Reliable Domain-Specific Exclusive Logic Gates Using Reconfigurable Sequential Logic Based on Antiparallel Bipolar Memristors

Taegyun Park, Yeong Rok Kim, Jihun Kim, Jinwon Lee, and Cheol Seong Hwang*

The development of memristor-based stateful logic circuits can minimize data movement during the computing process to achieve in-memory computing, mitigating the von Neumann bottleneck in the current computing architecture. Herein, a method to combine resistance–resistance (R–R) and voltage–resistance (V–R) logic gates to implement exclusive logic gates composed of APMR-two-2(XOR, IMP, RIMP) and APMR-three-4XOR, where APMR means antiparallel memristors with a series resistor, is suggested. The proposed gates can accelerate XOR logic operation in a single cycle and expand for the n-bit input. The performance of the proposed logic gate is then demonstrated with a 1-bit full adder–subtractor along with the comparison of an n-bit ripple carry adder. It shows that the implementation for the n-bit adder takes 4n+1 memristors within 2n+1 steps, which significantly improves the optimization in terms of space- and time-related costs compared with other memristive logic gates. Subsequently, the improved adder circuit can be further utilized to implement an n-bit multiplier. In addition, the evaluation of the device stress on the various logic gates confirms that the proposed logic gates are reliable.

1. Introduction

Modern computing systems are based on von Neumann architecture,[1] in which the processor and memory are physically separated. With the end of the era of Moore’s law[2] and Dennard scaling,[3] a new computing architecture with better efficiency in terms of area, latency, and power is required to compensate for data overflow.[4–7] In this context, the hybrid memory cube[8] and high bandwidth memory[9] have been proposed to reduce the physical gap between the memory and processor, also known as a memory wall. This could be achieved by enhancing bandwidth in the memory chip itself and stacking multiple memory chips by through-silicon via interconnections. However, despite the improvement of bandwidth, a portion of the physical gap still remains between memory and computing units.

The von Neumann bottleneck can be solved by in-memory computing (IMC) with emerging nonvolatile memory technology such as resistive switching random-access memory (RRAM). The high density and power efficiency can be achieved with a simple two-terminal metal–insulator–metal stack RRAM device. It takes advantage of the migration of charged species such as ions, electrons, and oxygen vacancies driven by field-induced drift and diffusion.[10] Through this characteristic, the device can make use of low-power and multilevel resistive states with switching modes of bipolar (BRS), unipolar (URS), and complementary resistive switching (CRS) depending on the conditional voltage.[11–13] The set and reset transition of memristors can change the state of the memristor to a low resistance state (LRS), Roff, and a high resistance state (HRS), Ron, respectively, by applying the threshold voltage (Vth) to the two terminals of the memristor. This property makes them suitable for digital and analog IMC.

The IMC or logic-in-memory encompasses combining the memory and computing units enhances the computation speed and energy in order of magnitude. For instance, a 32-bit addition takes ≈3.2 nJ in the conventional von Neumann architecture, considering dynamic random-access memory access energy, 32-bit adder operation, and cache write-back energies. However, only 0.1–1 nJ is necessary for the IMC approach depending on the adopted memristive logic gate types.[14] With further improvement of the logic family and the bandwidth of the IMC gate, there is still room for enhancement in computation speed and energy. So far, the general development direction of IMC has been to implement a functionally complete set of logic operations or Boolean operators. Similar to the trend of replacing general-purpose central processing unit (CPU) with domain-specific hardware, such as the graphic processing unit

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH

DOI: 10.1002/aisy.202100267
For the past decade, many types of IMC logic gates have been reported since the first logic family of IMPLY gates was conceived by Lehtonen et al. in 2009\cite{13} and demonstrated by Borghetti et al. in 2010.\cite{16} Using two parallel BRS memristors with a series resistor, they can be divided into two family categories: $V\rightarrow R$ logic\cite{17,18,19} and $R\rightarrow R$ logic.\cite{20,24} The $V\rightarrow R$ logic family is also known as sequential logic, which exploits the voltage form of input and the resistance form of output. In contrast, $R\rightarrow R$ logic, that is, stateful logic, utilizes resistance for both input and output.

The logic operation in the $R\rightarrow R$ logic family is based on the voltage divider of a memristive primitive of the circuit. The different basic logic gates of Boolean logic operations can be defined depending on the circuit configuration. In the authors’ previous review, PMR-two-2IMP,\cite{16} PMR-two-3NAND,\cite{21} PMASM-two-3NOR,\cite{25} and APMR-two-2(IMP, AND, TF)\cite{23} gates were introduced with the nomenclature rule and verified under sustainability, success, and stability constraints.\cite{26} As the input and output types are the resistances of the memristor, the $R\rightarrow R$ logic gates have an advantage in the cascading operation, which means that the output of the previous logic gate transfers to the input of the following logic gate. Therefore, the XOR logic function or any other combined gating can be implemented with multiple cycles using the cascading property. However, either high conditional bias or long latency due to the multiple cycles on combined memristive structures must be addressed for these cases. This can lead to more significant power consumption and a state drift problem.

On the other hand, the logic operation in the $V\rightarrow R$ logic family can be implemented by altering the bias voltage applied ($V_{bi}$) to two terminals of a memristor, providing high scalability, flexibility, and parallelism.\cite{27} The logic reconfiguration can be represented by a finite-state machine,\cite{17} providing high reconfigurable functions, such as AND\cite{28} and majority gate,\cite{29} in a single step. The implementation of XOR in $V\rightarrow R$ logic also requires multiple but fewer cycles. Moreover, it can increase the endurance of the memristors because half of the combinations of input states (‘00’ and ‘11’) do not participate in the resistive switching of the memristors owing to the zero voltage drop on the actual memristor. However, the complicated cascading between the subsequent gates could be a critical drawback of $V\rightarrow R$ logic gates. Consequently, the output of the first gate cannot be used in the input of the second gate, and the conversion from resistance to voltage for each cascading step has to be included, increasing the size, complexity, and power consumption.\cite{10}

In this work, a new method was suggested to solve this problem and accelerate XOR operation on the previously reported APMR-two-2(IMP, AND, TF) logic primitive circuit,\cite{23} composed of two anti-parallel BRS memristors (APM) and a series resistor ($R_s$). This new primitive circuit element was named “exclusive logic (Ex-logic) gate.” The proposed gate can implement XOR, IMP, and RIMP functions in a single cycle on predefined states with a new cascading method to perform a three-input XOR operation. Furthermore, the basic principle of the Ex-logic gate belongs to the $V\rightarrow R$ logic family, and the logic reconfiguration can be applied for functional completeness. At the same time, it can also feature the $R\rightarrow R$ logic family since the voltage divider is used on two Ex-logic gates for the three-input XOR operation. In this context, the feasibility and performance of the accelerated two-input and three-input XOR operations are demonstrated in a 1-bit full adder–subtractor (FAS) with the RRAM-based model. In addition, the efficiency of optimization in terms of space- and time-related costs for the $n$-bit ripple carry adder (RCA) is presented and compared with the previous memristive RCA implementation.\cite{30,39}

Next, the efficiency of implementation is further emphasized with the memristive $n$-bit multiplier. Moreover, the proposed algorithm has the advantage of lower device stress, which ensures the reliability of the logic operation. Although this work focuses on arithmetic logic circuits, the memristive crypto function, such as the advanced encryption standard,\cite{14,40} can be efficiently implemented with the proposed gate and will be presented elsewhere.

### 2. Reconfigurable Primitive Circuit and Path-Dependent State

The proposed primitive circuit consists of two APMs and an $R_s$, as shown in the inset of Figure 1. The flexible implementation in a 3D crossbar array (CBA) structure is possible. The APM can be configured by connecting two word-lines (WLs) and a common bit-line (BL). Similarly, the different logic primitive circuits, such as PMR-two-2IMP, PMR-two-3NAND, and PMASM-two-3NOR, are shown in Figure S1, online Supporting Information (SI). The logic primitive circuits can be reconfigured by selecting a combination of memristors. For instance, if the $2 \times 3$ CBA structure is given as in Figure 1, applying the conditional bias to WL$_{3t}$ and WL$_{5t}$ and connecting BL$_2$ to the ground (GND) will result in the APM structure in the M$_{12}$ cell, while biasing WL$_{1}$ and WL$_{2}$ and connecting BL$_1$ to the GND result in the PMR-two-2IMP gate in the M$_{11}$ and M$_{12}$ cells. Here, the WL is identified by its number and top (t) or bottom (b) position, given as the subscript (e.g., WL$_{3t}$ means the third WL located on top of the M). A memristor is indicated by the WL and BL numbers in the subscript (e.g., M$_{11}$ means the memristor is located at the cross point of WL$_{1t}$, WL$_{5b}$, and BL$_2$). The importance of the compatibility of the primitive circuits will be clarified in a later section.

Meanwhile, the proposed primitive circuit structure is identical to the previously reported APMR-two-2(IMP, AND, TF) gate with a different logic configuration as it adopts the $V\rightarrow R$ logic family. For the APMR-two-2(IMP, AND, TF) gate within the $R\rightarrow R$ logic family, the different bias voltages (to select a basic logic function) are applied to the top and bottom WLs and connected GND to the shared BL. The initial resistive states of the APM are predefined to the input states of $p$ and $q$. In contrast, the $V\rightarrow R$ logic family in this work applied a common bias voltage (input $p$) to the top and bottom WLs of the selected APM, for which the resistive state was predefined (to select the basic logic function), while the shared BL is biased to another value (input $q$). The output is then represented by the final resistance
of the combined memristor (between \( R_{\text{on}} \) and \( R_{\text{off}} \)). The \( R_{\text{on}} \) and \( R_{\text{off}} \) represent logical '1' and '0,' respectively, whereas the input states '1' and '0' are defined by the bias voltages of \( V_{\text{th}} \) and GND, respectively. Depending on the input combinations, the resistance of the APM, which represents the output, can be controlled. If the input states are either '11' or '00', no voltage drop across the APM occurred (the corresponding WL and BL voltages are identical), rendering the memristor states unvaried. On the other hand, when the input combination is '10', \(+V_{\text{th}}\) is applied to the APM. If the initial states of the two memristors are \{HRS, HRS\}, the \(+V_{\text{th}}\) can switch the resistance of the memristor located in a forward direction (FM) from '0' to '1' (set switching), resulting in the \{LRS, HRS\} state. Consequently, the \{LRS, HRS\} state is defined as the '1+' state. On the other hand, when the input combination is '01', \(-V_{\text{th}}\) is applied; the memristor located in the reverse direction (RM) is now switched from '0' to '1,' resulting in \{HRS, LRS\} states, which are defined as the '1-' state. Here, it should be noted that both the '1+' and '1-' states of the APM show the identical current of set state, but they can be discerned by the reading scheme in Figure 2a if necessary.

Compared with the conventional state system consisting of a single BRS memristor, as shown in Figure 2b, the suggested state system contains one additional logical state, as shown in Figure 2c, which can be regarded as a Moore machine. This new configuration offers great potential for implementing the single-cycle XOR function, which was improbable for the conventional single BRS memristor system. The essential aspect of this system is that the combinational states of '1+' and '1-' have equivalent resistances but are different in the subsequent state transition with a unique path dependency, the so-called "path-dependent state," which is shown in Table 1. For example, the conventional state system defines only '1' and '0' states for the LRS and HRS of a BRS memristor, respectively. The peripheral circuit can be minimized to account for the two different logic systems between single and combined memristive structures by sensing the equivalent resistance with one threshold line.

The conversion from '1' to '0' requires only '01' input state, which is the reset process of the memristor. In contrast, to reset the '1-' to '0' state in the APM system, different '10' input states have to be applied. On the other hand, the '1+' state has the same response as the conventional single BRS memristor system, meaning that the '01' input states reset it. This additional response method can be utilized to efficiently increase reconfigurability for functional completeness. Another crucial merit of

![Figure 1](image1.png)

**Figure 1.** The proposed Ex-logic gates implementation on 3D CBA. The inset represents a unit cell of the Ex-logic gate consisting of two APMs with \( R_s \). The forward and reverse memristors in the primitive circuit are represented by blue and orange boxes. The flexible logic reconfiguration can adopt other logic primitive gates.

![Figure 2](image2.png)

**Figure 2.** The path-dependent logic state in the Ex-logic gate. a) The state reading scheme to determine the system. The state transition diagram based on the finite-state machine on b) a conventional single BRS memristor and c) the proposed Ex-logic gate.
this primitive circuit is its convenient cascading property, despite it being a sequential logic, which is demonstrated in the FAS.

3. Logic Reconfiguration Based on the Proposed Primitive Circuit

The proposed primitive circuit provides three possible states that allow the configuring of three logic functions, XOR_pq, IMP_pq, and RIMP_pq, in this Ex-logic gate. The combined gate is named the APMR-two-(XOR, IMP, RIMP) gate with the predefined states (s) in a single cycle, as shown in Figure 3a. For the sake of clarity, T represents the bias condition of the three terminals (two WLs and one BL) of the proposed primitive circuit. When T is pq, it means that the input bias of p is applied to the two WLs of the memristor (WL1, WL1b) and that of q is simultaneously applied to BL of the memristor (BL1). The variable s represents the final state of the M11 cell after the Ex-logic operation on the initial s.

In this context, one of the advantages of using the path-dependent state on the primitive circuit is the functionality to implement the XOR logic operation in a single cycle with a simple circuit configuration. In addition, the proposed Ex-logic gate substantiates functional completeness. The reconfigured logic operations of the XOR_pq function with the given reconfiguration number (r) of 1, the number of steps from the configured function, are shown in Figure 3b. This means that the primitive circuit first operated the XOR_pq (light blue background in Figure 3b), while the other logic operations (light green background in Figure 3b) could be achieved in one additional operation step. Here, T is introduced to distinguish the first and second steps. For example, T = T1 T2 means the T1 bias is applied to WL1t and WL1b and a T2 bias is applied to BL1. Thus, the NAND_pq logic operation can be achieved by applying either T = p1 or 1q to the result of the previous XOR_pq function. The other configured functions of IMP_pq and RIMP_pq for r of 1 are summarized in Figure S2-1 in Section II of Supporting Information. Consequently, the 16 Boolean logic operations are presented in Table 2, with a maximum r of 2.

Most logic operations can be implemented within two cycles, except for the NIMP and XNOR logic functions, which can be realized with three sequential cycles. However, the number of steps can be further reduced with the aid of other logic primitive circuits if necessary. As the Ex-logic gate supports compatibility in the 3D CBA structure, the primitive logic gate of PMR-two-2IMP can simultaneously be implemented. For the forward and reverse directions of a single BRS memristor, their states are given by two equations.

\[ s' = (\text{RIMP}_pq) s + (\text{NIMP}_pq) \quad (1.1) \]
\[ s' = (\text{IMP}_pq) s + (\text{RNIMP}_pq) \quad (1.2) \]

For example, after the logic operation of XOR_pq in the Ex-logic M11 gate, fM11 can be partially selected, which is defined as the partial selection method, by connecting WL1t to the bias voltage and BL1 to GND. Then, the NIMP_pq function can be implemented in a single cycle, the same result as the primitive logic gate of a single BRS memristor proposed by Linn et al. for SM-two-(RIMP, NIMP) gate. Similarly, the RNIMP_pq function can be obtained by selecting rM11. In particular, Equation (1–2) shows an essential idea to implement the subtractor using Ex-logic gates, which will be discussed later. In addition, XNOR_pq can be reduced to a single cycle when one of the nodes is connected to the NOT gate. The negation of one input bit, defined as the negation method, on the XOR_pq operation coincides with the XNOR_pq function. The two reduction techniques of partial selection and negation methods will be exploited in the FAS implementation, as shown later.

4. Cascading via the V-2R Logic

The V–R logic family has an inherent cascading problem between consecutive logic gates due to the discrepancy between voltage-type input and resistance-type output. The correlation between the previous gate must be presented for the multi-input logic gate, but it is cumbersome to construct a relationship with the discrepancy. Generally, an external circuit, such as a potentiometer, is required to convert the resistance to voltage, which increases the space- and time-related costs in IMC applications. This requirement, however, can be mitigated by adopting domain-specific gates. Specifically, the target domain of the proposed Ex-logic is to accelerate the XOR function rather than the implementation of all logic functions. The domain-specific cascading method for V–R logic can be applied for the XOR logic function, but the generality of the application can be increased by combining it with the different logic implementation methods. Subsequently, the additional external circuit for the input conversion becomes unnecessary through the combination of V–R and R–R logic implementations, which is referred to as V–2R logic and explained below.

The principle of the V–2R logic concept could be understood by the voltage divider effect between two Ex-logic gates, M11 and M31, connected in series. This V–2R logic circuit can be used to implement a three-input XOR operation, for which the truth table is shown in Figure 4a. M11 cell is the first Ex-logic gate that stores the result of the XOR operation with the two inputs of d1 and d2 (WL1t, WL1b = d1 and BL = d2). For this XOR operation, M11 should be initialized to s = 0 (both fM11 and rM11 are in

Table 1. The proposed path-dependent states are compared with the conventional logic state system. The ‘1+’, ‘1’, and ‘0’ represent the resistive state combinations of {LRS, HRS}, {HRS, LRS}, and {HRS, HRS} in the antiparallel BRS memristors.

| Conventional Logic States | LRS | HRS |
|---------------------------|-----|-----|
| '1'                       |    |    |
| '0'                       |    |    |
| Proposed Logic States     | '1+'| '1' | '0' |

Adv. Intell. Syst. 2022, 4, 2100267 2100267 (4 of 19) © 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH
The third input state of \( d_3 \) is stored in the fM21 of the M21 cell as resistance, and the rM21 cell is settled to the HRS state at the initialization step. Then, the two Ex-logic XOR gates are connected in such a way that WL1t and WL1b are biased with the conditional voltage, and WL2t and WL2b are connected to GND, as shown in Figure 4b. fM31 is currently neglected.

When the conditional voltage of \( V_a \) is applied to the two series Ex-logic gates, the voltage drop on the M21 cell can be calculated by the voltage divider between the two series cells.

\[
V_{M21} = \left( \frac{R_{M21}}{R_{M11} + R_{M21} + 2R_s} \right) V_a \tag{2}
\]

where \( R_{M11} \) and \( R_{M21} \) represent the equivalent resistance of M11 and M21 cells, respectively. \( 2R_s \) accounts for the two Ex-logic gates connected in series as each has an \( R_s \). As the number of ‘1’ state in the M11 cell increases after the first XOR operation between \( d_1 \) and \( d_2 \), the equivalent resistance for the M11 cell decreases from \( R_{off}/2 \) down to \( R_{on}R_{off}/(R_{on} + R_{off}) \), resulting in an increment in the voltage drop of the M21 cell. The critical feature of this operation is that the minimum of the equivalent resistance of M11 is not \( R_{on}/2 \) as the proposed state transition diagram of Ex-logic does not allow the \( \{LRS, LRS\} \) state. In addition, the input condition \( (d_1, d_2) \) of (11) results in the identical output of M (no change) when \( (00) \) is the input. Thus, cases 7 and 8 in the truth table become identical to cases 1 and 2, respectively. This makes the proposed voltage divider scheme possible for the three-input XOR operation as the equivalent resistance of the previous XOR operation containing only one number of ‘1’ is always the lowest. As a result, the cascaded XOR result is stored as a form of equivalent resistance of the M21 cell. This gate can be called “APMR-three-4XOR.” Therefore, the proposed algorithm enables a connection between \( V\rightarrow R \) and \( R\rightarrow R \) logics without requiring an external active circuit. Further, this \( V\rightarrow 2R \) method can be expanded for the \( n \)-input XOR logic operation that follows Algorithm 1.

![Figure 3](image-url)
Table 2. Implementation of 16 Boolean logic operations based on the reconfiguration.

| Logic operation | Input | Output | Cycle 1 \((r = 0)\) | Cycle 2 \((r = 1)\) | Cycle 3 \((r = 2)\) |
|----------------|-------|--------|---------------------|---------------------|---------------------|
|                | \(p\) | \(q\)  | \(s\) | \(T\_1\) | \(T\_2\) | \(s'\) | \(T\_1\) | \(T\_2\) | \(s''\) | \(T\_1\) | \(T\_2\) | \(s''\) |
| TRUE           | 0     | 0      | 1     | 1           | 0     | 0     | 1           | 1           | 1       |        |        |        |        |
|                | 0     | 1      | 1     | 1           | 0     | 1     | 1           | 1           | 1       |        |        |        |        |
|                | 1     | 0      | 1     | 1           | 0     | 0     | 1           | 1           | 1       |        |        |        |        |
|                | 1     | 1      | 1     | 1           | 1     | 1     | 1           | 1           | 1       |        |        |        |        |
| FALSE          | 0     | 0      | 0     | 0           | 0     | 0     | 0           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 0     | 0           | 0     | 1     | 0           | 0           | 0       |        |        |        |        |
|                | 1     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
| \(p\)          | 0     | 0      | 0     | 0           | 0     | 0     | 0           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 0     | 0           | 0     | 1     | 0           | 0           | 0       |        |        |        |        |
|                | 1     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
| \(q\)          | 0     | 0      | 0     | 0           | 0     | 0     | 0           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 0     | 0           | 0     | 1     | 0           | 0           | 0       |        |        |        |        |
|                | 1     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
| \(\text{NOT}_p\) | 0     | 0      | 1     | 0           | 0     | 0     | 0           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 1     | 0           | 0     | 1     | 0           | 1           | 0       |        |        |        |        |
|                | 1     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
| \(\text{NOT}_q\) | 0     | 0      | 1     | 0           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 0     | 0           | 0     | 1     | 0           | 0           | 0       |        |        |        |        |
|                | 1     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
| \(\text{AND}_pq\) | 0     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 0     | 0           | 0     | 1     | 0           | 1           | 0       |        |        |        |        |
|                | 1     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
| \(\text{NAND}_pq\) | 0     | 0      | 1     | 0           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 1     | 0           | 0     | 1     | 0           | 1           | 0       |        |        |        |        |
|                | 1     | 0      | 1     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 1     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
| \(\text{OR}_pq\) | 0     | 0      | 0     | 0           | 0     | 0     | 0           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 0     | 0           | 0     | 1     | 0           | 0           | 0       |        |        |        |        |
|                | 1     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
| \(\text{NOR}_pq\) | 0     | 0      | 1     | 0           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 1     | 0           | 0     | 1     | 0           | 1           | 0       |        |        |        |        |
|                | 1     | 0      | 1     | 1           | 0     | 0     | 1           | 1           | 0       |        |        |        |        |
|                | 1     | 1      | 1     | 1           | 0     | 0     | 1           | 1           | 0       |        |        |        |        |
| \(\text{XOR}_pq\) | 0     | 0      | 0     | 0           | 0     | 0     | 0           | 0           | 0       |        |        |        |        |
|                | 0     | 1      | 0     | 0           | 0     | 1     | 0           | 0           | 0       |        |        |        |        |
|                | 1     | 0      | 0     | 1           | 0     | 0     | 1           | 0           | 0       |        |        |        |        |
|                | 1     | 1      | 0     | 0           | 0     | 1     | 0           | 1           | 0       |        |        |        |        |
Table 2. Continued.

| Logic operation | Input | Output | s       | Cycle 1 (r = 0) | Cycle 2 (r = 1) | Cycle 3 (r = 2) |
|-----------------|-------|--------|---------|-----------------|-----------------|-----------------|
|                 | p     | q      |         | T   T1 T2 s'    | T   T1 T2 s''   | T   T1 T2 s''' |
| IMP_pq          | 0     | 0      | 1       | 1   0 0 1-     | 0   1 1-        | -   -           |
|                 | 0     | 1      | 1       | 0   1 1-       | 0   1 1-        | -   -           |
|                 | 1     | 0      | 0       | 1   0 0       | 1   0 0         | -   -           |
|                 | 1     | 1      | 1       | 1   1 1-      | 1   1 1-        | -   -           |
| RIMP_pq         | 0     | 0      | 1       | 1   0 0 1+    | 0   0 1+        | -   -           |
|                 | 0     | 1      | 1       | 0   1 0       | 0   1 0         | -   -           |
|                 | 1     | 0      | 1       | 1   0 1-     | 1   0 1-        | -   -           |
|                 | 1     | 1      | 1       | 1   1 1-      | 1   1 1-        | -   -           |
| p RNIMP q       | 0     | 0      | 1       | 1   0 0 1+   | 0   0 1+        | 0   1 0         |
|                 | 0     | 1      | 1       | 0   1 0      | 0   1 1-       | 1   0 0         |
|                 | 1     | 0      | 1       | 1   0 1-     | 1   1 1+      | 1   1 1-        |
|                 | 1     | 1      | 1       | 1   1 1-    | 1   1 1-       | 1   1 0         |
| p NIMP q        | 0     | 0      | 0       | 0   0 0       | p1 0 1 1-     | 1p 1 0 0       |
|                 | 0     | 1      | 0       | 0   1 1-     | 0   1 1-      | 1   0 0         |
|                 | 1     | 0      | 0       | 1   0 1-     | 1   1 1+      | 1   1 1-        |
|                 | 1     | 1      | 0       | 1   1 0     | 1   1 0       | 1   1 0         |
| p XNOR q        | 0     | 0      | 1       | 1   0 0       | q1 0 1 1-     | q1 0 0 1-      |
|                 | 0     | 1      | 0       | 0   1 1-     | 1   1 1-      | 1   0 0         |
|                 | 1     | 0      | 0       | 1   0 1-     | 0   1 0      | 0   0 0         |
|                 | 1     | 1      | 0       | 1   1 0     | 1   1 0       | 1   0 1-        |

Figure 4. Multi-input logic operation using Ex-logic gates. a) The truth table of the three-input XOR logic function. b) The schematic of the 1-bit FAS circuit consisting of six memristors. The result of the APMR-three-4XOR gate is stored in the M21 cell when conditional voltages are applied (blue).
5. The Memristor Model and Criteria for the Ex-Logic Gate

The selection of a resistive switching memristor is vital in implementing the IMC logic gate. The energy and latency of the logic operation are closely related to the memristor’s switching property. In addition, the characteristics of the memristor, such as endurance, retention, and variation, can affect the state transition. For simplicity, the consideration is limited to logic feasibility constraints based on the memristor model. Moreover, to operate the Ex-logic gates in parallel, the state transition diagram, mentioned in Figure 2c, must be satisfied with the same magnitude of the applied voltage for the set and reset. Thus, the reset voltage magnitude of the memristor has to be smaller than the set voltage magnitude for the conditional switching in the presence of $R_s$. Otherwise, the magnitude of the reset voltage becomes higher than $V_{th}$, which renders the circuit operation unfeasible. The candidate memristor for this property is reported on Cu–Te-based conductive bridge random access memory,[42] Cu/Go/Pt,[43] and a TiO$_x$-based memristor.[23,44] Among the candidates, the TiO$_x$-based memristor is selected to demonstrate the 1-bit FAS. Here, the dynamics of the conductive filament growth/dissolution compact model[45] are used for the memristor with the switching conditions, as shown in Figure 5a. It is worthwhile to note that the correction factors to address nonideal effects, such as sneak current and parasitic capacitance, are not considered in this work, which should be addressed in practical implementation.

The optimal conditions of $V_a$ and $R_s$ for the APMR-two-2XOR and APMR-three-4XOR gates can be found using the success constraints for each case of operations. A total of eight constraints have to be fulfilled (three constraints for the two-input gate and five constraints for the three-input gate), as shown in Section III of SI in detail. The constraints can be represented by linear equations, so that they can be plotted in Figure 5b to find an optimal solution. The yellow region, formed by two lower boundaries (blue lines) and one upper boundary (red line), represents the qualified region that satisfies the success constraints of the logic operation. Among the eight constraints, three of each Ex-logic operations participate in forming the optimal region. These are called active constraints and play an essential role in finding the appropriate $R_s$ value. Although the variation in devices is not considered explicitly, the logic constraints would have a certain range instead of the discrete value, as shown in Section IV of SI. For this case, the applied voltage at the given $R_s$ can be selected above the upper variation line at the cost of the device stress, in which the concept will be discussed in detail in a later section. Thus, the optimization between the device stress and variation is required. For the demonstration, the optimal $V_a$ and $R_s$ are selected to be $-1.58$ V and 100 $\Omega$, respectively. The reason for the selection with the active constraints will be highlighted in a later section.

After logic operation, the stability constraint is then appraised. When $V_a$ with the same magnitude is applied to the two- and

---

**Algorithm 1. Ex-logic $n$-input XOR logic operation.**

```
Procedure n-input XOR operation ($M_{1:n1}, d_{1:n}$)
    For $n \leftarrow 1 \text{ to } N-1$ do
        if $n \leftarrow 1$
            $M_1 \leftarrow 0$ //initialization
            $M_1 \leftarrow d_1 \oplus d_2$ //The result of two-input XOR
        else
            $rM_n \leftarrow 0$ //initialization
            $fM_n \leftarrow d_{n+1}$
            $M_n \leftarrow M_{n-1} \oplus d_n$ //The result of $n$-input XOR
        end if
    end for
    return $M_n$
end procedure
```

---

**Figure 5.** Optimization method using the BRS memristor model for the Ex-logic gates. a) The $I$–$V$ characteristic of a memristor model based on the experiment. b) The optimal regions to find the optimal $V_a$ and $R_s$ in the APMR-two-2XOR, IMP, RIMP) and APMR-three-4XOR gates. The constraint conditions for two-input and three-input logic operations are derived from the switching cases in Section III of Supporting Information.
three-input Ex-logic gates, the voltage drop on the output memristors with the two-input case is higher than the latter. For example, the required set voltages of the memristor in the two- and three-input cases are \(-1.546\) and \(-1.577\) V, respectively, when \(R_c = 100\ \Omega\). The relevant stability constraints determine these values. Subsequently, the selected \(V_s\) of \(-1.58\) V imposes a larger overvoltage for the two-input case \((|1.58| - |1.546| = 0.034\) V) as compared with the three-input case \((|1.58| - |1.577| = 0.003\) V). Thus, the drift speed of ions in the memristor (if the switching mechanism isionic) of the two-input case is faster than that of the three-input case, resulting in a faster switching speed. The stability constraints with the dynamic switching speed are considered as described in Section V of SI. Moreover, the different overvoltage for the RRAM operation requires the selection of shorter and longer pulse lengths for the two- and three-input Ex-logic operations.

Therefore, 35 and 750 ns, respectively, are selected for this work. As a result, complete Boolean logic operations can be demonstrated. The state transitions based on the state transition diagram and NAND logic operation are conducted as demonstrated. A total of 12 possible cases can be obtained from the state transition diagram (four input cases for three initial states). However, inputs of ‘11’ and ‘00’ naturally retain the initial state (‘0’, ‘1’ or ‘0’), so only six cases with effective bias voltage being applied to the APM are depicted in Figure 6a. The ‘0|0|1’ state is denoted as the transition from ‘0’ state to ‘1’ state under the ‘01’ input combination. Four cases (cases 1–4) change the state, but the other two do not (cases 5 and 6). The demonstration of the NAND logic operation according to Figure 3b is shown in Figure 6b. The implementation takes two sequential steps XOR_{pq} à NAND_{pq} (\(T = p1\)) for each case. As there are four cases for the two input combinations, a total of eight results is presented. The two dashed lines in the result represent the initial and final points, respectively, when the conditional voltage \(V_s\) is applied to the Ex-logic gate. The resistance is derived with a read voltage of 0.3 V, and the results show the feasibility of the Ex-logic gate.

6. Demonstration of a 1-Bit Full Adder–Subtractor and Implementation for the Multibit Input

Full adder (FA) is a combinational circuit that performs the addition of two binary numbers, \(A (A_1, A_2, A_n)\) and \(B (B_1, B_2, B_n)\), with a bit-length \(n\). It takes two inputs and carry-in and produces sum and carry-out, for which the truth table is shown in Figure 7a. The sum of the half adder (HA) is first implemented by the APMR-two-2XOR gate, while that of the FA can be by the APMR-three-4XOR gate. The carry can be calculated by reconfiguring AND and OR functions using the APMR-two-2XOR gate, but borrowing from the other primitive gates, such as the majority gate (SM-three-1MAJ), could be an even more efficient method.[29]

Further, using the flexible connection in the 3D CBA structure, the carry-out bit can be achieved in a single cycle. Combining with the Ex-logic gates, the 1-bit FA can be implemented on five memristors in three cycles, as shown in Figure 7b. The circuit structure is identical to that shown in Figure 4b. The carry-in bits are initialized to the two memristors of \(rM_{21}\) and \(fM_{31}\), and the other memristors are set to the HRS state in parallel. After initialization, an APMR-two-2XOR gate is biased with two inputs of \(V_A\) and \(V_B\) on the \(M_{21}\) to achieve the sum of the HA. Then, the carry-out bit is calculated on a single memristor of \(fM_{31}\) by applying \(V_B\) to \(WL_{3t}\) and \(V_A\) to \(BL_1\), which corresponds to the SM-three-1MAJ gate.

Finally, the sum of the FA can be calculated and stored in the \(M_{21}\) gate by operating the APMR-three-XOR gate connecting the \(M_{11}\) and \(M_{21}\) cells in series. The demonstration of the 1-bit FA is shown in Figure 7c, where the four dashed lines are the reference lines dividing five regions to indicate the initial state, three operation regions in the clocking scheme table, and the final state of the memristors.

Similarly, the 1-bit full subtractor (FS) can be implemented in the same circuit, as shown in Figure 8. The calculation of the borrow-out bit can be done using two IMP and RNIMP logic operations as it can be expressed by the following equation.

\[
B_{out} = (A + B)B_m + (\overline{A}B)\overline{B}_m
\] (3)

This is the same implementation on a single reverse memristor mentioned in Equation (1–2). The designer can either choose the forward or reverse memristor to implement the borrow-out bit (or carry-out in FA) by changing the bias voltage direction. The forward memristor \(fM_{31}\) is selected for the bring-in bit by applying the \(V_B\) to \(WL_{3t}\) and \(V_A\) to \(BL_1\). As a result, in the aforementioned partial selection, selecting a single memristor from an Ex-logic gate, and negation, inversing one of the input bits, are applied to the Ex-logic gates to further optimize the FAS implementation. Consequently, the optimized FAS can operate in three operational cycles with five memristors.

For a practical comparison between state-of-the-art memristive logic circuits, the \(n\)-bit FAS is considered. First, the RCA method is selected for larger \(n\)-bit binary numbers, which can be constructed by cascading the 1-bit FA. The main delay of the RCA adder originates from the computation of the carry-out as the previous FA gate must transfer the carry-out to the carry-in of the next adder gate. Although the propagation delay can be decreased by other types of adders, such as carry look-ahead and parallel prefix, the RCA still has the advantage of a simpler memristive structure without requiring transistors for better modularity and periphery circuits. Figure 9 shows the block schematic diagram and clocking scheme table of a 4-bit RCA \((n = 4)\) using 1-bit FAs of Ex-logic gates. Instead of repeating the entire process of 1-bit FA for \(n\) times, the sum operations of APMR-three-4XOR can be applied in parallel through the WL, which reduces the total latency of the RCA from 4\(n\) to 2\(n\) + 1. The total number of memristors used in the \(n\)-bit adder is 4\(n\) + 1, where the +1 accounts for the memristor storing carry-out bit. As a result, the 4-bit RCA can be implemented on 17 memristors in 9 steps, as shown in Figure 9b. Table 3 compares \(n\)-bit FA implementation using other memristive logic gates. Here, it should be noted that other complex logic gates that consist of transistors and memristors[46–48] are not considered because different space-related comparisons are required. The number of memristors (space-related) and steps (time-related) are trade-off relationships. Due to its cascading ability, the IMPLY-type logic gate uses fewer memristors (low space-related cost) to implement an \(n\)-bit FA but requires more steps (high time-related cost) in return. ORNOR gate is a type of IMP logic gate but uses three memristors instead of two, making it possible to
implement the combination of the OR and NOR functions (ORNOR).\[^{39}\] Combining ORNOR and AND functions can reduce the costs of an XOR operation as compared with that of an IMP function.

\[
p_{\text{XOR}}q = (p\text{IMP}q)\text{IMP}((q\text{IMP} p)\text{IMP}0) = \overline{pq} + \overline{p} + q \quad (4)
\]

On the other hand, the SIXOR gate takes advantage of a single-cycle XOR operation and reduces the number of steps (low time-related cost),\(^{49}\) but more memristors (high space-wise cost) have to be included and initialized to prevent input drift problems. Thus, overall, the proposed Ex-logic gate shows the best optimization in terms of space- and time-related costs.

Figure 6. Demonstration of the Ex-logic gates based on the memristor model with the device parameters as shown earlier. a) The demonstration of the state transition diagram. The results of the input biases '00' and '11' are not shown as no logic transition is made on the zero-bias condition. b) Results of the NAND logic operation based on the reconfiguration of the XOR logic function with \(T = p1\) and a read voltage of 0.3 V. Each case corresponds to two results representing before the reconfiguration (XOR) and after the reconfiguration (NAND).
In addition, unlike other R–R logic families, the proposed Ex-logic gates rely on the V–2R logic family, which increases the conversion speed between memristive addition and subtraction in the FAS circuit. Although initialization is not considered in the comparison, the R–R logic family requires the additional remapping of the input bit to each corresponding memristor or an additional logic operation when the transition between FA and FS is required. Consequently, this increases overall device stress and latency as the state transitions of the memristor are involved. In contrast, the proposed V–2R logic family does not involve additional initialization and can be implemented by simply negating the input bias lowering overhead cost.

7. Memristive Multipliers Using Ex-Logic Gates

Along with adder implementation, a multiplier is involved in many essential applications. The main optimization of the memristive multiplier (mMP) comes from reducing the partial products, which have been attempted by shift-and-add, Wallace-tree, and Dadda.[50–54] When utilizing the massive parallelism of the CBA structure and IMC, the operation speed of the mMP can be enhanced, while the area overhead remains minimal. In this work, a new mMP design is proposed using improved Ex-logic FAS with the modified Wallace-tree algorithm. Figure 10 shows the method used to represent the partial products of mMP, which are indicated by the black dot in the inset. The partial products...
can be distinguished by multiplicand \((A_n)\) and multiplier \((B_n)\) bit positions as in \(P_{01}\), which represents the partial product of \(A_0\) and \(B_1\) input bits. Unlike the conventional Wallace-tree reduction method, carry is not involved in the grouping step of the reduction. Instead, the carry is propagated through the entire system. Although the propagation delay can be the bottleneck of the operation, as already mentioned in the 4-bit RCA implementation, the parallelism of finding the sum in the simpler circuit structure can mitigate the overall speed issue.

The modified Wallace-tree reduction consists of three components: 1-bit FA, one 1-bit HA, and one fM of the Ex-logic gate (Figure 11). The reduction method starts with categorizing the boxes into groups of two sets of partial products in the same row, as shown in the red boxes. The \(P_{00}\) position is always left with one partial product as the least significant bit of the product is the first product itself, so the single fM takes its place in the structure. Next, if the first red box is composed of \(P_{10}\) and \(P_{01}\), the first carry bit can be generated. Thus, HA can be used to calculate the sum and carry-out. Therefore, the red box is implemented with the HA, and the other boxes are implemented with the FA as the carry-in should be included. For the remaining ungrouped partial products, they should be implemented with the HA between carry-in and partial product, but the incompatibility between in and out must be addressed as input states in the proposed APMR-two-2XOR gate should be mapped in the voltage form, whereas the cascaded carry-in would be the resistance form. Thus, FA is used instead with the auxiliary input of '0' to convert the voltage-type partial product to resistance, and

Figure 8. Demonstration of a 1-bit FS using Ex-logic gates. a) The truth table of the 1-bit FS represented by eight cases. b) Clocking scheme on each memristor and c) demonstration on five memristors in three cycles with a pulse delay of 150 ns and read voltage of 0.3 V. The dashed gray lines divide initial state, three logic operation steps, and final state. The conversion from the FA to the FS is possible by negating the first input bit.
Figure 9. Implementation of 4-bit RCA. a) The block schematic of RCA. The gray block represents 1-bit FA, and the previous carry-out of the FA block is propagated to the next carry-in of the block. b) The output address of RCA in the 3D CBA configuration. c) The proposed clocking scheme for each memristor. A total of 17 memristors with 9 cycles are required to implement 4-bit RCA consisting of four APMR-two-3XOR, SM-three-1MAJ, and APMR-three-4XOR gates in the proposed clocking scheme. The 4-bit FS can be implemented in the same circuit by negating the input bit A.

Table 3. Comparison between the proposed logic gates for n-bit FA implementation.

| Design | Logic family | Method | Number of memristors | Number of steps |
|--------|--------------|--------|----------------------|-----------------|
| Reference | | | Total | n = 16 | Total | n = 16 |
| [30] | R–R | IMPLY Serial | 2n+3 | 35 | 22n | 352 |
| [31] | R–R | IMPLY Serial | 3n+3 | 51 | 29n | 464 |
| [31] | R–R | IMPLY Parallel | 9n | 144 | 5n+18 | 98 |
| [32] | R–R | IMPLY Parallel | 4n+1 | 65 | 5n+16 | 96 |
| [33] | R–R | IMPLY Semi-Serial | 2n+6 | 38 | 10n+2 | 162 |
| [34] | R–R | IMPLY Semi-Parallel | 2n+3 | 35 | 17n | 272 |
| [35] | R–R | MOL | 4n | 64 | 6n+1 | 97 |
| [36] | R–R | MAGIC Area Optimized | 5 | 5 | 15n | 240 |
| [36] | R–R | MAGIC Latency Optimized | 11n–1 | 175 | 12n+1 | 193 |
| [36] | R–R | MAGIC Transpose I | 22n–3 | 349 | 15n+1 | 241 |
| [36] | R–R | MAGIC Transpose II | 13n–3 | 205 | 10n+3 | 163 |
| [37] | R–R | MAGIC RCA | 14n+1 | 225 | 12n+1 | 193 |
| [38] | R–R | MAGIC Native mapping | 15n | 240 | 12n | 192 |
| [38] | R–R | MAGIC Compact mapping | 24n | 384 | 16n | 256 |
| [39] | R–R | ORNOR | 6n+6 | 102 | 2n+15 | 47 |
| [49] | R–R | SIXOR | 6n+3 | 99 | 4n+2^4 | 66 |

Additional initializations are required during the FA operation. Also, FLEX OR and SIXOR cannot operate in parallel without separate sources and circuits.
the sum can be calculated using the APMR-three-4XOR gate. Therefore, partial products can be combined in the block diagrams, as shown in Figure 11b, d, and f. The white box represents the single memristor, the gray box represents the HA, and the dark-gray boxes represent the FAs. Thus, \( n \)-bit mMP using Ex-logic gates can be represented with \( \frac{n^2 + n}{2} \) blocks. Consequently, the proposed methodology for memristive multiplication comprises the same components of logic operations, which are APMR-two-3XOR, SM-three-1MAJ, and APMR-three-4XOR gates but in different combinations and initializations.

For representation, the \( 2 \times 2 \) mMP is demonstrated in Section VI of SI, and \( 3 \times 3 \) mMP is implemented in Figure 12. To avoid confusion with the previous notation, the sum of partial products is expressed by \( S_{r,c} \), where \( r \) and \( c \) represent row and column numbers in the reduction method diagram, as shown in Figure 11. In the given blocks, the reduction of the partial products can be divided into three stages: partial product mapping, carry cascading, and folding (Figure 12a). The first stage involves storing the partial products in the system. This can be mapped using the APMR-two-2XOR gates by summing two partial products on each designated block. Thus, the total steps of partial

Figure 10. In-memory Wallace tree structure for mMP. The partial product can be represented by the black dots, as shown in the inset of the figure. The partial products can be named using multiplicand bit and multiplier bit positions.

Figure 11. Proposed modified Wallace tree reduction method for the mMP using Ex-logic FAs. a) Grouping method in Wallace tree for the \( 2 \times 2 \) partial product matrix and b) block diagrams using FAs, a HA, and a fM to store the first product. c,d) \( 3 \times 3 \) partial product matrix and e) and f) \( 4 \times 4 \) partial product matrix.
Figure 12. Implementation of 3 × 3 mMP using Ex-logic gates on a) three stages of partial product mapping, carry cascading, and folding take 15 cycles to compute results on b) the 4 × 5 CBA structure. The minimum number of devices required is 11 memristors, as shown in c) the state-mapping diagram with the clock number.

Product mapping depend on the total number of blocks in the system. The location of mapping is determined by the block diagram. Each column of the block diagram corresponds to two WLS in the Ex-logic circuit, whereas each row of the block diagram corresponds to a single BL. Thus, implementing the 3 × 3 mMP requires four WLS and five BLs, as shown in Figure 12b. The partial products are mapped accordingly, as shown in Figure 12c, to account for the carry position.

Next, the carry-in is propagated through the system starting from C1. As the carry-out can be calculated using the SM-three-1MAJ gate, the parallel operation to find the final sum of the partial products can be conducted through (WL2t, WL2b) and (WL4t, WL4b). One additional step is added for the reinitialization of BL4 and BL5 to reverse to the original state for the folding stage. However, this can be omitted in the fabrication process when M24 and M25 are not required by replacing the resistive switching layer of the memristors with an insulating layer. In the folding stage, the parallel implementation of the sum using APMR-three-4XOR gates is defined. Consequently, the products are stored in the Ex-logic gates located on (WL4t, WL4b), after performing 15 cycles with 20 devices. For the n-bit mMP, the latency can be found by considering each stage. The partial product mapping takes the value of n(n + 1)/2 by finding the number of the blocks. The carry cascading, including reinitialization steps, is \( \frac{1}{2}(3n^2 - 7n + 6) \), and folding is 2n/2 – 1. The total number of the devices can be determined by the number of FAs, a HA, and a single memristor. As the single HA and FM are used (addition of three memristors), the number of the FA can be found by subtracting two from the total number of the blocks, \( n^2 + n/2 \). FA block takes two APM cells, multiplying four in the total number of FAs as \( 2(n^2 + n - 4) \). The addition of the four memristors for the single-memristor operations results in \( 2n^2 + 2n - 4 \). Compared with the other proposed logic gates for n-bit mMP, the implementation shows significant improvement in the latency with RCAs, which is comparable with the carry-save adder in MultiPIM design, as shown in Table 4, due to the accelerated XOR operation using Ex-logic gate and parallelism of the IMC. Although the single device of the Ex-logic gate consists of the APM, stacking the memristor ensures high density.

Another important aspect is energy. However, the energy comparison between the different suggested logic gates could not be performed due to the difficulty of comparing the different memristors used. Still, the magnitude of the applied bias compared with \( V_{th} \) closely relates to the energy consumption and endurance of memristors. The proposed Ex-logic gate has the advantage of finding optimal \( V_a \) using active constraints with \( R_a \). The relationship between \( V_a \) and \( V_{th} \) is described in detail in the next section.

8. Mapping Analysis Method Using the Calculation of Device Stress

So far, many types of IMC logic gates have been proposed and compared in terms of latency and area[10, 26, 55–57] but the reliability of logic operations has not been fully considered. Reliability is still an open challenge for memristive IMC logic gates, and there is actively ongoing research to address related issues such as state drift and soft error. The error correction code and triple-modular redundancy are suggested as ways to decrease the error[58]. However, this requires additional initialization operations to address the errors and only applies to soft errors. Instead, addressing and minimizing their source is an efficient way to increase reliability without the extra cost of additional operations. In this regard, minimizing the memristor device stress while satisfying operational constraints must be feasible. Thus, the normalized device stress factor (κ) is introduced to quantify the device stress level depending on the types of suggested logic gates.

Regardless of the R–R and V–R logic family, the conditional operation involves the set and/or reset transition of memristors. Therefore, \( V_a \) must exceed \( V_{th} \) for the set transition, but the degree of exceeding must be minimized. Otherwise, it may
suffer from a hard breakdown (HBD). The voltage drops on selected and unselected memristors can be derived by Kirchoff's law. The degree of overvoltage can be represented by $\kappa$, defined by Equation (5).

$$\kappa = \frac{V_s - V_{th}}{V_{th}}$$  \hspace{1cm} (5)

$\kappa$ must be negative for the sustainability of unselected memristors (i.e., $V_s$ should be lower than $V_{th}$ for unselected memristor). On the other hand, a higher positive $\kappa$ to the selected memristor may ensure the successful and rapid switching of the cell, but this may sacrifice endurance. The maximum allowable $\kappa$ until the HBD depends on the device’s properties and operating conditions. For instance, the Ta$_2$O$_5$-based memristor in a 45 $\times$ 45 nm CBA structure showed set and reset switching with $0\approx+4$ V and $0\approx-2.5$ V of $V_s$ with a compliance current of 10 $\mu$A. However, it had the HBD at +5 V when the compliance current was increased to 100 $\mu$A. This means allowable $\kappa$ is 0.25 ($V_s = 5$ V, $V_{th} = 4$ V).

$V_s$ to memristors in the logic primitive circuit can be calculated by finding the common node voltage ($V_c$). The general equation for $V_c$ can be represented by Equation (6), as derived from a previous review.

$$V_c = \gamma_{us} V_{us} + \gamma_r V_s$$  \hspace{1cm} (6)

The output of the logic operation is stored in the selected memristor. The parameters for the representative primitive logic circuit are summarized in Table S1, Supporting Information. $V_1$, $V_2$, and $V_3$ are the conditional voltages that follow $|V_1| < |V_2| < |V_3|$. $R_c$ represents the effective resistance of the logic primitive circuit, which is the sum of the conductance of each memristor in the branch. $\gamma_{us}$ and $\gamma_r$ represent the ratio of the effective resistance to the unselected and selected memristor states of the primitive circuit, respectively. $V_{us}$ and versus are the driving voltage applied to unselected and selected memristors, respectively. $V_c$ can be calculated by the weighted sum of each node, and thus, the voltage drops on each memristor in the branch, $V_{M1}$, $V_{M2}$, and $V_{M3}$, can be obtained.

The $R$–$R$ logic family often requires the driving voltage in the region beyond the normal operation of the single memristor due to its connected memristive structure rather than that of a single memristor. Thus, even if the success constraints of the logic operations are satisfied for the given driving voltage applied to the memristor, that is, $V_s > V_{set}$, to change the state from HRS to LRS, the upper limit of the memristor has to be considered. For example, the PMASM-two-3NOR logic gate utilizes the reset operation of the output memristor for the NOR operation. The voltage drop and $\kappa$ across the input and output memristors can be calculated with the extracted device parameters, as shown in Figure 3 and 5. The blue and orange colors indicate set and reset transitions. The light colors represent the sustainability constraints, whereas the dark colors represent the success constraints.

The negative $\kappa$ value of the unselected cell indicates that unwanted switching can be prevented. Nonetheless, the magnitude of the negative $\kappa$ should be sufficiently high to avoid the probability of unwanted switching due to the stochastic property of memristors. The voltage applied to the memristor near $V_{th}$ can still induce stochastic switching. In this context, the mathematical relationship between the reliability and switching probability of the PMASM-two-3NOR gate in an unsafe write regime has been reported. In cases where unwanted switching has occurred, they can be switched back to the original state by initialization. However, HBD is catastrophic and can hardly be recovered.

The success constraint is used to drive the correct result of the logic gate, which may require $\kappa \geq 0$. However, satisfying this with minimal overvoltage stress depends on the detailed operation method. For instance, APMR-two-2(IMP, AND, TF) can implement six basic logic functions of IMP$_p$, IMP$_q$, AND$_p$, AND$_q$, TF$_p$$_q$, and TF$_q$$_p$ with six different applied voltages but has the highest device stress ($\kappa = 5$) for the TF$_q$$_p$ function, resulting in the HBD of the memristor, whereas IMP$_p$ and IMP$_q$ have optimal device stress ($\kappa \approx 0$). As a result, only an experimental demonstration of IMP functions could be presented. Overall, it is still quite challenging to find the optimal logic gate, ensuring the operational condition of $\kappa \geq 0$. Thus, improvement in the algorithm approach should first be undertaken.

Figure 13a shows the optimal $\kappa$ region for the logic operation. $\kappa$ should be within the $+\alpha$ region for the success constraint without the HBD and below the $-\beta$ region for the sustainability constraint without the stochastic switching. While $\alpha$ and $\beta$ vary

---

Table 4. Comparison between the proposed logic gates for n-bit mMP.

| Design | Method            | Number of memristors | Number of steps |
|--------|-------------------|----------------------|-----------------|
|        |                   | $n = 8$              | $n = 8$         |
| [50]   | FloatPIM          | $20n - 5$            | $13n^2 - 14n + 6$ | 726   |
| [52]   | APIM              | $15n^2 - 9n - 1$     | $15n^2 - 11n - 1$ | 871   |
| [51]   | Semi-Serial IMPLY | $2n^2 + n + 2$       | $[\log_2 n](10n + 2) + 4n + 2$ | 280 |
| [54]   | RIME              | $15n - 12$           | $2n^2 + 16n - 19$ | 237   |
| [53]   | MultiPIM          | $14n - 7$            | $n \log_2 n + 14n + 3$ | 139 |
| [53]   | MultiPIM-Area     | 10n                  | $n \log_2 n + 23n + 3$ | 211 |
|        | Proposed          | $2n^2 + 2n - 4$      | $2n^2 - 3n + 2$ | 114   |

© 2022 The Authors. Advanced Intelligent Systems published by Wiley-VCH GmbH
depending on the memristor structure, switching speed, and measurement temperature, the safe value can be reasonably set
to be 1.0 and 0.2, respectively, by surveying the previous
works in Section VII of Supporting Information. Based on these
criteria, the κ values for the different memristive logic circuits are
compared with the proposed Ex-logic gate, as shown in
Figure 13b. It can be understood that the suggested gate has opti-
mal operation conditions. In addition, the κ values for the four
memristors of the suggested APMR-three-4XOR comprising the
1-bit FA are also analyzed, as shown in Figure 13c. In this case,
the device overvoltage to the selected cells remained minimal
(κ = 0.0015–0.0022), and the unselected cells were safely unaf-
fected (κ < -0.49). These results demonstrate that the suggested
Ex-logic gates can be feasibly operated without involving the high
risk of HBD and stochastic switching.

9. Conclusion
Starting from the finite-state machine, the new state system is
proposed to provide an additional path to increase logic config-
urability. The system can be realized in two antiparallel bipolar
resistive switching memristors and a series resistor that provides
the XOR function in a single cycle. Then, the well-known cascading
problem of V–R logic is addressed using a novel method of
V–2R logic that connects V–R and R–R logic gates. Consequently,
APMR-two-2(XOR, IMP, RIMP) and APMR-three-4XOR gates
can be implemented with the proposed Ex-logic gates. Utilizing compatibility in a 3D CBA structure, a 1-bit FAS is dem-
onstrated with negative memristors in three steps. The ripple carry
method is adopted to account for the larger input bit length
on the n-bit full adder–subtractor. The comparison between
the n-bit full adder shows that the new implementation has
an advantage on the adder–subtractor conversion speed and opti-
mization in terms of space- and time-related costs. This advan-
tage is further verified using the n × n nMP.

Furthermore, the proposed implementation shows low device
stress per operation and a low chance of unwanted stochastic
switching. Thus, competitive memristive logic gates have been
made as outlined earlier. It is envisioned that the Ex-logic gate
can further be improved and enable efficiency in multiple appli-
cations, such as error correction and image encryption, which
will be evaluated elsewhere.

Figure 13. Proposed optimization method for the IMC logic gates based on a) the optimized κ region for the logic operation, b) κ-plot for the sustain-
ability (left) and success constraint (right) of the representative logic gates, and c) κ-mapping method for the proposed logic gate. The dark blue and
orange colors indicate the device stress of the success constraint, and light blue and orange indicate that of the sustainability constraint for the set and
reset transitions.
10. Experimental Section

The I–V curves of the memristor shown in Figure 5a were achieved from the sample fabricated using the following method. The stack of the Au/Pt/TiO2/Ti memristor was fabricated on the SiO2/Si wafer. The 100 nm silicon dioxide (SiO2) was grown by dry oxidation. The bottom electrode of 50 nm platinum (Pt) and 10 nm titanium (Ti) was deposited by electron beam evaporation. The Ti layer was used as an adhesion layer. The 35 nm TiO2 thin film for the resistive switching layer was fabricated via radio frequency (RF) sputtering with 125 W RF power in 15 mTorr reactive working pressure with 20% O2% and 80% Ar gas ambient at room temperature, using a 3° TiO target. Then, 100 nm Au and 50 nm Pt were deposited as top electrodes using electron beam evaporation. All electrodes were patterned by photolithography and followed by a lift-off process. The electrical measurement of the fabricated device was done using a semiconductor parameter analyzer (HP 4145B) to conduct the electro-forming and current-voltage sweep. The bias voltage was applied to the top electrode, and the bottom electrode was grounded. The mathematical tool “MathWorks MATLAB” was used to calculate the optimized condition for logic operation. The analog circuit simulation tool was performed by “Synopsys HSPICE,” the results of which are shown in Figure 6–8.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

This work was supported by the National Research Foundation of Korea (2021R1A3B2079882).

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

T.P. conceived the idea, wrote the manuscript, fabricated the memristor, performed the optimization equation and simulation. Y.R.K. supported the simulations. Y.R.K. and J.K. supported the analysis of the algorithm. J.W.L. participated in fabrication. C.S.H. supervised the whole research and manuscript preparation.

Data Availability Statement

Research data are not shared.

Keywords

arithmetic operations, in-memory computing, logic cascading, memristors, nonvolatile memory, sequential logic, stateful logic

Received: December 17, 2021
Revised: January 29, 2022
Published online: February 25, 2022

[1] J. von Neumann, IEEE Ann. Hist. Comput. 1993, 15, 27.
[2] G. E. Moore, IEEE Solid-State Circuits Soc. Newsletter. 2006, 11, 33.
[3] R. H. Dennard, F. H. Gaensslen, Y. U. Hwa-Nien, V. Leo Rideout, E. Bassous, R. L. Leblanc, Proc. IEEE 1999, 87, 668.
[4] W. J. Dally, Y. Turakhia, S. Han, Commun. ACM 2020, 63, 48.
[5] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, W. J. Dally, in Proc. - 2016 43rd Int. Symp. Comput. Archit. ISCA 2016, Vol. 16, IEEE, Piscataway, NJ 2016, pp. 243–254, https://doi.org/10.1109/ISCA.2016.30.
[6] Y. Turakhia, G. Bejerano, W. J. Dally, ACM SIGPLAN Not. 2018, 53, 199.
[7] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, S. Amarasinghe, in Proc. ACM SIGPLAN Conf. Program. Lang. Implement., Association for Computing Machinery (ACM), New York, NY 2013, pp. 519–530, https://doi.org/10.1145/2499370.2462176.
[8] J. T. Pawłowski, in 2011 IEEE Hot Chips 23 Symp. HCS 2011, IEEE, Piscataway, NJ 2016, pp. 1–24, https://doi.org/10.1109/HOTCHIPS.2011.7477494.
[9] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, Y. J. Kim, Y. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, J. H. Cho, K. H. Kwon, M. J. Kim, J. Lee, K. W. Park, B. Chung, S. Hong, in Dig. Tech. Pap. - IEEE Int. Solid-State Circuits Conf., Vol. 57, IEEE, Piscataway, NJ 2014, pp. 432–433, https://doi.org/10.1109/ISSCC.2014.6757501.
[10] D. Ielmini, H. S. P. Wong, Nat. Electron. 2018, 1, 333.
[11] K. J. Yoon, M. H. Lee, G. H. Kim, S. J. Song, J. Y. Seok, S. Han, J. H. Yoon, K. M. Kim, C. S. Hwang, Nanotechnology 2012, 23, 185202.
[12] K. J. Yoon, S. J. Song, J. Y. Seok, J. H. Yoon, T. H. Park, D. E. Kwon, C. S. Hwang, Nanoscale 2014, 6, 2161.
[13] T. Kim, H. Son, I. Kim, J. Kim, S. Lee, J. K. Park, J. Y. Kwak, J. Park, Y. J. Jeong, Sci. Rep. 2020, 10, 1.
[14] N. Du, H. Schmidt, I. Polian, Nano Mater. Sci. 2021, 3, 186.
[15] E. Lehtonen, M. Laiho, in 2009 IEEE/ACM Int. Symp. Nanoscale Archit. NANOCARCH 2009, IEEE, Piscataway, NJ 2009, pp. 33–36, https://doi.org/10.1109/NANOCARCH2009.5226356.
[16] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, R. S. Williams, Nature 2010, 464, 873.
[17] E. Linn, R. Rosezin, S. Tappertzhofen, U. Böttger, R. Waser, Nanotechnology 2012, 23, 305205.
[18] T. You, Y. Shuai, W. Luo, N. Du, D. Bürger, I. Skorupa, R. Hübner, S. Henker, C. Mayr, R. Schüffny, T. Nikolajick, O. G. Schmidt, H. Schmidt, Adv. Funct. Mater 2014, 24, 3357.
[19] N. Xu, K. J. Yoon, K. M. Kim, L. Fang, C. S. Hwang, Adv. Electron. Mater. 2018, 4, 1.
[20] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. C. Friedman, A. Kolodny, U. C. Weiser, IEEE Trans. Circuits Syst. II Express Briefs 2014, 61, 895.
[21] P. Huang, J. Kang, Y. Zhao, S. Chen, R. Han, Z. Zhou, Z. Chen, W. Ma, M. Li, L. Liu, X. Lai, Adv. Mater. 2016, 28, 9758.
[22] G. C. Adam, B. D. Hoskins, M. Prezioso, D. B. Strukov, Nano Res. 2016, 9, 3914.
[23] N. Xu, T. G. Park, H. J. Kim, X. Shao, K. J. Yoon, T. H. Park, L. Fang, K. M. Kim, C. S. Hwang, Adv. Intell. Syst. 2020, 2, 1900082.
[24] F. Wei, X. Cui, X. Cui, IEEE J. Electron Devices Soc. 2020, 8, 57.
[25] B. Hoffer, V. Rana, S. Menzel, R. Waser, S. Kvatinsky, IEEE Trans. Electron Devices 2020, 67, 3115.
[26] N. Xu, T. Park, K. J. Yoon, C. S. Hwang, Phys. Status Solidi RRL 2021, 15, 1.
[27] L. Luo, Z. Dong, X. Hu, L. Wang, S. Duan, AEU Int. J. Electron. Commun. 2021, 129, 153542.
[28] G. Papandroulidakis, I. Vourkas, N. Vasileiadis, G. C. Sirakoulis, IEEE Trans. Circuits Syst. II Express Briefs 2014, 61, 972.
[29] J. Reuben, J. Low Power Electron. Appl. 2020, 10, 1.
[30] S. G. Rohani, N. Taherinejad, in Can. Conf. Electr. Comput. Eng., IEEE, Piscataway, NJ 2017, pp. 17–20, https://doi.org/10.1109/CCECE.2017.7946813.
