IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 April 2024, accepted 10 July 2024, date of publication 19 July 2024, date of current version 6 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3431206

RESEARCH ARTICLE

Cascade and Extensible In-Memory Arithmetic Computing in 2T1R ReRAM Arrays Using Time-Sum-Logic Design

WEI ZHU[1,2,3], YI-XING HE[1,2,3], HAO-NAN LI[1,2,3], XIAN-QIN LIU[1,2,3], SIWEN ZHANG[1,2,3], LEI WANG[1,2,3], JIANG ZHU[1,2,3], YUE-QI WANG[1,2,3], JINCHENG ZHANG[1,2,3,4], (Member, IEEE), YUE HAO[1,2,3,4], (Senior Member, IEEE), AND HAIJIAO HARSAN MA[1,2,3,4]

[1]Low Dimensional Quantum Physics and Device Group, School of Microelectronics, Xidian University, Xi'an 710071, China
[2]State Key Discipline Laboratory of Wide Band Gap Semiconductor Technology, School of Microelectronics, Xidian University, Xi'an 710071, China
[3]Collaborative Innovation Center of Quantum Information of Shaanxi Province, Xidian University, Xi'an 710071, China
[4]State Key Laboratory of Wide Bandgap Semiconductor Devices and Integrated Technology, Xidian University, Xi'an 710071, China

Corresponding author: Haijiao Harsan Ma (mahj07@xidian.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2022YFB3605600 and Grant 2021YFA0715600, by the Fundamental Research Funds for the Central Universities under Project QTZX23078, and by the Interdisciplinary Cultivation Program of Xidian University.

ABSTRACT Many designs of in-memory logic computing using ReRAM have been proposed and even a few of them have been experimentally demonstrated. However, the ability to cascade and extend the designed circuits is limited. In this work, we elaborate a design called time-sum-logic based on 2T1R memristor arrays and implement arithmetic computing circuits using memristors by integrating both time and space dimension. The core method of our design is to use sum of minterms to transfer logic to time sequence by 2T1R memristor arrays. This method is extensible by extending the number of layers to realize multi-bit arithmetic computing. Remarkably, cascading between different layers is elaborated carefully and pipelines are adopted to manage the sequence of memristors' writing and reading and lower the complexity and latency. In this way, the proposed circuits will be more efficient and can support high data throughout. Based on our design, 4-bit full adder and 4×4 multiplier are designed with pipelines, and n-bit full adders and m×n multipliers could be easily constructed. The low latency and high efficiency confirm the advantages we have illustrated and show a promised application prospect.

INDEX TERMS Memristor, logic design, arithmetic computing, crossbar.

I. INTRODUCTION

The complexity of integrated circuits, especially digital integrated circuits, increases at an exponential rate according to Moore's law [1]. However, this rate slows down [2] because of various limiting factors such as difficulties in reducing the size of silicon-based transistors into subnanometer and leakage induced by quantum effects at subnanometer scales. Therefore, new materials and devices are invented to conquer these difficulties. Among them, nonvolatile memristors and in-memory computing circuits based on memristors are very promising candidates. Memristors were first proposed as the fourth basic element of electronic circuits by L. Chua in 1971 [3]. About forty years later, HP labs announced the first report of memristors using titanium oxide nanowires [4], which stimulated the search and development of memristors and in-memory computing circuits.

Memristors are two terminal resistive devices with switching dynamics and the ability to remember their states of resistivity [5]. Abundant works on analog computation using memristors have been proposed and/or demonstrated [6], [7], [19]. Meanwhile, there are some previous works on logic computation using memristors. For example, current existing design topologies such as Material Implication Memristor

The associate editor coordinating the review of this manuscript and approving it for publication was Artur Antonyan.

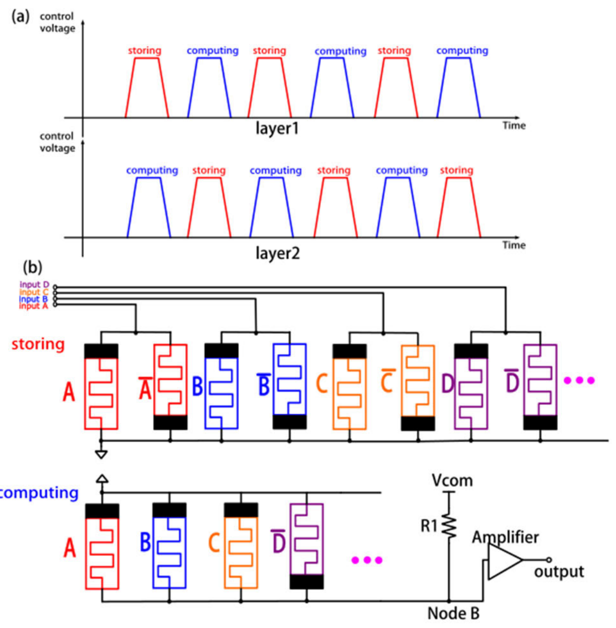


FIGURE 1. **Basic structure of Time-Sum-Logic:(a) The complementary control signals of different layers. (b)Topology of different operations.**

Logic (IMPLY) [8], Memristor Ratioed Logic (MRL) [9], Memristor-Aided Logic (MAGIC) [10], CMOS/Memristor Threshold Logic [11], CMOS-like Memristor Complementary Logic [12], Parallel Input-Processing Memristor Logic [13], ratioed logic in ReRAM [14], [15], [30] have been proposed. IMPLY adopts material implication to replace Boolean logic because of the nonvolatile property of memristors. Whereas, MRL makes use of the linear resistive property of memristors to perform logic functions. MAGIC, however, stores results in memristors, which is different from other designs. CMOS/Memristor Threshold Logic also makes use of resistivity but from a different angle which is more like analog computation. Inspired by CMOS technology and Boolean logic, CMOS-like Memristor Complementary Logic replaces MOSFET with memristors in traditional digital circuits. Parallel Input-Processing Memristor Logic makes use of the conduction of memristors to compute in parallel. We notice that redundant transistors and memristors can be used in these strategies, which may be difficult when integrating them into large-scale circuits.

Recently, several works [14], [15], [16], [17] have been reported to perform logic operation in ReRAM with external circuits. The basic operations of these works are based on NOR gate, and flip-flops work as buffers for the purpose of cascading. Several considerations like variability and endurance of memristors are also discussed in those works. We find that the basic logic operation types of the work above are limited and several logic operations need multistep to be achieved, which can be a limiting factor when integrating large-scale logic circuits to memristor arrays. By the way, each operation needs a flip-flop to hold the output, which can consume extra components.

In this paper, we propose a novel approach named 'Time-Sum-Logic' to do logic computation using memristors which integrates both the time and space dimensions of circuits. The time dimension is that, unlike the traditional CMOS circuits or some previous work, in our design, the logic outputs are indicated by the control voltage sequence of the accessing transistors. By changing the sequence of control voltage, different logic operations can be implemented with the same data, which overcomes the limiting factor of works above and is beneficial for data-reusing. Pipelines can also be adopted in the design process, which increases the efficiency and decreases the complexity of circuits and sequence significantly. Considering the background of big data in the modern information technology, data-reusing and pipelines in the very underlying levels of circuits can promote the speed and the power factors of modern processors. More details of time dimension are illustrated carefully in section II. Moving to the space dimension, we illustrate how to map the proposed sequence to a real memristor array. The basic unit structure we use is a combination of two transistors and one memristor (2T1R) which differs from traditional 1T1R structure [14], [15]. 2T1R structure adds another output by adding one more transistor comparing to 1T1R structure. As we will show, this delicate change will increase the efficiency of logic operations and will make full use of the memristors. In the last part of the paper, we design a 4-bit full adder. The comparisons with some prior works are given, too. The total number of electronic components are fewer by using our design compared with previous works [21], [22] with low latency under the sense of pipelines. Based on the adder, we further design a 4*4 multiplier, with the pipelines adopted. Comparing with previous works [25], we find that the components of our proposed multiplier can increase significantly when data bits go up, but the delay will remain the same. The ignorable latency and less complexity under the sense of pipelines demonstrate the huge application prospect towards building modern processors.

II. THE DESIGN PRINCIPLE: TIME-SUM-LOGIC

A. GENERAL IMPLEMENTATION OF TIME-SUM-LOGIC

As the first application circuits of memristors, the stateful logic is partially operating in the time dimension [18], which indicates that the logic operation using memristors may at least partially be organized in the time dimension. In our design, especially cascading between different layers, is operated complementary in the time dimension. In this section, we will discuss different operations over time while the space dimension will be elaborated in the next section.

As Fig. 1(a) shows, in our design, the ideal bitcell is two complementary memristors. Multiple bitcells of memristors can be grouped as a so-called layer. The basic operations of each layer are storing and computing. Cascaded layers share the complementary period of operations. Taking two layers for instance, in period i, the storing operation of layer 1 is carried out, which is meanwhile the computing period of layer 2. While in period ii, the logic output of layer 1 is computed, and transported to the cascaded memristor of layer 2 as layer 2 is in the storing period. It is worth noting that the storing and computing operations of two cascaded layers 'rotate', meaning that the computing period of the last layer is almost the storing period of the next layer and vice versa. Note that delay between different layers accumulates only if these layers work at the same time. However, in our work, cascaded layers share complementary working time (computing period). Owing to this, computing delay in different layers will not add up, which is of much benefit to large-scale circuits. Moreover, this also enables pipelines to be implemented between different layers, which will be illustrated in the next subsection. Next, topologies of different operations in the same layer will be discussed.

As shown in Fig. 1(b), when a layer is in its storing period, the logic input is stored in two contiguous memristors instead of just one memristor, which are 'complementary'. This a difference from prior work [8], [9], [10], [14], [15], [16], which may seem to 'waste' some memristors. But this is actually a trade-off, meaning that we use more memristors to store the same data, but both the input data and its inversion are stored. As we will see later, such storing operation can help generate minterms more conveniently in the computing period. Another notable point is that in our design, logic '1' is stored as high resistance (HRS), whereas logic '0' is stored as low resistance (LRS), which is another difference from some previous work [10], [14], [15], [16]. Considering the nonvolatile and polarity property of memristors, the memristor with the top electrode connecting to the ground stores the logic input itself, whereas the complementary memristor stores its inversion, respectively.

When entering the computing period, the whole structure is 'inverted' for the purpose of parallelism as can be seen from Fig. 1(b). The method to achieve this topology transformation will be illustrated in the next section. After that, a relatively small computing voltage called Vcom will be applied to this structure through a peripheral circuit. The voltage should be chosen carefully here because the state of memristors should not be changed in the computing period. Meanwhile, the resistance R1 chosen here must be much lower than Roff (the higher impedance of the memristor) but much higher than Ron (the lower impedance of the memristor). Only with the proper voltage and resistor chosen here, can the Time-Sum-Logic be performed. To show the details of computing, here we take the three-inputs logic operation for example. In the computing period, three memristors or less are connected to Node B. It is obvious that only when the three memristors are all in the state of high resistance, can the memristor array has high resistance: Roff/3, which is much higher than R1. The output voltage B, or the 'precursor' of the output of this layer, is given by the formula below in this situation.

$$\mathrm{V}_B = \frac{\frac{V_{com}R_{off}}{3}}{\frac{R_{off}}{3} + R_1} \approx V_{com} \tag{1}$$

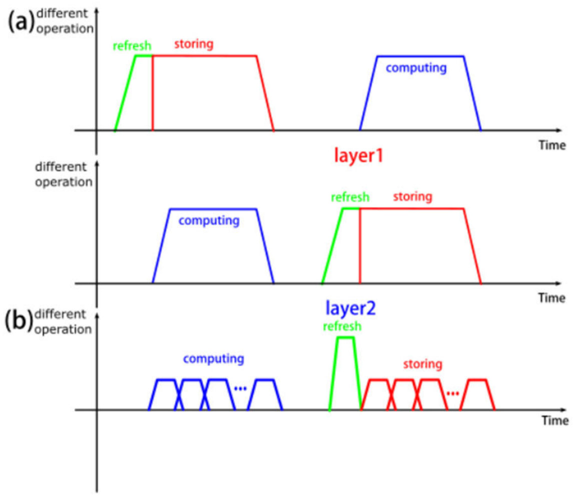


FIGURE 2. Cascading process of Time-Sum-Logic: (a)details of different periods of two cascaded layers. The y axis represents the different operations (b) Different Subperiods of Cascading operation of layer 2.

On the contrary, however, if one of (or some of) the memristors is in the state of low resistance, the 'precursor' of the output will be given by the formula below.

$$V_B \approx \frac{V_{com}R_{on}}{R_{on} + R_1} \approx 0 \tag{2}$$

It is obvious that the operation here is actually a voltage divider, and several works have taken the similar idea to implement logic previously [14], [15], [16]. However, except we invert the way of connecting the memristor to the divider, the main difference here is that we enable the dynamic connection to voltage divider by changing the control voltage. As we will explain below, this change enables the general way of expressing logic function, i.e. sum of minterms to be implemented to the sequence, as well as the cascading between different layers. We believe this is a novel way of designing logic circuits based on memristor arrays.

The output of the voltage divider, or the 'precursor' of the output voltage will be then amplified to be enabled to change the state of the cascaded memristor of the next layer as an input if it is nonzero. In the next period, the cascaded memristor will participate in the computing period of the next layer as an input.

As we mentioned above, HRS corresponds to logic '1', whereas LRS corresponds to logic '0'. According to the discussion above, only when the logic variables of the memristors are all '1', can the output be high voltage, which means that we have generated the logic AND of three inputs. Compared with the operation NOR used in some relative works, like [14], this AND operation can be seen as a complementary function, because in these works, HRS represents logic '0'. Recall that we use two complementary memristors to store the same data, meaning that both the data and its inversion are stored. Thus, we can generate any minterm (i.e. logic AND) of these inputs by changing the memristors connected

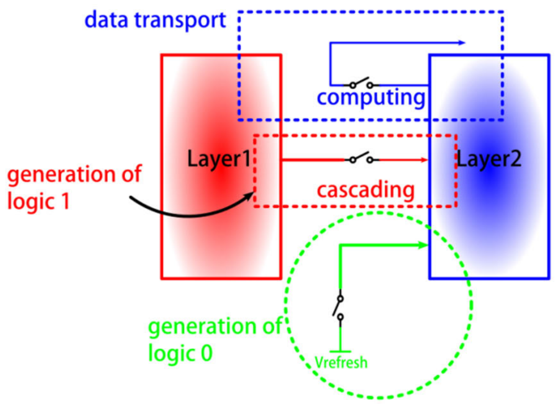


FIGURE 3. Summary of Time-Sum-Logic. The cascading period corresponds to the computing period of layer 1. Control signals: storing, refresh and computing of layer1 and 2 are generated by external circuits.

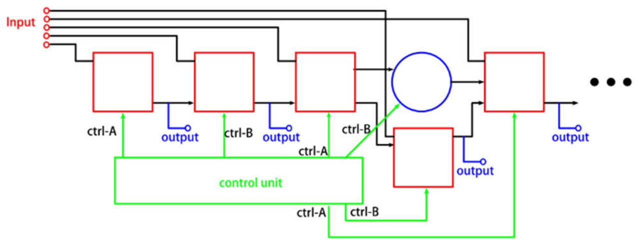


FIGURE 4. Basic structure of pipelines of memristor arrays. The square shown in the figure is one set (layer) of memristors. The circle represents the buffer. The whole structure is under the control of two complementary signals {ctrl-A, ctrl-B} which are generated by the control unit.

to the external circuit. For instance, if we want to generate $\bar{A}$BC under the condition of three input logic circuits, we can choose memristors of variables $\bar{A}$, B, and C to be connected to the voltage divider. But in most cases, the logic output is the sum of minterms. In traditional CMOS circuits, this operation can be realized by adding a multiple input OR gate. This will introduce delay because the delay will accumulate as the cascaded layers become more. To overcome this weakness, the operation of sum, or logic 'OR' in our design, however, is executed in the time dimension (Time-Sum-Logic). In brief, we connect different memristors in different subperiods of computing period dynamically, and the output in different subperiods can be summed up by the same memristor of the next layer owing to the non-volatile property of memristor. The details are shown in Fig. 2. Taking two cascaded layers for instance, the storing period of the layer 2 can be divided into two sub-periods, in the refresh subperiod, a negative voltage is applied to the cascaded memristor, which will then have the low resistance (with its reversion high). This operation generates logic '0' of the logic output. In the subperiod of 'storing', which corresponds to the computing period of the last layer1, the combination of computed memristors in layer1 is changed on the basis of the logic expression in the form of minterms. If one of (or some of) the minterms is true, the high voltage generated by the last layer1, will be applied to the cascaded memristor and it will be changed to high impedance (with its reversion low). If none of the minterms is true, the cascaded memristor will remain low impedance (logic '0') because of its nonvolatile. In this process, we separate the generation of logic '0' and logic '1' in the time dimension. Here is an example of the whole process. Assuming that the logic expression is ABC + $\bar{A}\bar{B}$C, memristors of logic variables 'A', 'B', and 'C' should be computed first, while memristors of logic variables $\bar{A}$, '$\bar{B}$', and 'C' will be computed then. The order of computation usually should be considered carefully, and the reason will be illustrated in the next section. After all the minterms (here are ABC and $\bar{A}\bar{B}$C) have been computed, the state of the cascaded memristor will store the result owing to its nonvolatile and participate in the computation of next layer. As a general way of representing logic function, any combinational circuits can be expressed in the form of sum of minterms, meaning that any logic expression can be realized by Time-Sum-Logic without any transformation. Because of the above reason, compared with prior works [8], [14], our work is more flexible and extensible.

In Fig. 3, we summarize our design briefly. Again, the core insight here is that we separate the generation of logic '0' and logic '1' in time dimension by dynamic connections between memristor arrays and voltage dividers. Logic '0' is generated first. Sum of minterms is organized in different subperiods of the following computing period and the logic state of the cascaded memristor can be overwritten to logic '1' if and only if the logic output is true. In the next subsection, pipelines will be introduced to the design flow, with the core conception and insight illustrated here.

B. PIPELINES IN TIME-SUM-LOGIC

In this subsection, we will discuss how to apply pipelines in our design methodology and show the advantages of such design consideration. In the beginning, we want to point out that pipeline technology is the core technology to improve the efficiency in our design, and a natural application of the nonvolatile property of memristor. Prior works [8], [9], [10], [14], [15] consider little about this technology, which may result in the waste of the excellent characteristics of memristors and decrease the efficiency of computing. Another advantage of pipelines is that the complexity of both circuits and sequence will decrease because the complex work will be decomposed into several simple tasks. Moreover, delay through these pipelined tasks will not accumulate as illustrated in the last subsection, so pipeline can be seen as an effective way to eliminate latency in large scale circuits.

The basic structure of pipelining in Time-Sum-Logic is shown in Fig. 4. Every memristor in our design can be both seen as a computing processor and a pipeline register. The resistive property of memristor corresponds to the ability of computing and the nonvolatile property corresponds to the pipeline register. In the last subsection, we have discussed the different periods of operation of the same layer and cascading between two layers. With more layers being added to the general design flow, one can easily find that there are only two sets of control signals on the whole because the cascaded layers share two complementary operation periods, which can be marked as ctrl-A and ctrl-B. Owing to this, the complexity of generating gate voltage signals of transistors(switches) will reduce greatly. Moreover, as we have mentioned in the last subsection, the storing stage and computing stage of two continuous layers rotate, which means that when layer1 is in the computing period, layer2 is in the storing period, and layer 3 is in the computing layer just as layer1. Extending this conclusion, one can easily find that the Odd layers share the same period of computing and storing, and even layers share the complementary period as well. Thus, after one whole period of storing and computing, there will be N outputs of different inputs, here N is the number of memristor layers. These N outputs are parallel under the sense of pipelines and can be used in other external circuits if they share the same set of control signals {ctrl-A, ctrl-B}. However, it can be seen that the core problem in our memristor pipelines is the sequence match. Data inputs of the same output must be generated at the same time, meaning that not only the same set of control signal should be adopted to these input memristor layers, but the pipeline delay of different input layers should keep same. To realize sequence match, buffers are introduced to our structures. Here, we use one memristor as a buffer, it stores whatever signals transported from the last layer, and transport the output unconditionally in its computing period. With buffers used, we can change the set of control signals and the pipeline delay of the intermediate data, which enables us to control the pipeline manually. To build a deeper and more practical understanding of this subsection, case study, namely a multiplier is illustrated in section IV.

III. PROCESS OF TIME-SUM-LOGIC

Recently, many memristor crossbar circuits have been built [6], [7], [27], [28], among all these structures, 1T1R configuration is a very promising structure that enables external circuits to access each memristor independently. In this section, to realize Time-Sum-Logic, a novel configuration called 2T1R is built. Comparing with the existing 1T1R structure, 2T1R adds one more transistor above one of the electrodes without any other changes, which enables our logic circuit compatible with the existing crossbars. To implement arithmetic computing on 2T1R parallelly, we design a crossbar based on 2T1R, as shown in Fig. 5(a). As can be seen from this subfigure, we bind two dual 2T1R (the electrodes added transistors are different) together as a bitcell. Two transistors from different memristors are connected with one wire and the remained two are connected by another. The third wire connects two electrodes without transistors. Compared with the traditional crossbar bitcell. two memristors are used to store the data and its inversion, which is redundant in some ways. To make full use of memristors, we add another wire for each bitcell to add one more output. After combining these bitcells, the crossbar can be utilized to realize the

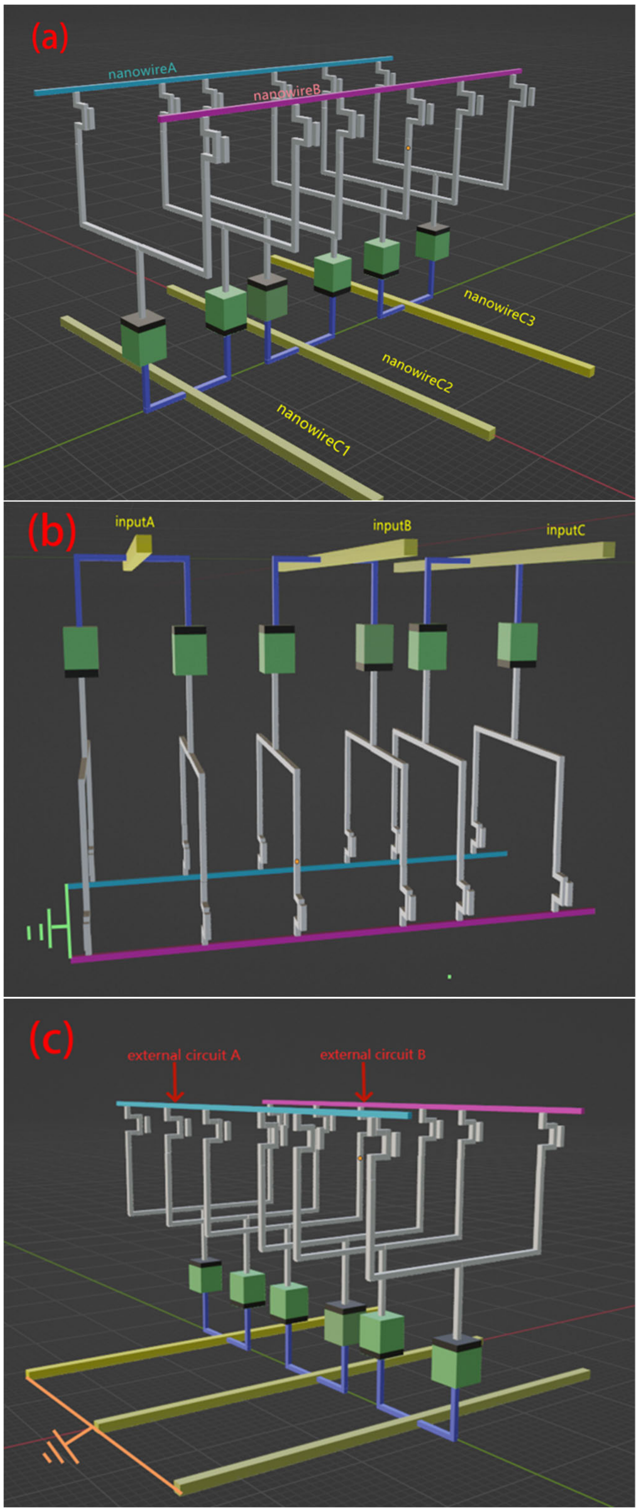


FIGURE 5. Proposed 2T1R crossbar structure: (a) Fundamental 3D model (b) Topology of storing period (c) Topology of computing period.

proposed logic operation, here are the working details: In period i (storing) we discussed previously, wire A and wire B will be connected to the ground. wires Ci will be connected to corresponding logic inputs, including the refresh signal,

TABLE 1. The truth table of one-bit full adder.

A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

which is illustrated in Fig. 5(b). Under this topology, input signals can both change the state of two dual memristors and different inputs can be applied to different dual memristor pairs parallelly, which is important to reduce delay when there exist multiple inputs. In period ii (computing), however, wires Ci are connected to the ground, while wire A and B will be connected to different external circuits respectively to compute different logic functions, which is illustrated by Fig. 5(c). Under this topology, wire A and B can access a certain memristor and its dual individually, which bring the possibility of increasing efficiency. It is worth noticing that in our design, the logic function is implemented by Sum of minterms. Although the order of minterms in the logic function doesn't count, when we want to compute two logical functions parallelly, the relative order of minterms in two logic functions makes sense because the order of minterms represents the computing order of different memristor combinations. If the same memristor is computed by two wires at the same time, the output will be the same. For instance, if wire A is processing 'ABC', while wire B is processing 'A$\bar{\text{B}}\bar{\text{C}}$'. It is obvious that these wires are connected to each other through memristor A, which will make outputs confused. Thus, the computing sequence of two different wires must be complementary, meaning that if wire A is generating minterm ABC, wire B can only generate its bitwise NOT minterm, which is $\bar{\text{A}}\bar{\text{B}}\bar{\text{C}}$. If the logic function associated with wire B doesn't contain the bitwise NOT minterm of wire A, wire B should be connected to the ground.

Taking one-bit full adder for example, the outputs are given by the formulas below. According to the truth table given by Table 1, one possible computing sequence is illustrated in Fig.6.

$$S_i = A_i\bar{B}_i\bar{C}_i + \bar{A}_iB_i\bar{C}_i + A_iB_iC_i + \bar{A}_i\bar{B}_iC_i \tag{3}$$

$$C_{i+1} = A_iB_i + A_i\bar{B}_iC_i + \bar{A}_iB_iC_i \tag{4}$$

As illustrated in Fig. 6, computing periods of each layer are divided into 4 subperiods. In each subperiod, different transistors are turned on, and different topologies are implemented as shown in the figure. As what we expect, two parallel outputs Si and Ci+1 are generated through the 2T1R

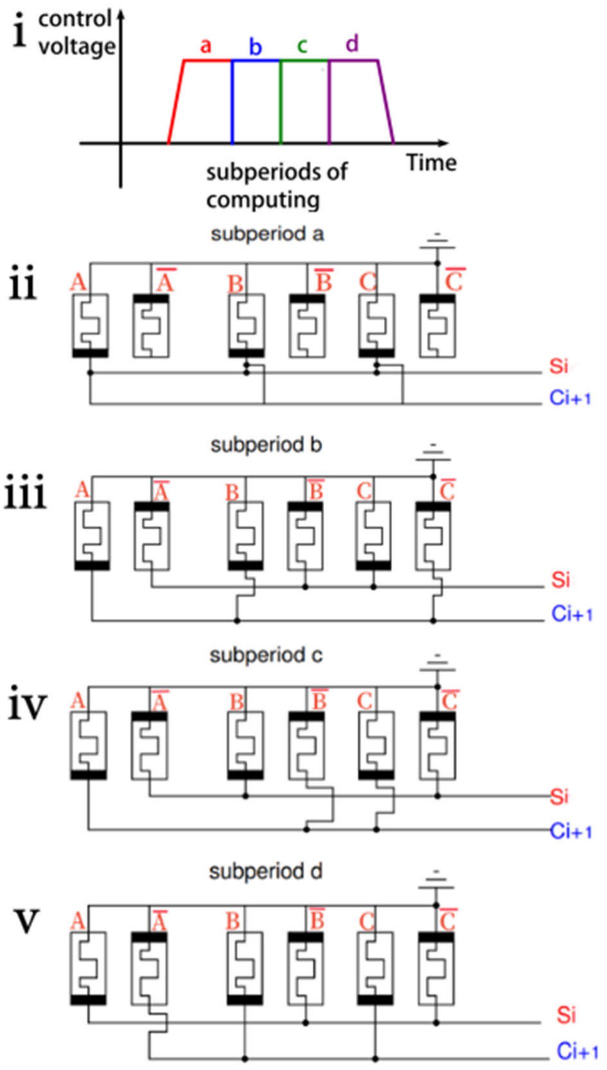


FIGURE 6. One possible computing sequence of one-bit full adder: (i) Subperiods of computing (subperiods of operation 'or') (ii) Topology of subperiod a (iii) Topology of subperiod b (iv) Topology of subperiod c (v) Topology of subperiod d.

memristor array at the same time, which increase the efficiency of computing greatly.

The core insight of this 2T1R crossbar is that we enable topology change in different periods and the ability of accessing every memristor in different subperiods in the period of computing. However, this relies on the accurate control by the peripheral circuits. These circuits may cost a lot when the number of input increases and when the logic expression contains too many minterms, which may be a limiting factor in the practical circuits. Further works may be explored to focus on this control circuit issue.

Besides, compared with traditional MPU, a very important insight into Time-Sum-Logic is that the storing and computing procedures are carried out in the same place, with logic functions processed in different periods. The storage and

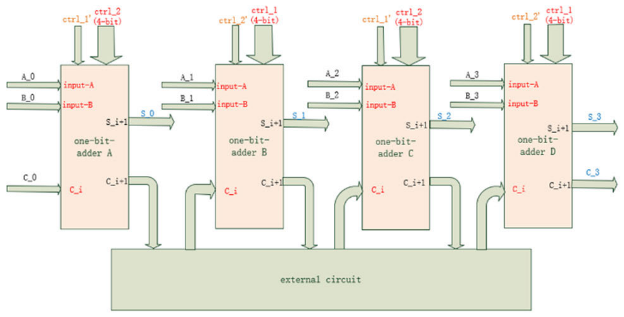


FIGURE 7. Proposed 4-bit full adder design implemented by Time-Sum-Logic: Diagram of 4-bit full adder built by four one-bit adders.

process units of data are separated in traditional MPU, which is one of causes of 'Memory Wall'. Furthermore, a specific process unit corresponds to a specific logic function, which cannot be changed once produced. On the contrary, Time-Sum-Logic is programmable, and the same data can be reused for different logic functions without being moved, which is of great meaning to eliminate 'Memory Wall'.

IV. FUNDAMENTAL OPERATIONS OF ARITHMETIC: DESIGN AND SIMULATION

A. FOUR-BIT FULL ADDER

In this subsection, a 4-bit full adder is built in spice based on the previously illustrated principles. The memristor simulation model is used according to [20]. As shown in Fig. 7, a four-bit full adder is divided to four cascaded one-bit adders for the purpose of pipelines. A_0 to A_3 represents the first four-bit input of the adder and B_0 to B_3 represents the second. C_0 represents the carry input; S_0 to S_3 represents the output of the adder, and C_3 represents the carry output. The control signal is complementary between two close layers. Thus, the whole adder only needs two different sets of control signals which lowers the cost of generating control signals as we have discussed before. There exists prior work on the driving circuit [29] and the generation of control signals is beyond the scope of our discussion. Thus, to simulate and verify the function of the four-bit full adder, we use ideal voltage sources in spice to represent the control signals.

The simulation results are shown in Fig. 8. Here, we set A as '0010', B as '1101', and carry input as 1. The result shown in Fig. 8 is '10000', which is exactly the true result. It is worth pointing out again that the 4-bit full adder is actually decomposed into four 1-bit full adders under the sense of pipelines, and in after each period of storing and computing, there will be 5 'parallel' outputs of different input. As a result, the efficiency increases, the total latency decreases, and more important, the complexity of circuits and sequence are lower as we expect. Moreover, the method can be extended to any n-bit adder by adding more one-bit adders in pipelines with all the advantages above.

In Table 2, based on the results shown in [21] and [22], we compare the cost of transistors and memristors in this

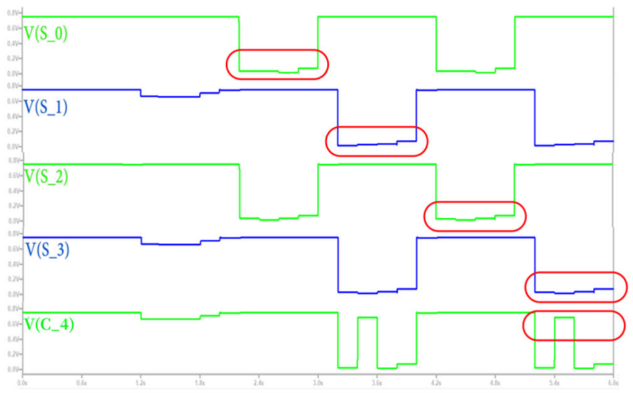


FIGURE 8. Outputs of the proposed 4-bit adder: The complementary color (i.e. green and blue) represents the pipelined outputs, which are generated in different periods as circled. The control signals are generated by external circuits.

TABLE 2. Comparison of different 4-bit CLA.

4-bit full CLA	Number of memristors	Number of transistors
MRL(improved in [21])	76	80
MRL [22]	108	96
CMOS [21]	-	248
IMP Logic [22]	88	Not specified
Time-Sum-Logic	24	48

work and previous 4-bit full carry-lookahead adders, considering the 'parallel' property of our work which has been explained above. The result shows fewer transistors and memristors are required in this work with all the advantages illustrated before. We notice that the peripheral control circuits are ignored in the comparison similar as previous works [8], [9], [10], [11], [12], [25]. However, as we pointed out in section III, the peripheral circuits may be big and cost lots of transistors, so in a practical situation, maybe only fewer memristors can be achieved. In summary, this 4-bit full adder works parallelly and efficiently with little ignorable delay and costs fewer memristor components, which is of big application potential.

B. 4 × 4 MULTIPLIER

Multiplication is a core operation in many areas such as signal processing, 3D graphics, and image processing [23]. Multiplies consume significant power and area, which is a tremendous problem in VLSI [24]. Several works have been proposed to design multipliers based on IMPLY [25], MAD [25], and MRL [26]. Although all these works have been optimized, there exists some space for improvement. For example, when the multiplier is designed by IMPLY or MAD, latency which is proportional to the number of input bits exists even when adopting pipelines to reduce [25]. This can be a serious issue when integrating them into large-scale circuits. Furthermore, MRL relies on linear memristors which is slower than threshold-type memristors, and the applied

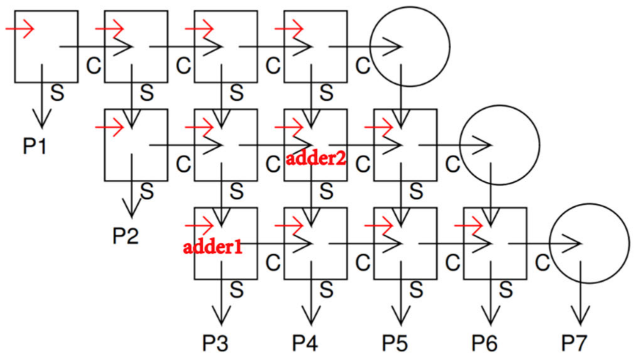


FIGURE 9. Workflow of 4*4 multiplier, the squares in the figure represent one-bit adders and the circles represent buffers. The black arrows show the data flow through different one-bit full adders and indicate latency. The red arrows represent the input data.

voltage affects the performance of circuits greatly. To optimize multipliers designed by memristors, in this subsection, we will illustrate a novel way of designing an M*N Braun multiplier based on Time-Sum-Logic.

In general, we design Braun multipliers by adders. The basic workflow is shown in Fig. 9. Pipelines are used in our design, so the proposed multiplier achieves high throughput performance and multidigit parallel output. The squares in the figure represent one-bit adders and the circles represent buffers. The black arrows show the data flow through different one-bit full adders and indicate delay in the pipeline. The red arrows represent the input data. Note that the input data used in this multiplier is the logic and of different bits of the multiplicand and the multiplier, which can be generated previously. Moreover, the data should be inputted delicately in different periods and meet with the sequence of the data flowing through adders.

Taking 4×4 multiplier for example, generally, the output will have eight bits. Assuming the inputs are A3A2A1A0 and B3B2B1B0, the output is P7P6P5P4P3P2P1P0. P0 is the logic and of A0 and B0, and other bits of the output can be computed as the figure shows. For instance, if we want to compute P4, three inputs should be given as shown in Fig 9. They are the carry output of adder 1, the sum output of adder 2, and one external input. The carry output of adder 1 and the sum output of adder 2 are computed at the same time as can be seen in Fig 9, and if the input given by external circuits is A1 and B3 at this time, P4 will be computed correctly. It is worth pointing out again that 4*4 multiplier will generate 8 bits of data in one period parallelly, but these data correspond to different inputs because the proposed multiplier is based on pipelines. Moreover, owing to the extensibility and flexibility of Time-Sum-Logic any m×n multiplier can be constructed by adding the number of adders in the line in Fig. 9 to n and changing the number of lines to m-1 in pipelines.

The 4×4 Braun multiplier simulation results are shown in Fig. 10. To simplify the simulation, we use voltage sources to replace the input data. The inputs are set as 1011 and 1111,

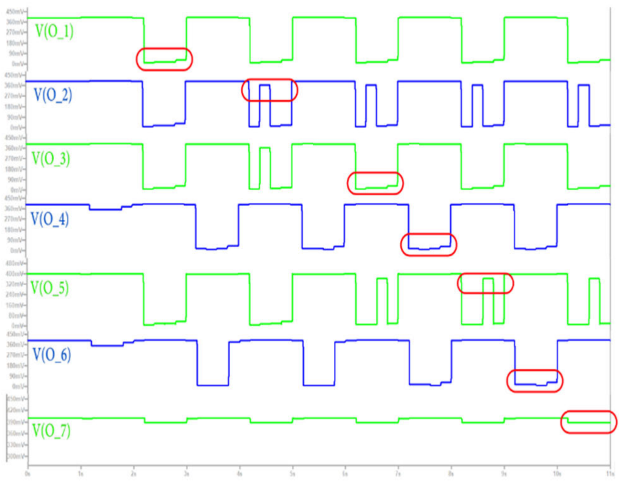


FIGURE 10. Simulation results of the proposed 4*4 multiplier. The complementary color (i.e. green and blue) represents the pipelined outputs, which are generated in different periods as circled. The control signals are generated by external circuits.

TABLE 3. Delay and area comparison of different multipliers.

	CMOS (shift-and-add type)[25]	IMPLY (proposed in [25])	MAD (proposed in [25])	Time-sum-logic
Delay (pipelined steps)	$2N^2 + 23N$	24N	4N	(15N-25+5R)/R
Area	132N + 6 MOSFETs	7N + 1 memristors 7N drivers 8N – 1 switches	5N memristors 3N + 2 drivers 14N switches	$6N^2 + 5N - 7$ memristors $12N^2 + 10N - 14$ MOSFETs

and the different bits of output are generated at different points in time, the final higher 7 bits of result are 1010010 as marked in the figure.

An advantage of pipelining is that it can promote the efficiency greatly. Taking the N×N multiplier for instance, it needs 15N-20 steps (we take single subperiod in the computing period as one step to show the advantage) to get the first full output. However, it only needs five steps to get each remaining output. Assuming that there are R outputs needed to be computed, the average delay is (15N-25+5R)/R. As R goes bigger, this delay will eventually approach 5. Compared with the prior works [25] as shown in Table 3, if R is bigger enough and if N is greater than 1 (which is reasonable in practice), the latency of our proposed multiplier is the least. In summary, our proposed multipliers use more components, but can compute with ignorable fixed latency under the sense of pipelines, if the number of required outputs is big. This is because the adders in our design are either storing data or computing without idling owing to pipelines. The continuous operation of every memristor ensures the high efficiency of our design. The memristors used here are also threshold-type, which have less delay than the memristors in MRL [26]. And the delay accumulates between layers in MRL, which can be a problem when the scale of circuits becomes larger.

In summary, the proposed Braun multiplier is built by adders under the sense of pipelines. Due to the cascade of our circuits, the parallel outputs and the constant ignorable delay illustrate the high efficiency and the huge application prospects of Time-Sum-Logic.

V. CONCLUSION

In this work, a novel logic computation methodology (Time-Sum-Logic) using 2T1R memristor arrays is proposed. The basic principle, the idea of pipelines and data reusing are illustrated carefully to decrease the latency and complexity and increase the efficiency and flexibility. Moreover, classic combinational circuits like adder and multiplier are designed with considerations based on the methodology. The demonstration of 4-bit full adders shows the low cost, high efficiency and low latency of our design. Modifying this 4-bit full adder and 4×4 multiplier were designed, confirming the flexibility of our methodology. Due to the cascade ability n-bit full adders and $m \times n$ multipliers could be easily constructed. The ignorable delay, parallel output and high efficiency demonstrate the great potential to design higher performance processors with 2T1R ReRAM arrays in the very fundamental layer. With the general principle illustrated in our work, complex combinational logic circuits and sequential logic such as flip-flops can be designed, which is of remarkable application prospects.

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Proc. IEEE*, vol. 86, no. 1, pp. 82–85, Jan. 1998.

[2] T. N. Theis and H.-S. P. Wong, "The end of Moore's law: A new beginning for information technology," *Comput. Sci. Eng.*, vol. 19, no. 2, pp. 41–50, Mar. 2017.

[3] L. Chua, "Memristor—The missing circuit element," *IEEE Trans. Circuit Theory*, vol. CT-18, no. 5, pp. 507–519, Sep. 1971.

[4] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[5] I. Vourkas and G. C. Sirakoulis, "Emerging memristor-based logic circuit design approaches: A review," *IEEE Circuits Syst. Mag.*, vol. 16, no. 3, pp. 15–30, 3rd Quart., 2016.

[6] Z. Wang et al., "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electron.*, vol. 1, no. 2, pp. 137–145, Feb. 2018, doi: 10.1038/s41928-018-0023-2.

[7] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020, doi: 10.1038/s41586-020-1942-4.

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

[9] S. Kvatinsky, N. Wald, G. Satat, A. Kolodny, U. C. Weiser, and E. G. Friedman, "MRL—Memristor ratioed logic," in *Proc. 13th Int. Workshop Cellular Nanosc. Netw. Appl.*, Aug. 2012, pp. 1–6.

[10] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[11] L. Gao, F. Alibart, and D. B. Strukov, "Programmable CMOS/memristor threshold logic," *IEEE Trans. Nanotechnol.*, vol. 12, no. 2, pp. 115–119, Mar. 2013.

[12] I. Vourkas and G. C. Sirakoulis, "Memristor-based combinational circuits: A design methodology for encoders/decoders," *Microelectron. J.*, vol. 45, no. 1, pp. 59–70, Jan. 2014.

[13] G. Papandroulidakis, I. Vourkas, N. Vasileiadis, and G. Ch. Sirakoulis, "Boolean logic operations and computing circuits based on memristors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 12, pp. 972–976, Dec. 2014.

[14] M. Escudero, I. Vourkas, A. Rubio, and F. Moll, "Memristive logic in crossbar memory arrays: Variability-aware design for higher reliability," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 635–646, 2019.

[15] C. Fernandez and I. Vourkas, "ReRAM-based ratioed combinational circuit design: A solution for in-memory computing," in *Proc. 9th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, Sep. 2020, pp. 1–5.

[16] C. Fernandez and I. Vourkas, "Reliability-aware ratioed logic operations for energy-efficient computational ReRAM," in *Proc. IFIP/IEEE 30th Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Oct. 2022, pp. 1–6.

[17] D. Bhattacharjee, L. Amaru, and A. Chattopadhyay, "Technology-aware logic synthesis for ReRAM based in-memory computing," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1435–1440.

[18] D. S. Jeong, K. M. Kim, S. Kim, B. J. Choi, and C. S. Hwang, "Memristors for energy-efficient new computing paradigms," *Adv. Electron. Mater.*, vol. 2, no. 9, Sep. 2016, Art. no. 1600090.

[19] X. Zhang, A. Huang, Q. Hu, Z. Xiao, and P. K. Chu, "Neuromorphic computing with memristor crossbar," *Phys. Status Solidi, A*, vol. 215, no. 13, Jul. 2018, Art. no. 1700875, doi: 10.1002/pssa.201700875.

[20] V. Mladenov, S. Kirilov, and I. Zaykov, "A general model for metal oxide-based memristors and application in filters," in *Proc. 11th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, Jun. 2022, pp. 1–4.

[21] G. Liu, L. Zheng, G. Wang, Y. Shen, and Y. Liang, "A carry lookahead adder based on hybrid CMOS-memristor logic circuit," *IEEE Access*, vol. 7, pp. 43691–43696, 2019.

[22] A. H. Shaltoot and A. H. Madian, "Memristor based carry lookahead adder architectures," in *Proc. IEEE 55th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2012, pp. 298–301.

[23] J.-Y. Kang and J.-L. Gaudiot, "A simple high-speed multiplier design," *IEEE Trans. Comput.*, vol. 55, no. 10, pp. 1253–1258, Oct. 2006, doi: 10.1109/TC.2006.156.

[24] J.-T. Yan and Z.-W. Chen, "Low-power multiplier design with row and column bypassing," in *Proc. IEEE Int. SOC Conf. (SOCC)*, Sep. 2009, pp. 227–230, doi: 10.1109/SOCCON.2009.5398054.

[25] L. Guckert and E. E. Swartzlander, "Optimized memristor-based multipliers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 2, pp. 373–385, Feb. 2017, doi: 10.1109/TCSI.2016.2606433.

[26] S. Baek, J. K. Eshraghian, S.-H. Ahn, A. James, and K. Cho, "A memristor-CMOS braun multiplier array for arithmetic pipelining," in *Proc. 26th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Genoa, Italy, Nov. 2019, pp. 735–738, doi: 10.1109/ICECS46596.2019.8964710.

[27] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electron.*, vol. 2, no. 7, pp. 290–299, Jul. 2019, doi: 10.1038/s41928-019-0270-x.

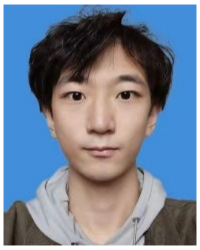
[28] J. S. Pannu, S. Raj, S. L. Fernandes, D. Chakraborty, S. Rafiq, N. Cady, and S. K. Jha, "Design and fabrication of flow-based edge detection memristor crossbar circuits," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 5, pp. 961–965, May 2020, doi: 10.1109/TCSII.2020.2984155.

[29] C. Fernandez, I. Vourkas, and A. Rubio, "Design and simulation of peripheral driving circuitry for computational ReRAM," in *Proc. 37th Conf. Design Circuits Integr. Circuits (DCIS)*, Nov. 2022, pp. 1–6.

[30] C. Fernandez, A. Cirera, and I. Vourkas, "Design exploration of threshold logic in memory and experimental implementation using knowm memristors," *Int. J. Unconventional Comput.*, vol. 18, pp. 249–267, Jan. 2023.

WEI ZHU is currently pursuing the B.S. degree in electronic information engineering with Xidian University, Xi'an, China. His research interests include in-memory processing and memristor-based logic circuits design.

YI-XING HE received the B.S. degree from Wuhan University of Technology, Hubei, China, in 2023. He is currently pursuing the M.S. degree with Xidian University, Xi'an, China. His research interests include antiferromagnetic device and MRAM-based in-memory processing circuits.


HAO-NAN LI received the B.S. degree from Hebei University, Hebei, China, in 2023. He is currently pursuing the M.S. degree with Xidian University, Xi'an, China. His research interests include memristor, its manufacture, and memristor-based logic circuits design.

XIAN-QIN LIU received the B.S. degree from Xidian University, Xi'an, China, in 2023, where she is currently pursuing the M.S. degree. Her research interest includes aluminium nitride-based memristor device.

SIWEN ZHANG received the B.S. degree from Xidian University, Xi'an, China, in 2023, where she is currently pursuing the Ph.D. degree. Her research interest includes microelectronics in low temperature.

LEI WANG received the B.S. degree from Chang'an University, Xi'an, China, in 2021. He is currently pursuing the M.S. degree with Xidian University, Xi'an. His research interests include oxide-based memristor and memristor-based logic circuits design.

JIANG ZHU received the B.S. degree from Xidian University, Xi'an, China, in 2022, where he is currently pursuing the M.S. degree. His research interests include ferroelectric film and its manufacture.

YUE-QI WANG received the B.S. degree from Changsha University of Science and Technology, Hunan, China, in 2022. She is currently pursuing the M.S. degree with Xidian University, Xi'an, China. Her research interests include ferroelectric film and its manufacture.

JINCHENG ZHANG (Member, IEEE) received the M.S. and Ph.D. degrees from Xidian University, China, in 2001 and 2004, respectively. He is currently a Professor with Xidian University. He has authored and co-authored more than 200 journal and conference papers. His current research interests include wide bandgap semiconductor GaN and diamond materials and devices.

YUE HAO (Senior Member, IEEE) received the B.S. degree in semiconductor physics and devices from Xidian University, China, in 1982, and the Ph.D. degree in computing mathematics from Xi'an Jiaotong University, Xi'an, in 1990. He is currently a Professor with the School of Microelectronics, Xidian University, and an Academician with Chinese Academy of Sciences, Beijing, China.

HAIJIAO HARSAN MA received the bachelor's degree in physics from Lanzhou University, in 2011, and the Ph.D. degree in condensed matter physics from the National University of Singapore, in December 2015. In 2018, he joined Xidian University. His research interests include nonvolatile memories, such as memrisotrs, ferroelectric materials, and in-memory computing based on these novel memories. He was awarded the Most Outstanding Ph.D. Thesis (Medal Prize) in Singapore by the Material Research Society of Singapore in 2016. In 2017, he was awarded the Humboldt Scholar.

• • •