdot (a + b) \cdot (c_{\mathrm{in}} + b) \tag{7}$$

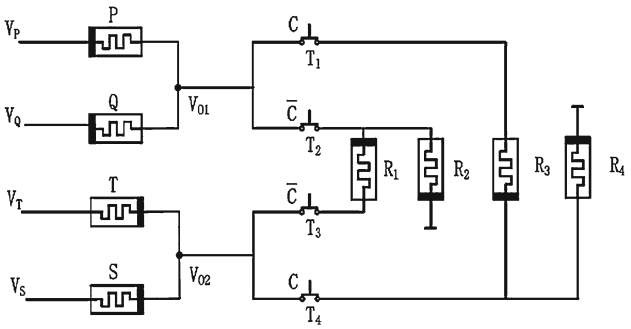


Fig. 4 The utilized 1-bit memristor-based full adder in [17]

Based on the CNF, 89 steps in total are required to perform a 1-bit full adder, 48 steps to calculate sum and 39 steps to calculate carry-out. The remaining steps are used for the FALSE operation on the working memristors. The utilized full adder in [16] suffers from a huge delay time. Therefore, the speed (response time) of this approach is very low.

In [20], the authors have improved the speed of the algorithm provided in [16] by presenting a parallel structure. This parallel structure requires 7N + 1 memristors and 2N + 19 steps. Although this method improves the speed of full adder, but it requires a large number of memristors (7N + 1), switches (8N) and drivers (15N + 1). This method structure is shown in Fig. 3.

The structure and algorithm utilized in [20] suffers from a huge layout area, which is not efficient enough for processors and other computational applications.

In [22], a new IMP logic-based full adder is utilized that can reduce the number of memristors and the required steps to calculate sum and carry-out.

The utilized formula for sum and carry are as follows [22]:

$$S = ((\bar{a}\ \mathrm{IMP}\ b)\ \mathrm{IMP}((a\ \mathrm{IMP}\ \bar{b})\ \mathrm{IMPc}))\ \mathrm{IMP}\ \overline{((a \oplus b)\mathrm{IMP}\ \bar{c})} \tag{8}$$

$$\mathrm{Cout} = \overline{\left[(\bar{a}\ \mathrm{IMP}\ b)\ \mathrm{IMP}\ \overline{((a\ \mathrm{IMP}\ \bar{b})\ \mathrm{IMP}\ c)}\right]} \tag{9}$$

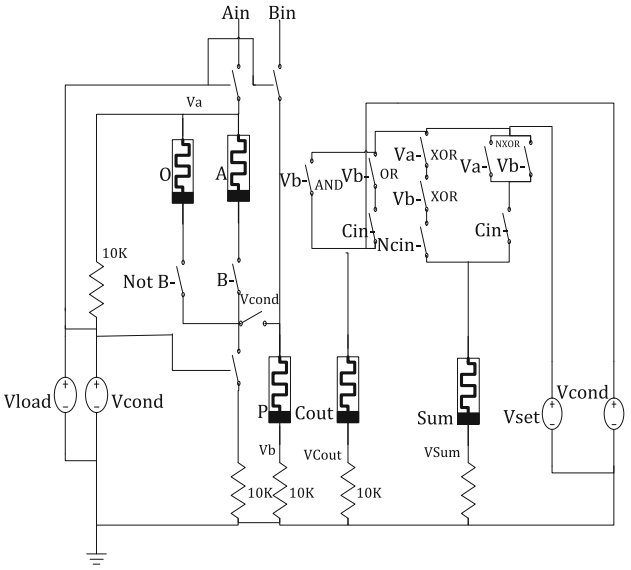


Fig. 5 The utilized 1-bit memristor-based full adder in [18]

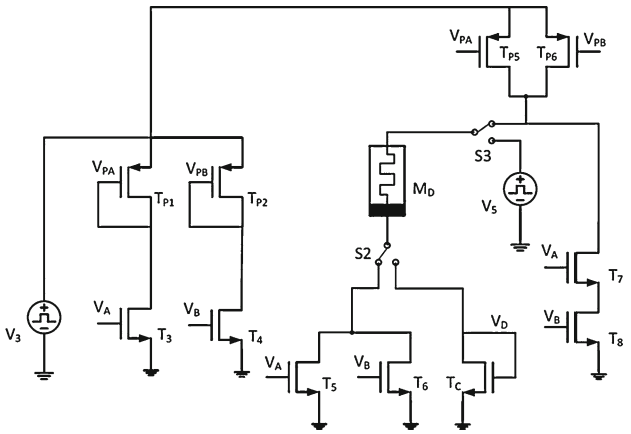


Fig. 6 The utilized memristor-based 1-bit full adder (accumulating intermediate sum) in [19]

This method requires many steps to calculate the outputs [22].

The advantage of IMP-based full adder in comparison with other forms of memristor-based full adders is that they only require an memristor in their structure, and the serial approach presented in these works requires the least possible number of memristors. As a result, the layout area is less than other architectures. Therefore, at first the serial IMP-based full adder is optimized, and then novel serial and parallel-serial architectures are proposed in this paper.

In [17], the authors have proposed an XOR gate and utilized it in a full adder to reduce the number of required steps to accumulate the sum and carry-out.

Figure 4 shows the utilized memristor-based full adder in [17].

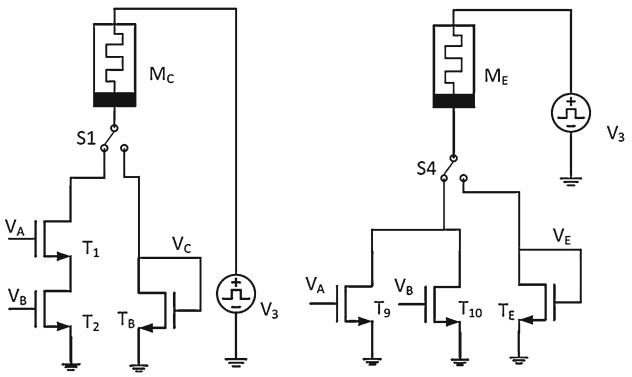


Fig. 7 The utilized 1-bit memristor-based full adder (accumulating intermediate carry) in [19]

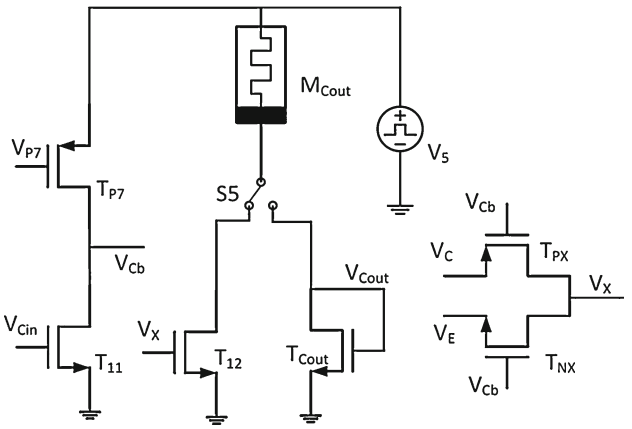


Fig. 8 The utilized 1-bit memristor-based full adder (accumulating final carry-out) in [19]

As it is shown in Fig. 4, this full adder requires four memristors to control the voltage that arrives at V_{O1} and V_{O2}, four memristors to store the results (R_1, R_2, R_3 and R_4) and 4 switches (T_1, T_2, T_3 and T_4).

The problem with this device is that 64 memristors and 32 switches are required for an 8-bit memristor-based full adder. This means that a huge layout area is necessary for this design.

In [18], the authors have designed a full adder that its baseline (for 1-bit full adder) requires four memristors, five resistors, 13 switches and four drivers. This architecture is shown in Fig. 5.

Although the output is driven out only by three steps, this full adder suffers from a large layout area and huge power consumption. This is due to the usage of many resistors in its structure.

In [19], the authors have used hybrid nano/CMOS architecture in their designs, which are shown in Figs. 6, 7 and 8.

Figure 6 shows the operation in charge of accumulating the intermediate sum. Then, the intermediate and final carry-out are calculated in Figs. 7 and 8, respectively. Finally, the output

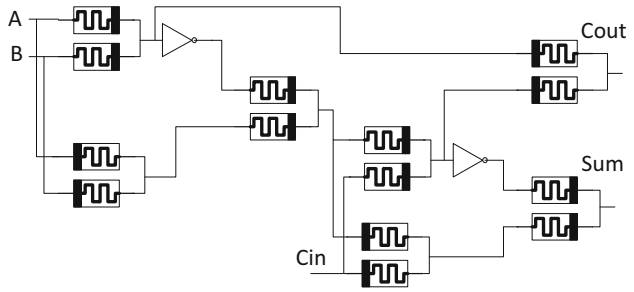


Fig. 9 The utilized 1-bit memristor-based serial adder in [20]

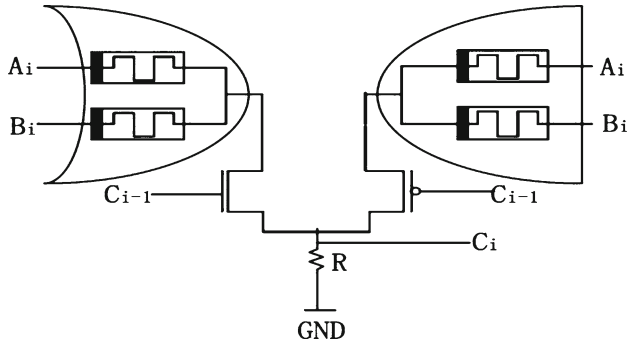


Fig. 10 The utilized 1-bit memristor-based full adder (accumulating carry) in [21]

sum is accumulated by using a same architecture similar to the one shown in Fig. 6.

The memristors are used to save the data and the CMOSs' architecture is responsible for the logic gates operation. The utilized method uses current to change memristors state (ON and OFF states). The problem with this approach is that it utilizes CMOS technology in its architecture. Therefore, the same limitations that forced us to look toward a new technology such as memristors are still unsolved. Another disadvantage of this method is that it requires a huge layout area.

In [20], another hybrid nano/CMOS approach is utilized to accumulate the outputs, which is shown in Fig. 9.

The utilized architecture in [20] contains 14 memristors and four MOSFETs (including NOT gates). This architecture requires huge layout area, but the advantage of this method is that it reduces the number of necessary steps to accumulate the outputs.

In [21], the author utilized XOR and XNOR gates to design full adder. The mentioned method is shown in Figs. 10 and 11.

Based on the Figs. 10 and 11, this method is based on Nano/CMOS structure. Although the speed of this Full adder is improved, it requires a large layout area.

The main problem with all of these designs beside the required layout area is that they suffer from a large number of steps toward accumulating the sum and carry-out [16–23]. These problems have been considered, and the quality of the proposed full adders in this paper is improved in comparison with other similar works.

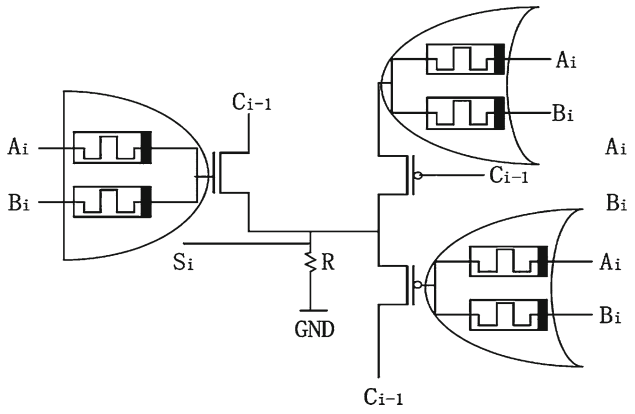


Fig. 11 The utilized 1-bit memristor-based full adder (accumulating sum) in [21]

3 Memristor-based logic operations

3.1 IMP logic rules

With the help of IMP rules and based on the rules provided in this section, the optimization process could be done on any memristor device or to change any given Boolean logic operation to an IMP logic operation.

One of the main properties of IMP operation is expressed as follows [30]:

$$p \to q \ \ (p \ \text{IMP} \ q) = \bar{q} \to \bar{p} \ \ (\bar{q} \ \text{IMP} \ \bar{p}) \quad \text{(NOT gate in Boolean logic)} \tag{10}$$

$$p \to \bar{p} \ \ (p \ \text{IMP} \ \bar{p}) = \bar{p} \ \text{or} \ p \to 0 \ \ (p \ \text{IMP} \ 0) = \bar{p} \quad \text{(Buffer gate in Boolean logic)} \tag{11}$$

$$p \to p \ \ (p \ \text{IMP} \ p) = p \tag{12}$$

The proof of these rules is given by its truth table shown in columns 5–8 in Table 1.

The following rules are also driven out by checking the truth table for each and every one of them. Their proofs lie within Tables 2 and 3

$$(p \ \text{IMP} \ \bar{q}) \ \ \text{IMP} \ \bar{q} == (\bar{p} \ \text{IMP} \ q) \ \ \text{IMP} \ p == q \ \text{IMP} \ p \tag{13}$$

$$(p \ \text{IMP} \ \bar{q}) \ \ \text{IMP} \ \bar{p} == (\bar{p} \ \text{IMP} \ q) \ \ \text{IMP} \ q == p \ \text{IMP} \ q \tag{14}$$

$$\bar{p} \ \text{IMP} \ q == \bar{q} \ \text{IMP} \ p \ \text{(OR gate in Boolean logic)} \tag{15}$$

$$p \ \text{IMP} \ \bar{q} == q \ \text{IMP} \ \bar{p} \ \text{(NAND gate in Boolean logic)} \tag{16}$$

$$\overline{\bar{p} \ \text{IMP} \ q} == \overline{\bar{q} \ \text{IMP} \ p} \ \text{(NOR gate in Boolean logic)} \tag{17}$$

$$\overline{p \ \text{IMP} \ \bar{q}} == \overline{q \ \text{IMP} \ \bar{p}} \ \text{(AND gate in Boolean logic)} \tag{18}$$

$$p \ \text{IMP} \ \ (q \ \text{IMP} \ z) == q \ \text{IMP} \ \ (p \ \text{IMP} \ z) \tag{19}$$

Table 1 Truth table for IMP logic rules (part 1)

p	q	$\bar{p}$	$\bar{q}$	p IMP p	p IMP $\bar{p}$	p IMP q	$\bar{q}$ IMP $\bar{p}$	p IMP $\bar{q}$	$\bar{p}$ IMP q	$\bar{q}$ IMP p	$\overline{\bar{p}\ \text{IMP}\ q}$
0	0	1	1	0	1	1	1	1	0	0	1
0	1	1	0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	0	1	1	1	0
1	1	0	0	1	0	1	1	0	1	1	0

Table 2 Truth table for IMP logic rules (part 2)

p	q	$\overline{\bar{q}\ \text{IMP}\ p}$	$\overline{(p\ \text{IMP}\ \bar{q})}$	$(p\ \text{IMP}\ q)\ \text{IMP}\ \overline{(\bar{p}\ \text{IMP}\ \bar{q})}$	$(p\ \text{IMP}\ q)\ \text{IMP}\ \overline{(q\ \text{IMP}\ p)}$	$(p\ \text{IMP}\ q)\ \text{IMP}\ \overline{(q\ \text{IMP}\ p)}$	$(p\ \text{IMP}\ \bar{q})\ \text{IMP}\ \overline{(\bar{p}\ \text{IMP}\ q)}$
0	0	1	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	0	1	1	1	0
1	1	0	1	0	0	0	1

Table 3 Truth table for IMP logic rules (part 3)

p	q	z	p IMP $(q$ IMP $z)$	q IMP $(p$ IMP $z)$	$(z$ IMP $\bar{p})$ IMP $\overline{(\bar{z}\ \text{IMP}\ \bar{q})}$ If $(z = 0$ then output $= q)$ If $(z = 1$ then output $= p)$
0	0	0	1	1	0
0	0	1	1	1	0
0	1	0	1	1	1
0	1	1	1	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	1	1	1

$$(p\ \text{IMP}\ \bar{q})\ \text{IMP}\ \bar{p} == (\bar{p}\ \text{IMP}\ q)\ \text{IMP}\ q \tag{20}$$

$$(p\ \text{IMP}\ q)\ \text{IMP}\ \overline{(\bar{p}\ \text{IMP}\ \bar{q})}\ \text{(XOR gate in Boolean logic)} \tag{21}$$

$$(p\ \text{IMP}\ q)\ \text{IMP}\ \overline{(q\ \text{IMP}\ p)}\ \text{(XOR gate in Boolean logic)} \tag{22}$$

$$\overline{(\bar{p}\ \text{IMP}\ q)\ \text{IMP}\ \overline{(p\ \text{IMP}\ \bar{q})}}\ \text{(XOR gate in Boolean logic)} \tag{23}$$

$$(p\ \text{IMP}\ \bar{q})\ \text{IMP}\ \overline{(\bar{p}\ \text{IMP}\ q)}\ \text{(XNOR gate in Boolean logic)} \tag{24}$$

$$(z\ \text{IMP}\ \bar{p})\ \text{IMP}\ \overline{(z\ \text{IMP}\ \bar{q})}\ \text{(MUX in Boolean logic)} \tag{25}$$

It should be noted that there is no limit for the rules of IMP logic operations. Their combinations can provide endless IMP logic rules; it is possible to improve many devices, such as previous full adders based on the provided equations [16–23].

3.2 Full adder formulas

The basic formulas for sum and carry-out in the full adder are given in (4) and (5). Based on these formulas, the sum in IMP logic is equivalent to all of the given formulas in the following:

$$S = (c\ \text{IMP}\ (p \oplus q))\ \text{IMP}\ \overline{\left(\bar{c}\ \text{IMP}\ \overline{(p \oplus q)}\right)} \tag{26}$$

$$S = (c\ \text{IMP}\ (p \oplus q))\ \text{IMP}\ \overline{((p \oplus q)\ \text{IMP}\ c)} \tag{27}$$

$$S = \overline{(\bar{c}\ \text{IMP}\ (p \oplus q))\ \text{IMP}\ \overline{\left(c\ \text{IMP}\ \overline{(p \oplus q)}\right)}} \tag{28}$$

The carry-out is also equivalent to the following formulas:

$$\text{Cout} = \left[c\ \text{IMP}\ \overline{(p \oplus q)}\right]\ \text{IMP}\ \left(\overline{p\ \text{IMP}\ \bar{q}}\right) \tag{29}$$

$$\text{Cout} = (p\ \text{IMP}\ \bar{q})\ \text{IMP}\overline{\left[c\ \text{IMP}\ \overline{(p \oplus q)}\right]} \tag{30}$$

If both Eqs. (29) and (30) are selected for the calculation of sum and carry-out, then some parts of both equations are similar. Therefore, it is reasonable to accumulate both sum and carry-out at the same time.

Another factor that can be helpful toward designing an optimized 1-bit full adder is the number of memristors in its architecture. If it is necessary for the full adder to save the input data value as well as the outputs, seven memristors crossbar array are required, but if saving the inputs is not a necessity, only five memristors are enough to design a serial IMP logic-based full adder. It should be mentioned that saving the input data means the value of input memristors are not changed; therefore, they can be used in other parallel

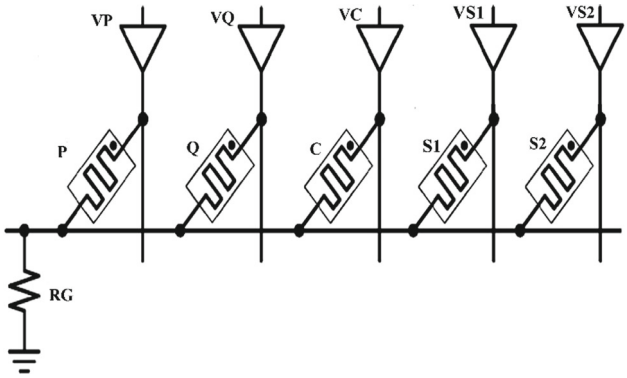


Fig. 12 The proposed 1-bit memristor-based full adder architecture

systems. As most of the time the output is only required to be stored [18,20], the focus is on saving the outputs in this paper. This approach helps to reduce the number of utilized memristors.

4 The proposed memristor-based full adder architectures

Most of the utilized memristor-based full adder architectures in previous works suffer from a huge layout area or a slow response. Therefore, the objectives design in the proposed full adder architectures are (a) reducing the required layout area and (b) improving the speed of full adder architectures.

Table 5 The parameters of memristor provided in [33]

k_{on}	-8000 m/s	k_{off}	0.0403 m/s
v_{on}	-0.53 V	v_{off}	0.5 V
w_{on}	0	w_{off}	10 nm
α_{on}	3	α_{off}	1
R_{on}	0.1 kΩ	R_{off}	2.5 kΩ

4.1 The proposed 1-bit memristor-based full adder architecture

The proposed 1-bit memristor-based full adder architecture is shown in Fig. 12.

Each step after clearing $S1$ and $S2$ ($S1 = 0$ and $S2 = 0$), toward accumulating the output sum and carry-out are shown in Table 4.

As it is shown in Fig. 12 and Tables 4 and 5 memristors and 21 steps are required to accumulate the outputs in this architecture. The unique property of the proposed memristor-based full adder is that the steps towards accumulating sum and carry can be divided into three sec-

Table 4 The proposed 1-bit memristor-based full adder's algorithm and the required steps to accumulate sum and carry

Step	Operation	Equivalent IMP logic
1	p IMP $S1 == S1'$	$\bar{p}$
2	q IMP $S2 == S2'$	$\bar{q}$
3	$S1'$ IMP $q == q'$	$\bar{p}$ IMP q
4	p IMP $S2' == S2''$	p IMP $\bar{q}$
5	CLEAR $p \Rightarrow p = 0$	FALSE p
6	q' IMP $p == p'$	$\overline{(\bar{p}\ \text{IMP}\ q)}$
7	$S2''$ IMP $p == p''$	$\overline{(p \oplus q)}$
8	CLEAR $q \Rightarrow q = 0$	FALSE q
9	p'' IMP $q == q'$	$p \oplus q$
10	CLEAR $S1 \Rightarrow S1 = 0$	FALSE $S1$
11	C IMP $S1 == S1'$	$\bar{C}$
12	C IMP $p'' == p'''$	c IMP $\overline{(p \oplus q)}$
13	CLEAR $C \Rightarrow C = 0$	FALSE C
14	$S2''$ IMP $C == C'$	$\left(\overline{p\ \text{IMP}\ \bar{q}}\right)$
15	p''' IMP $C == C''$	Cout $= \left[c\ \text{IMP}\ \overline{(p \oplus q)}\right]$ IMP $\left(\overline{p\text{IMP}\bar{q}}\right)$
16	$S2''$ IMP $q == q'$	$\bar{c}$ IMP $(p \oplus q)$
17	CLEAR $S1 \Rightarrow S1 = 0$	FALSE $S1$
18	p''' IMP $S1 == S1'$	$\overline{\left(c\ \text{IMP}\ \overline{(p \oplus q)}\right)}$
19	q' IMP $S1' == S1''$	$(\bar{c}$ IMP $(p \oplus q))$ IMP $\overline{\left(c\ \text{IMP}\ \overline{(p \oplus q)}\right)}$
20	CLEAR $q \Rightarrow q = 0$	FALSE q
21	$S1''$ IMP $q == q'$	$S = \overline{(\bar{c}\ \text{IMP}\ (p \oplus q))\ \text{IMP}\ \overline{\left(c\ \text{IMP}\ \overline{(p \oplus q)}\right)}}$

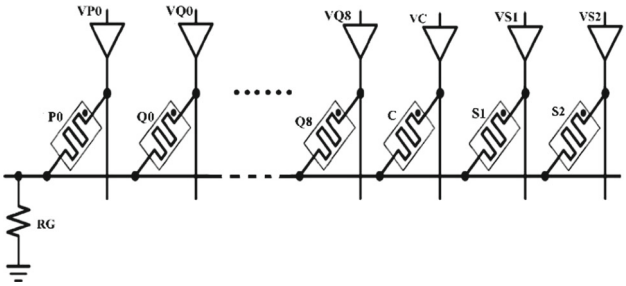


Fig. 13 The proposed serial 8-bit memristor-based full adder architecture

tions. The first section is in charge of calculating $p \oplus q$, the second section is in charge of calculating the output carry (Cout $= \left[c \text{ IMP } \overline{(p \oplus q)}\right] \text{IMP}(\overline{p \text{ IMP } \bar{q}})$), and finally, the last section is in charge of accumulating the sum $\left(S = \overline{(\bar{c} \text{ IMP } (p \oplus q))} \text{ IMP} \overline{\left(c \text{ IMP } \overline{(p \oplus q)}\right)}\right)$. Based on this algorithm, each of these divisions can operate simultaneously, which is further discussed in Sect. 4.3.

4.2 The proposed serial 8-bit memristor-based full adder

It should be noted that if the number of inputs in the memristor-based full adder architecture is increased, it is only necessary to increase the number of input memristors. For example, it is required to have 19 memristors (17 memristors as inputs and two working memristors) for an 8-bit full adder.

The proposed serial architecture is shown in Fig. 13. Based on Fig. 13, the proposed serial 8-bit memristor-based full adder architecture utilizes 19 memristors. It should be noted that as this is a serial approach, all of these steps must be retaken for each of the two inputs. For example, the number of steps to accumulate [$p_1 + q_1 +$ Carry-in or $p_2 + q_2 +$ Carry-in or $p_3 + q_3 +$ Carry-in] is 23 for 8-bit full adder architecture. Since this is a serial approach, each of these accumulations should be performed after the previous one is finished, which means it requires 182 steps ($21 + 7 \times 23 = 182$) in total for an 8-bit full adder. In addition, at the end of each sum and carry, memristors $S1$ and $S2$ must be cleared. Therefore, two additional steps must be done after the first calculation ($p_0 + q_0 +$ Carry-in).

4.3 The proposed 8-bit parallel-serial memristor-based full adder

Although the proposed serial approach has some advantages, there is room for further improvement. Another approach is to use parallel-serial architecture, instead of a serial one. In the proposed serial architecture, it is possible to calculate some of these steps simultaneously. Based on Table 4, from the first step to step 10, no changes happen in the carry value. In addition, the only value that must be presented for the next calculations is carry. Therefore, it is possible to calculate the first 10 steps at the same time for all arrays. When the carry-out value is calculated in step 15, the next array can start its calculation without interfering with the calculation of the sum. These five steps cannot be done simultaneously because the first two bits' carry-out is the next two bits' carry-in. The proposed design is shown in Fig. 14.

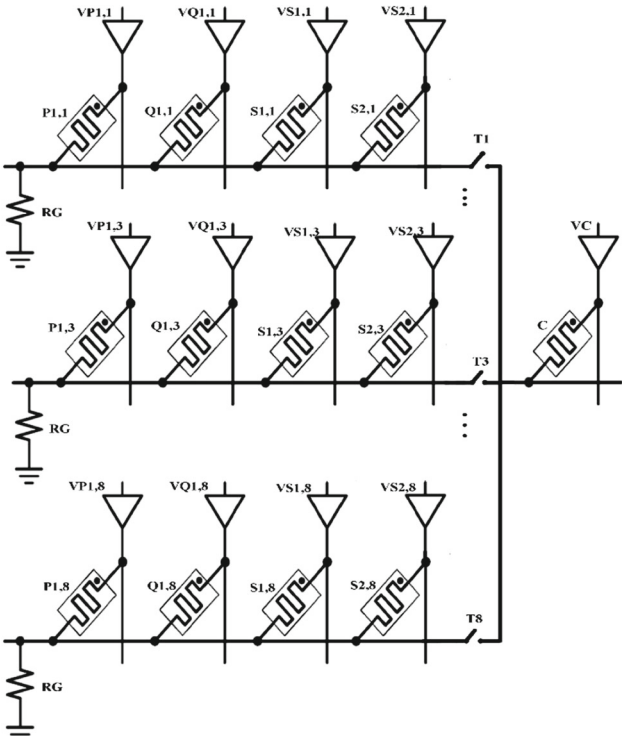


Fig. 14 The proposed 8-bit parallel-serial memristor-based full adder architecture

In the proposed parallel-serial architecture, 33 memristors are utilized, which means that the layout area is increased, but only 56 steps are required to accumulate the outputs. Some other modifications are also done such as adding connections between the rows of the crossbar, which are easily applicable. This approach is most suitable for fast applications that require a reasonable layout area.

5 Simulation results and comparison

The proposed memristor-based full adders are simulated in Pspice utilizing the model presented in [33] by VTEAM. The current–voltage curve of this model is also presented in Fig. 15.

The parameters of the provided memristor in [33] are shown in Table 5.

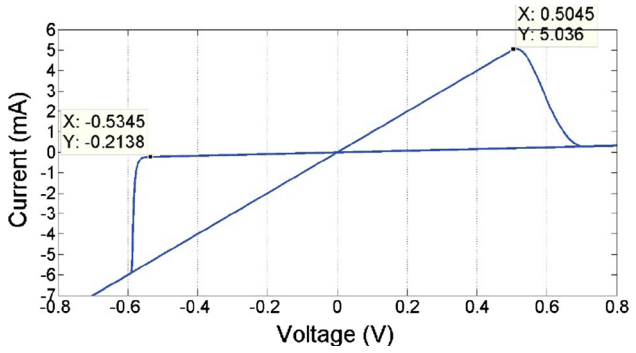


Fig. 15 The current–voltage curve of VTEAM's memristor [33]

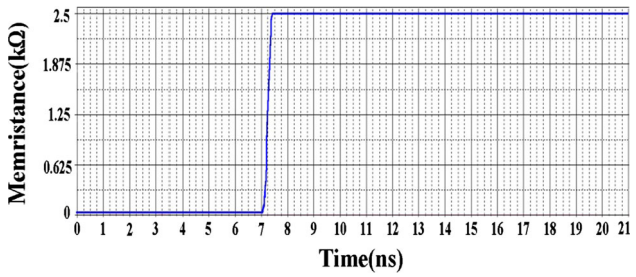


Fig. 16 The simulation results for the memristor Q's memristance in the proposed 1-bit memristor-based full adder when $p = 1, q = 1$ and $C_{in} = 0$

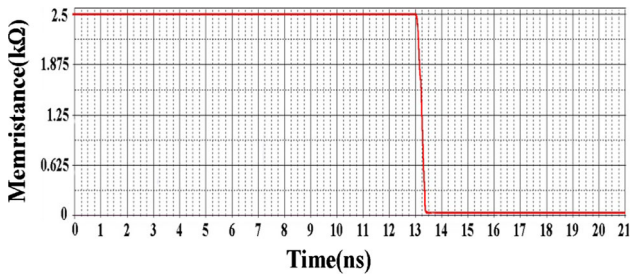


Fig. 17 The simulation results for the memristor C's memristance in the proposed 1-bit memristor-based full adder when $p = 1, q = 1$ and $C_{in} = 0$

The VTEAM memristor is utilized due to many factors, which are flexibility, generality, sufficiency, accuracy, etc. As shown in Table 5, it has two threshold voltages, which means it can be helpful to simulate physical behavior accurately [34].

It should be noted that the required time to change the state of a memristor depends mostly on its structure and components [30,35]. The authors of [35] showed that the required time to change the state is only 10 ps for a tantalum oxide memristor. The provided memristor in [33] requires less than 1 ns to change its state from high resistance (logic value of "0") to low resistance (logic value of "1").

Here, the outputs of 1-bit memristor-based full adder for the input logics of $p = 1$, $q = 1$, $C_{in} = 0$, are shown in Figs. 16 and 17. As the sum is stored in the input memris-

Table 6 Features of memristor-based full adder architectures

Reference	Number of memristors					Number of CMOS	Number of resistors	Number of steps		
	Input	Output	Reused	Work	Total			IMPLY	FALSE	Total
[23]	2n + 1	n + 1	1	2	3n + 3	–	1	15n	8n	23n
[17]	4n	4n	4n	4n	8n	–	–	2n	1	2n + 1
[20]	10n	4n	N	n	14n	12n	–	n	1	3n + 4
[18]	2n + 1	n	N	2n	5n	–	4n + 1	4n	1	4n + 1
[19]	3n	5n	–	5n	8n	32n	–	n	–	3n*
[21]	10n	–	–	10n	10n	7n	2n	n	–	n + 1
This paper (serial architecture)	2n + 1	n + 1	N		2n + 3	–	1	14n	7n	21n
This paper (parallel-serial architecture)	2n + 1	n + 1	N	2n	4n + 1	–	n	7n + 7	2n + 5	5n + 16**

* This method utilizes current to drive out the output, and it doesn't use IMP logic; therefore, the number of steps are actually the number of operations done to accumulate the output
** It should be noted that in the proposed parallel-serial architecture some steps happens simultaneously, therefore the total number of steps is not equal to the sum of FALSE and IMPLY steps

Table 7 Layout area comparison of the memristor-based full adder architectures

Reference	Proposed serial layout area improvement			Proposed parallel-serial layout area improvement		
	1-bit (%)	4-bit (%)	8-bit (%)	1-bit (%)	4-bit (%)	8-bit (%)
[23]	16	26	29.6	16	− 13.34	− 22.23
[17]	37	65.6	70.32	37	46.87	48.5
[20]	64	80.35	83	64	69.6	70.53
[18]	0	45.0	52.5	0	15	17.5
[19]	37	65.6	70.32	37	46.87	48.5
[21]	50	72.5	76.25	50	57.5	58.75

tor and the carry-out is stored in the input carry, these two memristors' resistances are shown in the following.

The state of memristor Q or its resistance, which stores the accumulated sum at the end of 21 steps, is depicted in Fig. 16.

As it is shown in Fig. 16, the input value of the memristor Q is equivalent to 1 (R_{ON}), at the beginning of step 8, which clears the value of this memristor, its state changes to represent logic 0 (R_{OFF}). As the output of step 9 (p'' IMP $q == q'$), step 16 ($S2''$ IMP $q == q'$), step 20 (CLEAR $q \Rightarrow q = 0$) and step 21 $\left(\overline{(\bar{c}\ \text{IMP}\ (p \oplus q))\ \text{IMP}\ \overline{\left(c\ \text{IMP}\ \overline{(p \oplus q)}\right)}}\right)$ are all equal to the logic value of 0, the logic value of memristor Q does not change. Therefore, the resistance of this memristor stays unchanged.

In addition, the resistance of memristor C, which stores the accumulated carry-out at the end of 15 steps, is shown in Fig. 17.

As the input carry's value is equal to 0, the resistance of memristor C starts at R_{OFF}. This value stays the same until step 15. This is because the state of memristor Q can only be changed in the step 13 (CLEAR $C \Rightarrow C = 0$), step 14 ($\overline{p\ \text{IMP}\ \bar{q}}$) and step 15 $\left(\text{Cout} = \left[c\ \text{IMP}\ \overline{(p \oplus q)}\right]\ \text{IMP}\ \left(\overline{p\ \text{IMP}\ \bar{q}}\right)\right)$. Since the logic state of steps 13 and 14 are equal to 0, the logic state of this memristor stays unchanged until the step 15 is initialized with the logic state of 1. In the following steps, this value does not change and stays the same.

The features of the proposed n-bit memristor-based full adder architectures and the previous works are shown in Table 6.

The layout improvements of the proposed memristor-based full adder architectures are shown in Table 7.

Based on these results the proposed 8-bit serial memristor-based full adder architecture has the following distinctive characteristics:

- The number of required memristors to accumulate sum and carry-out of the proposed 8-bit serial memristor-based full adder architecture is lower than similar works, which prove that the layout area is reduced. This improvement in comparison with [17–20,23] and [21] is 70.32, 52.5, 70.32, 83, 29.6 and 76.25%, respectively.
- The proposed 8-bit serial memristor-based full adder architecture requires only 168 steps to accumulate the output. Therefore, the speed of this full adder is also faster than the previous works. This improvement in comparison with [23] is by about 5%.

In addition, the proposed 8-bit parallel-serial memristor-based full adder architecture has the following distinctive characteristics:

- The speed of the proposed 8-bit parallel-serial memristor-based full adder architecture is improved in comparison with [23] by about 64.48%.
- It also requires smaller layout area in comparison with the former designs in [17–21]. The layout area is improved by about 48.5, 17.5, 48.5, 70.53 and 58.75% in comparison with [17–21], respectively.
- Although the layout area of the proposed 8-bit parallel-serial memristor-based full adder is increased, the required steps of this full adder in comparison with [23], is reduced by about 64.48%.

These comparisons prove that both of these methods have their own advantages. If only the layout area is important for the full adder, the serial approach is the obvious choice. If it is required to have both fast responses and a suitable size, the parallel-serial approach has the best performance compared to other memristor-based full adder architectures.

6 Conclusions

In this paper, different memristor-based full-adder architectures have been studied. To improve the performance of the memristor-based full adder, a novel 1-bit IMP logic memristor-based full adder was designed. Based on the proposed 1-bit full adder, two novel 8-bit full adder architectures were designed, a serial memristor-based full adder and a parallel-serial memristor-based full adder. In the proposed

8-bit serial and parallel-serial method, the number of memristors and the necessary steps to accumulate sum and carry-out are reduced. The comparison of layout area based on the number of utilized memristors, between the proposed 8-bit serial approach and the former architectures shows, at least 29.6% in comparison with [23] and at best 83% reduction in comparison with [20], respectively. This approach also has reduced the number of necessary steps to accumulate the output by about 5% in comparison with [23].

In addition, the layout area of the proposed 8-bit parallel-serial architecture improvement in worst and best scenarios is by about 17.5 and 70.53 % in comparison with [18,20], respectively. The speed of this architecture was also improved in comparison with [23] by about 64.48%.

References

1. Karimi, A., Rezai, A., Hajhashemkhani, M.M.: A novel design for ultra-low power pulse-triggered D-Flip-Flop with optimized leakage power. Integr. VLSI J. **60**(1), 160–166 (2018)
2. Rashidi, H., Rezai, A., Soltani, S.: High-performance multiplexer architecture for quantum-dot cellular automata. J. Comput. Electron. **15**(3), 968–981 (2016)
3. Karimi, A., Rezai, A.: A design methodology to optimize the device performance in CNTFET. ECS J. Solid State Sci. Technol. **6**(8), 97–102 (2017)
4. Colinge, J.P.: Multiple-gate SOI MOSFETs. Solid State Electron. **48**(6), 897–905 (2004)
5. Amirsoleimani, A., Ahmadi, M., Ahmadi, A.: Logic design on mirrored memristive crossbars. IEEE Trans. Circuits Syst. II Express Briefs (2017). https://doi.org/10.1109/TCSII.2017.2729499
6. Nguyen, V.H., Sohn, K.Y., Song, H.: On-printed circuit board emulator with controllability of pinched hysteresis loop for nanoscale TiO_2 thin-film memristor device. J. Comput. Electron. **15**(3), 993–1002 (2016)
7. Kvatinsky, S., Belousov, D., Liman, S., Satat, G., Wald, N., Friedman, E.G., Kolodny, A., Weiser, U.C.: MAGIC-memristor aided logic. IEEE Trans. Circuits Syst. II Express Briefs **61**(11), 895–899 (2014)
8. Strukov, D.B., Stewart, D.R., Borghetti, J., Li, X., Pickett, M., Medeiros-Ribeiro, G., Robinett, W., Snider, G., Strachan, J.P., Wu, W., Xia, Q., JoshuaYang, J., Williams, R.S.: Hybrid CMOS/memristor circuits. In: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'10), pp. 1967–1970 (2010)
9. Kvatinsky, S., Wald, N., Satat, G., Friedman, E.G., Kolodny, A., Weiser, U.C.: Memristor-based material implication (IMPLY) logic: design principles and methodologies. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **22**(10), 2054–2066 (2013)
10. Ho, Y., Huang, G., Li, P.: Dynamical properties and design analysis for nonvolatile memristor memories. IEEE Trans. Circuits Syst. I Regul. Pap. **58**(4), 724–736 (2011)
11. Shang, Y., Fei, W., Yu, H.: Analysis and modeling of internal state variables for dynamic effects of nonvolatile memory devices. IEEE Trans. Circuits Syst. I Regul. Pap. **59**(9), 1906–1918 (2012)
12. Vijay, H.M., Ramakrishnan, V.N.: Radiation effects on memristor-based non-volatile SRAM cells. J. Comput. Electron. (2017). https://doi.org/10.1007/s10825-017-1080-x
13. Liu, K.C., Tzeng, W.H., Chang, K.M., Chan, Y.C., Kuo, C.C., Cheng, C.W.: The resistive switching characteristics of a Ti/Gd2O3/Pt RRAM device. Microelectron. Reliab. **50**(5), 670–673 (2010)
14. Borghetti, J., Snider, G.S., Kuekes, P.J., Yang, J.J., Stewart, D.R., Williams, R.S.: Memristives witches enable stateful logic operations via material implication. Nature **464**, 873–876 (2010)
15. Shaltoot, A.H., Madian, A.H.: Memristor based carry look ahead adder architectures. In: Proceedings of the IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'12), pp. 298–301 (2012)
16. Lehtonen, E., Laiho, M.: Stateful implication logic with memristors. In: Proceedings of the 2009 IEEE/ACM International Symposium on Nanoscale Architectures, pp. 33–36. IEEE Computer Society (2009)
17. Wang, X., Deng, H., Feng, W., Yang, u., Chen, K.: Memristor-based XOR gate for full adder. In: 35th Chinese Control Conference (CCC), pp. 5847–5851 (2016)
18. Guckert, L., Swartzlander, E.E., Jun, Y.: Optimized memristor-based multipliers. IEEE Trans. Circuits Syst. I Regul. Pap. **64**(2), 373–385 (2017)
19. Revanna, N., Swartzlander, E.E.: Memristor based high fan-out logic gates. In: IEEE Dallas Circuits and Systems Conference (DCAS) (Oct. 2016). https://doi.org/10.1109/DCAS.2016.7791136
20. Guckert, L., Swartzlander, E.E.: Optimized memristor-based ripple carry adders. In: 50th Asilomar Conference on Signals, Systems and Computers (2016). https://doi.org/10.1109/ACSSC.2016.7869644
21. Wang, X., Yang, R., Chen, Q., Zeng, Zh.: An improved memristor-CMOS XOR logic gate and a novel full adder. In: Ninth International Conference on Advanced Computational Intelligence (ICACI). IEEE (Feb. 2017). https://doi.org/10.1109/ICACI.2017.7974477
22. Rohani, S.G., TaheriNejad, N.: An improved algorithm for IMPLY logic based memristive full-adder. In: IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), (2017). https://doi.org/10.1109/CCECE.2017.7946813
23. Teimoory, M., Amirsoleimani, A., Shamsi, J., Ahmadi, A., Alirezaee, S., Ahmadi, M.: Optimized implementation of memristor-based full adder by material implication logic. In: 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 562–565 (2014)
24. Teimoory, M., Amirsoleimani, A., Ahmadi A., Alirezaee, S., Salimpour, S., Ahmadi, M.: Memristor-based linear feedback shift register based on material implication logic. In: Circuit Theory and Design (ECCTD) European Conference on IEEE, pp. 1–4 (2015)
25. Bickerstaff, K., Swartzlander, E.E.: Memristor-based arithmetic. In: Proceedings of the IEEE Conference on Asilomar Signals, Systems and Computers (ASILO-MAR'10), pp. 1173–1177 (2010)
26. Shaltoot, A.H., Madian, A.H.: Memristor-based modified recoded-multiplicand systolic serial-parallel multiplier. In: Proceedings of the IEEE International Conference on Communications, Signal Processing, and their Applications (ICCSPA'13), pp. 1–5 (2013)
27. Shin, S., Kim, K., Kang, S.M.: Reconfigurable stateful NOR gate for large-scale logic-array integrations. IEEE Trans. Circuits Syst. II Express Briefs **58**(7), 442–446 (2011)
28. Shin, S., Kim, K., Kang, S.M.: Field programmable stateful logic array. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **30**(12), 1800–1813 (2011)
29. Shin, S., Kim, K., Kang, S.M.: Resistive computing: Memristors-enabled signal multiplication. IEEE Trans. Circuits Syst. I Regul. Pap. **60**(5), 1241–1249 (2013)
30. Owlia, H., Keshavarzi, P., Rezai, A.: A novel digital logic implementation approach on nanocrossbar arrays using memristor-based multiplexers. Microelectron. J. **45**, 597–603 (2014)

31. Backus, J.: Can programming beliberated from the von Neumann style? A functional style and its algebra of programs. Commun. ACM **21**(8), 613–641 (1978)
32. Strukov, D.B., Likharev, K.K.: Reconfigurable nano-crossbar architectures. In: Waser, R. (ed.) Nanoelectronics and Information Technology, pp. 435–454. Wiley, Weinheim (2012)
33. Chen, Q., Wang, X., Wan, H., Yang, R.: A logic circuit design for perfecting memristor-based material implication. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **36**(2), 279–284 (2017)
34. Wang, X., Wu, Q., Chen, Q., Zeng, Z.: A novel design for Memristor-based multiplexer via NOT-material implication. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. (2017). https://doi.org/10.1109/TCAD.2017.2753204
35. Torrezan, A.C., Strachan, J.P., Medeiros-Ribeiro, G., Williams, R.S.: Sub-nanosecond switching of a tantalum oxide memristor. Nanotechnology **22**(48), 485203 (2011)