Memristive Stateful Logic for Edge Boolean Computers

Young Seok Kim, Myeong Won Son, and Kyung Min Kim*

Memristive stateful logic enables complete in-memory computing, allowing low-power and low-cost Boolean computing. Its characteristics are consistent with the requirements of edge computing devices, which will occupy a considerable segment in the coming Internet of things (IoT) era. In this review, recent developments in stateful logic technology are intensively explored. The topics include a summary of the evolution of stateful logic gates and their cascading strategies for Boolean computing. Also, array-level data manipulation is discussed, and its role in realizing massive computation inside the memory. Finally, the logic operation error issue is discussed, with feasible solutions.

1. Introduction

Scaling in semiconductor processing over the last several decades has led to a drastic revolution in performance. Moore’s law,\(^1\) predicting that transistor density doubles every two years for logic devices, and Hwang’s law,\(^2\) predicting that memory density doubles every year for memory devices, provide clear indications of just how fast their evolution has been. However, the long-expected physical limits of scaling have finally arrived: Moore’s law has been recently abandoned.\(^3\) This does not necessarily represent the end of technological evolution, but the beginning of a new era called the more-than-Moore era.\(^4\)

In the coming Internet of things (IoT) era, physical objects with built-in sensors and software exchange data with other devices and systems through the Internet.\(^5-7\) Consequently, the computing units are moving from clouds or servers to edges or users.\(^8-10\) As a result, the first virtue of computing is no longer performance, which has driven the technology revolution, but rather energy consumption. Accordingly, the next evolution will shift the computing paradigm away from the von Neumann architecture\(^11\), which has governed for over 70 years because of its powerful performance and personalization capability. But because of its high energy consumption, a consequence of its intrinsic operation principle, this architecture can no longer be used in edge devices.

Processing-in-memory (PIM) (or in-memory computing) is one of the candidate technologies for the coming paradigm shift. It is actually not a new concept; it was proposed and developed in the 1960s.\(^12-14\) Many studies have successfully demonstrated PIM feasibility using CMOS (complementary metal-oxide-semiconductor)-based technology. However, its debut in the market has failed so far because of unsatisfactory performance, low productivity, and higher cost than the von Neumann computing devices. Within the field of CMOS-based technology, PIM does not seem likely to replace the von Neumann computing architecture yet.

This situation may change with the advent of a new electrical latching element, the memristor.\(^15-17\) The memristor was introduced as a fundamental circuit element (although its definition is arguable yet)\(^18-20\) in which an applied voltage leads to a reversible conductance change. Although initial memory or storage applications have received attention, they are still struggling to debut in the market because their performance does not yet exceed that of conventional devices.

Meanwhile, another interesting memristive device was proposed, the so-called stateful logic, by Borghetti et al. in 2010.\(^21\) The stateful logic device can perform a Boolean logic operation inside memory, allowing complete PIM operation. However, despite its high potential, its practical performance seems unlikely to surpass that of CMOS-based devices. Accordingly, there may be no place for the device in the conventional market. However, the expansion of the IoT era will require a cheaper and slimmer computing device, and the characteristics of the stateful logic device can fulfill these requirements.\(^22,23\)

Thus, it may become highly feasible as a technology in future electronics.

In this review, the stateful logic technology is intensively explored, from basic operating principles to system-level issues. Section 1 discusses how the logic gates have evolved in chronological order. It also compares the kinds of stateful logic gates in various memristive systems and their characteristics. In Section 2, logic cascading is discussed using full adder execution as an example. Efforts to improve the logic cascading efficiency are shown in the full adder execution challenge. In Section 3, array-level computing issues are discussed, as well as requirements for better data traffic control and computing efficiency. In Section 4, possible operating error issues are discussed in relation to device realization, followed by their solution.
2. Evolution of Stateful Logic Gates

Stateful logic technology development began in earnest with the experimental unveiling of an IMP gate. Multiple studies have since developed various stateful logic gates. Their evolution has been motivated by efforts to overcome various issues. Understanding the evolutionary history is worthwhile because it can helpfully explain the current status of the technology.

2.1. Stateful Logic Gates of Bipolar Memristors

Most memristive stateful logic studies have used a bipolar-type memristor (bipolar resistive switching, BRS), primarily because of its proven high performance. In general, bipolar-type memristors are faster and more durable than unipolar-type (unipolar resistive switching, URS) or mixed-type memristors, making them well suited for logic devices.[24,25] Also, their opposite set and reset switching polarity permits a sufficient operating voltage margin during the gating operation, which is also advantageous.

Figure 1 summarizes the evolution history of stateful logic gates. Note that Figure 1 includes only a basic gate unit, executable by one voltage clock. It shows 14 Boolean logic gates, excluding TRUE and FALSE gates, which are simply achievable by set and reset switching, respectively. The following abbreviated forms are used to represent the logic gates—statement: BUFFER; inversion: NOT; conjunction: AND; disjunction: OR; alternative denial: NAND; joint denial: NOR; material implication: IMP; material nonimplication: NIMP; converse implication: CIMP; converse nonimplication: CNIMP; exclusive disjunction: XOR; and biconditional: XNOR. The open, transparent, and closed symbols refer to theoretical studies, single gate-level experimental studies, and cascading-level experimental demonstration studies, respectively. The blue and red colors refer to a SET-gate family (SGF) and RESET-gate family (RGF), respectively; the SGF realizes the logic operation by conditional set switching, and the RGF does it by reset switching. In most cases, the high-resistance state (HRS) is assigned to 0, and the low-resistance state (LRS) is 1. The underbar below the symbol indicates an exceptional case where the HRS and LRS were inversely assigned to 1 and 0, respectively.

Also, stateful logic can be implemented in various configurations. Figure 2 shows the representative configurations for executing the stateful logic gates. Figure 2a shows the basic configuration comprising two memristors, so it is called a “two-memristor configuration." In this case, when the two input gates are executed, the output would be overwritten on one of the inputs, so the data are destructive. Figure 2b shows a three-memristor configuration, where the output cell is separated from the two input cells, allowing nondestructive gate operation. Figure 2c shows a multiple-memristor configuration. It is the extension of the three-memristor gate that can perform logics of more than two inputs. Figure 2d,e shows a 3D crossbar array architecture integrated with a vertical structure (Fig. 2d) or a stacked structure (Figure 2e), allowing stateful logic along the vertical direction. Figure 2f shows a multiple-memristive system in a single cell that can perform stateful logic in a single cell.

The initial concept of stateful logic and the first stateful logic operation were theoretically proposed by G. Snider in 2005, who suggested BUFFER, NOT, and AND gates using bipolar memristors.[26] The BUFFER and NOT gates were possible using two parallel memristors, two voltage sources, and a resistor, as shown in Figure 2a. A diode was additionally required for the
NOT gate. The AND gate was possible by introducing a resistor and inducing the voltage divider effect between the resistor and memristors. It may utilize both the three-memristor (Figure 2b) and multiple-memristor configurations (Figure 2c) depending on the number of inputs. In 2008, Kuekes proposed the material implication (IMP) gate. \( (q' = 1 \text{MP} (p, 0)) \) The IMP gate utilized the two-memristor configuration in Figure 2a, requiring two voltage sources, a set voltage \( V_{\text{SET}} \) and a conditioning voltage \( V_{\text{COND}} \) and an additional resistor in between the bit line and ground source to induce the voltage divider effect. Then, the \( V_{\text{COND}} \) applied to the output cell \( M_{\text{O}} \) may increase the bit-line potential only if the state of \( M_{\text{O}} \) is the LRS (state 1) and inhibit the set switching of the input cell \( M_{\text{IN}} \), resulting in the IMP gate. Afterward, in 2010, Borghetti et al. experimentally proved the IMP gate operations using a TiO2-based memristor.\(^{[21]} \) The IMP gate could also be used for a NOT gate by setting one of the inputs to “0.” \( (q = 1 \text{MP} (p, 0) = \text{NOT} p) \) The importance of the work was not just the experimental proof of the IMP gate, but because complete stateful logic was first considered for Boolean computing. They showed that a NAND gate, a universal gate, could be synthesized by cascading the reset switching once and IMP gate operation twice over three memristors. \( (s' = 1 \text{MP} (q, 1 \text{MP} (p, s))) \) where \( s = 0 \). Moreover, they showed that all Boolean gates were possible in the same manner.

Although the leading studies were innovative, various issues arose, and subsequent studies were needed to deal with those issues. The first issue involved improving the efficiency of stateful logic computing. Although the completeness of stateful logic was achieved, computing-efficiency issues were not yet seriously considered. Various studies proposed new gates that offered a more efficient way of computation compared to the original IMP-gate-only stateful logic. The second issue was loss of input data. After the IMP gate operation, the input data were replaced with the output. Thus, the input could not be used multiple times, which can be problematic for logic cascading. Although copying the input data before the gate operation would provide a solution, it would also harm efficiency.

Considering the seriousness of the data-loss issue, the high-fan-in NOR gate by Kim et al. in 2011 was another notable study.\(^{[28–30]} \) In the study, they first introduced the three-memristor configuration shown in Figure 2b for a NOR gate and separated two input cells and one output cell. Then, by applying \( V_{\text{COND}} \) to input cells similarly to the IMP gate, the NOR gate’s output could be recorded to a dedicated output cell. The NOR gate was another universal gate that could ensure the completeness of the Boolean logic and was more efficient than the three-step NAND gate synthesized from the IMP gate. Also, the high-fan-in NOR gate was possible with the multiple-memristor configuration shown in Figure 2c. After this, to distinguish the gate configuration, the gates were labeled with a digit referring to the number of cells used. For example, the original IMP gate became a 2IMP gate, and the two-input NOR gate became 3NOR.

Up to this point, all of the gates were achieved using a conditional set switching of the output cell, that is, an SGF. Next, gates were discovered that could be operated using a conditional reset switching, that is, an RGF; these are underlined to distinguish them from the SGF. Kim et al. in 2011 theoretically proposed a 2AND gate and extended it to a 3OR gate.\(^{[28–29]} \) Lehtonen et al. in 2012 did a 2CIMP,\(^{[31]} \) Zhu et al. in 2013 did 2AND,\(^{[32]} \) and Kvatinisky et al. in 2014 did a 2NOT and 3NOR gates, respectively.\(^{[33]} \)

Even though the final outputs can be the same, the efficiency of the SGF and RGF can be different, situationally. For example, if the output cell is occupied by “1” already, the SGF requires a reset-initialization step before the gate operation, whereas the RGF does not, making the RGF more efficient. Such a situation can frequently happen during logic cascading. By combining

---

**Figure 2.** Configuration of various stateful logic gates. a) Two-memristor configuration; one is for input only, and the other one is for the input and output. One of the inputs is destructive. b) Three-memristor configuration; two are for inputs, and the other one is for the output. c) Multiple-memristor configuration; it can execute more than three input gates such as 4CARRY and 5SUM. d) A vertical configuration of the 3D structure. e) Double-layer stacked structure allowing both serial and parallel gate operation. f) A dual-bit (including BRS and URS bits) configuration for a single-cell stateful logic.
them in optimal ways, computational efficiency can be improved.\textsuperscript{[34]} Also, some of the RGFs are more robust against variations in the switching voltage of the device. For example, the 3OR gate of the RGF can allow a broader operating voltage range (OVR) than the 3OR gate of the SGF.\textsuperscript{[15]} This can be understood as follows. The output of the OR gates should be “1” for inputs of “01,” “10,” and “11,” and “0” for an input of “00.” Therefore “1” is the majority output, and thus it is more comfortable to start from the initial state “1” and reset the minority output. A more detailed discussion regarding the practical issues will be discussed in Section 4.

After stateful logic technology spread to the research community, more diverse gates and experimental demonstrations were widely reported. Lehtonen et al. in 2014 showed that a 2OR gate was possible instead of the 2IMP gate just by applying negative voltages to the input cells.\textsuperscript{[16]} Also, Chen et al. in 2015 experimentally proved the 3NOR gate,\textsuperscript{[37]} which had been theoretically proposed by Kim et al. in 2011. Li et al. in 2015 experimentally confirmed the 2NOT gate.\textsuperscript{[40]} Xie et al. in 2015 proposed the 2BUFFER, 2NOT, 3AND, and 3NAND gates.\textsuperscript{[39]} Here, they used a trick, assigning the HRS to 1. Assigning the HRS to 0 could convert the 3OR and 3NOR gates to the 3AND and 3NAND gates, respectively. In 2016, Huang et al. experimentally confirmed the 3AND and 3NAND gate operations.\textsuperscript{[40]} In the same year, Wald et al. developed the 3NAND gate by changing the applied voltages of the 3NOR gate.\textsuperscript{[47]} In 2017, Chen et al. theoretically proposed the 2CNIMP gate of the RGF.\textsuperscript{[42]} Also, Li et al. experimentally showed the 2BUFFER and 2BUFFER gates and theoretically proposed 2NOT and 2IMP gates in a 3D vertical structure, shown in Figure 2d.\textsuperscript{[43]}

Afterward, studies attempted to understand stateful logic systemically. In 2018, Sun et al. introduced a neural network concept to stateful logic.\textsuperscript{[44]} They calculated the voltage potential at the common node using Kirchhoff’s law and obtained the voltage across the output cell. In this way, the set switching condition of the output cell can be defined as the sum of the conductance of each cell multiplied by the voltages applied to the cell. It is consistent with the operating principle of a neural network where the output is defined as the sum of each input multiplied by each weight. Consequently, they showed 14 Boolean logic gates were theoretically possible with the two- or three-memristor configurations in Figure 2a,b, except the XOR and XNOR gates, and among them, they demonstrated 3AND and 3NOR gates experimentally. Especially noteworthy, they developed the novel 4CARRY and 5SUM gates, which could dramatically increase the full adder efficiency with the multiple-memristor configuration in Figure 2c. The meaning of those gates will be discussed in detail in Section 3. In 2018, Jang et al. demonstrated 3NOR and 2NOT gates using a flexible device.\textsuperscript{[45]} Gupta et al. in 2018 developed the 3NAND and 3OR gates by tuning the operating voltages of a 3NOR gate.\textsuperscript{[46]}

In 2019, Kim et al., Shen et al., and Cheng et al. experimentally confirmed their operation.\textsuperscript{[47–49]} The 3CNIMP, 3NIMP, 3OR, and 2OR gates of the SGF and the 2CNIMP and 2AND gates of the RGF were experimentally proven for the first time. In 2020, Hoffer et al. experimentally showed the 3OR and 3NIMP gates without a load resistor,\textsuperscript{[50]} which are more stable than the 3NOR gate shown in 2014. In the same year, Kim et al. showed that most of the Boolean gate operations were also possible in the RGF.\textsuperscript{[15]}

2.2. Stateful Logic Gates of Nonbipolar Memristors

Many studies also reported stateful logic gates from nonbipolar-type memristors, such as a unipolar-type memristor, and mixed-type memristors. Once the memristor types change, the set and reset switching conditions become different, making each of the stateful logic techniques distinctive.

2.2.1. Unipolar Memristor

In 2011, Sun et al. reported that IMP operation was possible in an Au/SrTiO3/Pt unipolar memristor device.\textsuperscript{[51]} This was the first report of stateful logic using a unipolar memristor. The authors claimed that the high on/off ratio of the device, which is one of the distinguishable features of the unipolar memristor, provided an advantage in the logic operation. However, unipolar-type memristors, where the set voltage ($V_{\text{SET}}$) amplitude is higher than the reset voltage ($V_{\text{RES}}$) amplitude at the same bias polarity, are less suitable for stateful logic operation because of the narrow voltage margin between $V_{\text{SET}}$ and $V_{\text{RES}}$. For example, the 2IMP gate operation required a $V_{\text{COND}}$ on input cells that had an amplitude smaller than the $V_{\text{SET}}$ and higher than 0. Unlike a bipolar-type memristor, the application of $V_{\text{COND}}$ can reset-switch the input cell. Furthermore, there is a possibility that the output cell can be spontaneously reset-switched after the conditional set switching. Accordingly, it would require a remarkable breakthrough for the unipolar memristor to be used in a stateful logic device.

In 2015, Zhou et al. shed light on the situation.\textsuperscript{[52]} They reported a stateful logic using a TaN/SiO2/Si unipolar-type memristor that inherently showed a higher $V_{\text{RES}}$ than $V_{\text{SET}}$. In this specific device, the application of $V_{\text{COND}}$ did not cause the unintended reset switching, making the stateful logic viable. Once its reliability, endurance, and reproducibility are confirmed, the unipolar memristor may be feasible.

2.2.2. Multiple Memristive System

Multiple memristive systems involve more than two distinct memristive components during operation. Those memristive components can be allocated spatially at a distinct area or presented intrinsically at a single device. Due to the interrelation between the memristive components, their set and reset operations are more complicated than those of a single memristive system. This may provide a greater number of potential combinational operations, making them more suitable for stateful logic application.

One example is the complementary resistive switch (CRS)-type memristor, where two bipolar-type memristors are located in series but with opposite directions.\textsuperscript{[53]} As a result, it can intrinsically store two bits per cell. Although stateful logic has to access more than two cells simultaneously, the CRS cell has the potential to provide the stateful logic operation itself in one cell. However, as the two memristor components are serially connected and not independent of each other, their logic...
operation is not fluent. Instead, more than two CRS cells can perform the gate operation like a bipolar memristor. Unlike the bipolar memristor, data reading in CRS devices is destructive. They store the data 0 and 1 as ON/OFF and OFF/ON combinations, which are not distinguishable by nondestructive reading. This makes their combinational logic operation in multiple cells quite complicated. In 2014, Yang et al. first reported a stateful logic using the CRS memristor.\[34\] In that study, they demonstrated the IMP gate in two CRS cells for three consequent operations, including converting the unveiled OFF/ON state to the readable LRS and reconverting them after gate execution. Later the authors reported that NAND-AND and NOR-OR gates can be achieved in three clocks.\[55\] Those gates were extended to the NOR, NOT, NAND, and NIMP gates. Even though the CRS has a two-bit-per-cell capability, it is not directly readable. Therefore, there is no advantage to adopting a CRS device for stateful logic.

The next example is using a spatially separated structure to compose multiple BRSS. In 2015, Balatti et al. experimentally demonstrated 2IMP, 2NOT of the RGF and 2AND, 2BUFFER gates of the SGF from a serial configuration of memristors instead of a parallel configuration.\[36,37\] For this serial logic operation, they suggested a double-layer stacking layout, where the cells can be serially stacked. In 2020, Xu et al. proposed that an antiparallel configuration of bipolar-type memristors can improve computing efficiency.\[34\] They utilized a TiO2-based memristor for the demonstration and confirmed that 2IMP and 2AND gates were possible. The antiparallel bipolar memristors can be integrated in two sections of the in-plane array, or two layers, by stacking two antiparallel arrays. Figure 2e shows the stacked structure, which can embrace both methods. In any case, their integration can be challenging.

The last example is introducing a mix of BRSS and URS. In 2017, Lebedh et al. reported stateful logic using a heterogeneous memristive system.\[58\] They used a unipolar memristor for the output and bipolar memristors for the inputs and demonstrated the XNOR gate. As a follow-up study, in 2018, Xu et al. showed the XOR gate was possible by simply changing the state designation method.\[59\] Though they were fanciful, it seemed unlikely to be feasible to integrate the complicated array configuration, with different types of memristors for each column. In 2019, Kim et al. utilized a dual-bit memristor that possessed both unipolar and bipolar switching capability in a TiO2-based memristor.\[60\] Figure 2f shows the gate unit configuration. The device could perform the 1AND, 1OR, and 1IMP gate operations in a single cell. However, the uncertain reliability of the device makes it impractical without significant innovation.

3. Competition for Full Adder Execution

All of the Boolean gate operations are variously possible in stateful logic. The next step is cascading the gates for computing. Many studies have demonstrated various logic circuits, such as an adder [34,35,37,40,44,47,49,55,57,60–69] multiplier,\[37,55,70–72\] and sequential logic circuits.\[29,80,81\] These operations can be used to implement image processing,\[82\] neural networks,\[83–85\] and hyperdimensional computing\[86,87\] in in-memory computing.

In stateful logic, the gate cascading is performed in a time dimension, which is the major difference compared to a conventional CMOS-based logic device, where it is done in a spatial dimension. Figure 3 shows the difference in logic cascading for CMOS logic and stateful logic, taking the full adder execution as an example.\[35,87\] Stateful logic requires a much more compact physical area but longer time. As a result, the stateful logic device has advantages in density and cost but disadvantages in speed. This suggests that stateful logic devices are most suitable for IoT and edge applications where low cost is required.

Nonetheless, further improvement in the computing efficiency of stateful logic technology is crucial. The spatiotemporal efficiency is related to the spatial and temporal usage for a given calculation. Accordingly, we define the spatiotemporal cost for full adder execution (STCPFA) by counting the number of required cells (Ncell) and clocks (Nclock) and multiplying them (STCPFA = Ncell × Nclock). Also, the energy efficiency is related to actual energy consumption during the calculation. To estimate the energy consumption, we assumed that only the active cells per step are consuming energy, whereas other resting cells are not. Also, we assumed the energy consumption per active cell is constant on average. Accordingly, we define the energy cost required for full adder execution (ECFA) by counting only the

Figure 3. Comparison of a full adder with CMOS logic and NOR-based stateful logic. a) A representative full-adder layout. In CMOS logic, an operation is implemented by properly arranging gates composed of transistors in the space. The operation requires only one clock. b) In stateful logic, the gate unit is much more compact; it occupies only 1/14 of the area compared to CMOS logic (left panel). The timing diagram (right panel) shows a 9 O’clock sequence of voltage amplitudes to execute the desired gate operation.
actively participating cells during the operating steps. Many studies have demonstrated efficiency improvements in their methods using the adder (either a half-adder or a full adder), which is the most representative operation in computing. The following shows how the adder operation has evolved.

Table 1 summarizes various proposed ways of adder execution via various gates. The first report is found in Lehtonen et al. in 2009, where they proposed the full adder operation using only 2IMP and FALSE gates. It required 39 clocks for sum and 48 clocks for carry operations over 8 cells, resulting in an STCFA of 696 and an ECFA of 139. In 2015, Chen et al. theoretically proposed that the full adder execution can be completed with an STCFA of 160 (10 clocks and 16 cells) and an ECFA of 21 using their 3NOR gate and experimentally confirmed it. Next, Balatti et al. demonstrated a full adder using two series-connected memristors with an STCFA of 77 (7 clocks over 11 cells) and an ECFA of 50. In this case, the devices were stacked in three layers. In 2016, Huang et al. demonstrated a full adder by cascading 3NAND and 3AND gates requiring an STCFA of 90 (10 clocks and 9 cells) and an ECFA of 30. In 2018, Sun et al. proposed an ultimate way of executing the full adder. They expanded the multiple cell stateful logic idea to a stateful neural network and realized 4CARRAY and 5SUM gates executable in one clock. In that way, they achieved an STCFA of 8 (2 clocks and 5 cells) and an ECFA of 8. In 2019, Kim et al. executed the full adder with 26 clocks and only 3 cells from the dual-bit memristor and its STCFA and ECFA values are 78 and 28, respectively. In 2020, Xu et al. cascaded logic gates for the full adder operation in 14 clocks over 9 cells, which produces a CFA of 126 and an ECFA of 28.

By the competitive validating of the full adder execution, there was a significant breakthrough in stateful logic and triggered other researchers to consider the next stage of the technology, the realization of a computing device. At the same time, practical issues arose regarding data manipulation in a large-scale array and the reliability of the operation, which will be discussed subsequently.

### 4. Data Manipulation for Logic Cascading and Efficient Computing

In conventional computing, a cache memory next to a processing unit loads data from main memory or storage, and the processing unit recalls the data from the cache and returns the processed data to it via a bus input/output (I/O) system. In that way, the data traffic can be well organized. In stateful logic, such a data manipulation strategy is more critical because all of the computing sequences must be executed inside the memory. Otherwise, the data can be overwritten, or their address will be lost. Various studies showed efficient sequential gate-cascading methods in one dimension, which consume little area but cause too much delay.

Parallel computing in a 2D array can increase the computing efficiency by executing multiple gate operations simultaneously. There are two types of parallel computing: in-memory parallel computing and near-memory parallel computing utilizing an additional cache memory. For parallel computing, the operands must be aligned before the operation, making the data manipulation more important. In this section, we will discuss the parallel operation of stateful logic and the data manipulation strategy for complicated stateful logic computing.

#### 4.1. Parallel Operation for More Efficient Computing

##### 4.1.1. Bitwise Parallelism

In stateful logic, gate units share one of the bit lines or word lines, and the other line is biased differently. In the array, the gate units can be allocated over multiple lines in parallel, aligning the input and output cells along the perpendicular direction. Figure 4a shows the logic gate configuration for parallel operation, taking the 3NOR gate as an example, where $A_n$ and $B_n$ are two inputs and $O_n$ are outputs. Lehtonen et al. in 2012 first proposed parallel operation using a 2IMP gate. We classify this as

| Year | References | Cascaded gates or functions | Basic gates | Cells [rows x columns (x layers)] | Steps | STCFA | ECFA |
|------|------------|-----------------------------|-------------|-----------------------------------|-------|-------|-------|
| 2009 | E. Lehtonen et al. | AND, OR, NOT | 2IMP, FALSE | 8 (1 x 8) | 87 | 696 | 139 |
| 2014 | S. Kvatinsky et al. | XOR, AND, OR | 2IMP, FALSE | 6 (1 x 6) | 29 | 174 | 51 |
| 2015 | M. Teimoori et al. | XOR, OR, AND | 2IMP, FALSE | 6 (1 x 6) | 20 | 120 | 38 |
| 2016 | B. Chen et al. | NOR, NOT | 3NOR, 2NOT | 16 (1 x 16) | 10 | 160 | 21 |
| 2017 | R. Li et al. | XOR, OR, AND | 2IMP, 3AND | 11 (1 x 4 x 3) | 7 | 77 | 50 |
| 2018 | B. N. Talati et al. | NOR, NOT | 3NOR, 2NOT | 10 (4 x 9) | 13 | 130 | 39 |
| 2019 | G. C. Adam et al. | NAND, NOT | 2IMP, FALSE | 6 (2 x 2 x 2) | 32 | 210 | 51 |
| 2020 | P. Huang et al. | NAND, AND | 3NAND, 3AND | 9 (1 x 9) | 10 | 90 | 30 |
bitwise parallelism. By combining the same gate operations and executing them at once, bitwise parallelism can reduce the total computing cost. Multiple studies have demonstrated the viability of parallel operation using various logic gates, and executing various computations.\[35,37,39,48,49,55,62,97,98\]

The most popular example of a bitwise parallelism application is the multibit adder operation. In the multibit adder operation, one also needs to consider data transfer between the parallel lines. For example, the carry-out data of the \(n\)th line should convey to the carry-in of the \((n+1)\)th line. In 2014, Kvatinisky et al. proposed a parallel eight-bit adder operation utilizing the well-known 2IMP gate.\[97\] To realize data transfer between the lines, they connected the terminals of the two bit lines using a transistor, as shown in Figure 4b. When the transistor was turned on, the two parallel lines were connected, and they acted as one double-length line. In this way, data could be easily moved along the double-length line. By combining the bitwise parallelism and the transistor connector configurations, they showed that parallel operation could reduce the total number of clocks from 29\(N\) for the complete serial operation to 5\(N+18\), where \(N\) is the number of bits.

In 2019, Siemon et al. made use of a more complex structure.\[62\] Because the cells related to the carry bits are connected to respective voltage sources, and the transistors are connected to the parallel lines, the carry-out bit can be directly transferred to the target position where the carry bits are aligned, without having to move data around. Using ORNOR, 2IMP, and FALSE gates, this \(N\)-bit adder required 2\(N+15\) clocks.

Later, more convenient data transfer methods were reported that utilized both the horizontal and vertical gate units without needing the additional circuit features. Cheng et al. in 2019\[49\] presented an efficient \(N\)-bit adder operation based on 2IMP, 2OR, 3NOR, and 3OR gates, and Kim et al. in 2020\[35\] reported one based on 2BUFFER, 2NOT, and 3NOR gates. In both cases, the calculation and alignment steps of the carry bits were performed sequentially, while all other processes were computed in parallel. Cheng et al. used 2OR gates to convey the carry bit and executed an \(N\)-bit adder in 6\(N+6\) clocks. Meanwhile, Kim et al. used 2NOT and 2BUFFER gates to convey the carry and executed an \(N\)-bit adder in 5\(N+8\) clocks.

Recently, bitwise parallel operation computing has been used for hyperdimensional computing\[46\] which deals with long binary vectors. The limitation of bitwise parallelism, however, is that only the same operation is available in parallel within a restricted area, where the data are well ordered, to avoid interference. In other words, in bitwise parallelism, different gates cannot be implemented simultaneously, and even the same gates in multiple regions cannot be used at the same time.

4.1.2. Blockwise Parallelism

To improve the computing efficiency of stateful logic, it is crucial to increase the number of actively operating cells. Dividing the entire array into multiple subarray units with individual controller circuits and systematically connecting them together will provide another degree of freedom in parallelism.\[28,29,83,99–102\] Figure 4c shows the basic concept of blockwise parallelism. This concept is similar to that of a multicore processor, where
the processing unit is composed of multiple small processing cores. We classify this as blockwise parallelism.

In each of the subarray units, each logic gate can be independently executable. In between the subarrays, a transistor may be present to connect them. When the transistor turns on, the subarrays can be merged to constitute a new subarray. This idea was first proposed by Kim et al. in 2011.\[28,29\] By introducing blockwise parallelism, they showed that a function \( f = pq + mn \) can be executed in six clocks by simultaneously processing the preceding two NAND gates. Previously it was performed in nine clocks with three consecutive NAND gates. Although the advantage of bitwise parallelism was obvious, there was no detailed methodology for systematically dividing the sub-arrays.

In 2014, Lehtonen et al. suggested systematically dividing a large memristor crossbar array into regular subarrays of \( 8 \times 8 \) size, the size of 1 byte.\[99\] They demonstrated the effectiveness of blockwise parallelism with circular shift, content-addressable memory, parity, and the hamming weight of a binary vector.\[99,100\]

Such blockwise parallelism is helpful for executing multi-input adders.\[101,102\] Also, it can be applied to perform vector–matrix multiplication,\[85\] the building block of neural networks.

### 4.2. Data Manipulation in a 2D Array

In conventional computing, the data are distributed over the memory array and then accessed by an I/O device, based on their physical address recorded on the address bus circuitry. Consequently, the physical location of the data does not matter. In stateful logic, the physical location of the data is crucial; they must be allocated exactly to the appointed row and column. Therefore, data manipulation is needed before gate operations can be implemented to relocate or redistribute data in the memristor array.

One of the approaches for dealing with the data manipulation issue adopts a passing transistor or reconfigurable interconnects, which can change the physical connectivity of the data.\[102,112\] The passing transistors can provide a direct pathway between any cells in the array. Similarly, reconfigurable interconnects can connect any cells along the different rows between memristor blocks, as shown in Figure 4d. To integrate the transistors and interconnections, these methods require a larger space than a memristor array, which reduces memory density.

Another data modulation approach utilizes a stateful COPY operation.\[104,107\] The stateful logic can be executed along one of the vertical or horizontal directions. By combining two BUFFER or NOT gate operations along two orthogonal directions, it becomes possible to copy any bits to any location. Figure 4e shows the two-step data relocation method. It requires one unoccupied bit between the initial and final data locations. However, as the complexity of computing and data density increase, the in-memory data COPY operation becomes more complicated. For example, if the temporal bit is already occupied, the data must detour to another empty bit, at least one or more times. Some studies have tried to use an optimized searching algorithm to find the proper location for the output before the operation to reduce the copy operations.\[103,105,106\] However, this cannot completely eliminate the data traffic problem when the data are distributed sporadically.

To make the data manipulation more systemic and minimize the data traffic issue, Kim et al. in 2020 proposed an in-memory bus architecture.\[35\] In their method, the entire memory area is divided into a memory area for the main storage, a logic area dedicated to the computing operation, and a data bus area for data relocation. Figure 4f shows the proposed array architecture. By dividing the roles of each section, not only can the data manipulation be well organized, but also the size of the controller circuit can be minimized. For example, the main storage area only requires programming voltage and buffer gate voltage sources. The buffer area only requires a buffer gate voltage source. Only the logic area requires multiple voltage sources for stateful logic gate operation.

### 5. Stochastic Errors and Their Correction

Memristive devices possess device-to-device or cycle-to-cycle switching voltage or resistance variations, which mainly originate from the stochastic nature of the switching process.\[113\] Figure 5a shows a typical I–V set of switching cycles, displaying the switching voltage variation and the resistance variation. Figure 5b shows the histogram of the switching voltages. Such variations are issues for all memristive devices, even for the memory application. Unfortunately, in a stateful logic device, such variations are fatal as they cause a malfunction of the logic operation.\[35,114\] It becomes even more problematic when multiple cells have to be accessed simultaneously. Simply, assuming the success rate of any operation per cell is \( r < 1 \), the logic operation success rate becomes the \( n \) power of \( r \), where \( n \) is the number of cells involved during the logic operation.

Furthermore, resistance variation makes the problem more serious. In the memory operation, resistance variation is only needed for the marginal reading of the 1 and 0 data. In the logic operation, the resistance variation can affect the node voltage of the cell, which is determined by the voltage divider effect between cells.

In previous sections, the proposed logic gates and their cascading assumed an ideal operation without concern for the possibility of errors, which is impractical. This section will discuss error issues that occur with stateful logic and efforts to solve the issues.

#### 5.1. Quantifying the Tolerance of the Logic Gate

In stateful logic, most of the basic two-inputs Boolean logic gates, and even more, three-inputs gates such as a carry gate and sum gate can be executed in a single clocking by precisely controlling the applied voltage conditions. The OVRs are theoretically calculable once the switching voltages are fixed, from Kirchhoff’s law. Here, different logic gates have different OVRs. But, commonly, when the switching voltages (i.e., the set voltage and reset voltage) are not a constant value but distributed, the OVR will be accordingly narrowed. When the switching voltage variation is significant, the OVR will eventually collapse, and the correspondent logic gate cannot be viable. Therefore, once the switching voltage variation is given from experiments, one can calculate not only the viability of the logic gate but also its error rate.
Kim et al. in 2020 suggested a methodology that would quantitatively represent the tolerance of stateful logic gates against such switching voltage variations. In the method, they derived equations representing the boundary between the success and failure of the gate operation for all input and output cases. Also, they considered the boundary after the gate operation to prevent unwanted switching of other memristors due to redistributed voltages. They solved the simultaneous equations to find the solutions considering switching voltage variations. Accordingly, the author defined a variation tolerance factor (VTF) for each gate that identifies the allowable switching voltage variation of the device for successful logic gate operation.

Figure 5c–e shows the VTF values of all SGF gates assuming various memristor set and reset voltages. The VTF values of RGF gates can be found elsewhere. As a result, there are more viable gates having higher VTF values than others. The 3NOR, 2IMP, and 2NOT gates have a VTF value of 0.333. To utilize those gates, the set voltage variation should be controlled to less than 0.333 of the $V_{SET}$, which is feasible in a typical memristive device.

4CARRY and 5SUM gates become viable, and then the 4 bit full-adder operation can be executed in $2 \times 4$ clocks using an $11 \times 1$ array. This switching variation control is crucial in stateful logic as it directly influences computing efficiency.

In addition to the switching voltage variation, the results also suggested that a higher reset voltage amplitude can allow a higher VTF value. This is because one can apply a higher amplitude of negative conditioning voltage on the inhibiting cell. Therefore, developing a higher reset voltage device would be a feasible solution for a more efficient stateful logic device.

5.2. Adoption of Error Correction Strategy

A stateful logic device requires highly controlled switching voltage distribution. In a practical device, the variation should be guaranteed not only over cycle to cycle and cell to cell, but also over device to device, which becomes more challenging. Therefore, a fundamental solution is needed that can make the stateful logic device work regardless of the variation issues. This is possible using error correction.

Error correction requires sacrificing a certain amount of the advantages of in-memory computing because it requires auxiliary peripheral circuits. Although it takes the computing regime from in-memory to near-memory, error correction is unavoidable for device realization. Then, the challenge with error correction is making the circuit as simple as possible.
In et al. in 2020 first proposed a viable error correction method for stateful logic gates.\textsuperscript{116} In the method, auxiliary peripheral CMOS-based circuits were added beside the array, which was designed to perform error detection and correction. Error correction was possible after understanding the principles of error occurrence. For example, in the 3NOR gate, the complete elimination of error is impossible when the switching voltage variation is higher than the VTF value. However, one can control the error cases by controlling the operating voltages.

Figure 6a shows the error rates for the 3NOR gate operation with respect to different input cases at different operating voltages. When the operating voltages were 0.59 and 0.71 V, the total error rate was lowest, but errors can occur for all input cases, which is hard to deal with. However, when the operation voltages were 0.59 and below 0.6 V, although the total error rate was increased, the error occurred only in one input case, “00.” Then, the error correction unit needed to consider only whether the “00” input gave the output correctly to “1.” For that purpose, the CMOS-based NOR gate is the optimum solution as it generates a TRUE output only when all inputs are FALSE.

In detail, the error correction sequence will be as follows. First, the gate operation is executed. Second, the inputs and outputs are delivered to inputs of the error correction module. That data delivery is possible simply by amplifying the output. At this moment, there is no need to read the data yet, making it operate in a near-memory regime. Then, the error correction unit generates the TRUE output indicating the error when all inputs and outputs are FALSE, which is the only error case. Then, the controller can forcibly program the TRUE value onto the output cell to correct the error. This zeros-counting method can be used for NOR-type gates, such as 3NOR, 2NOT, and 2IMP gates, making the error correction unit universal. In other words, those gates can share the same error correction unit so that error correction can be simple and efficient. The authors also proposed another error correction method based on odd parity checking for non-NOR gates. Although the circuit used is different depending on the combination of XOR gates, the detailed sequence is almost identical; it receives the input and output data from the array and generates an error verification output.

The error correction method requires at least three additional computation clocks (data delivery, error checking, and error correction), so it is likely to deteriorate the computing efficiency. This is true if an identical pathway is used for calculation. However, the error correction makes previously nonviable gates possible, which can provide a shortcut for computation. Eventually, the total computing efficiency can be improved even when the error correction is implemented. Figure 6b compares computing efficiency at the most optimized condition for executing the full adder. The error correction method enabled the 5SUM gate, which can significantly contribute to improving efficiency, outweighing the efficiency lost by error correction.

6. Conclusion

The advantages of memristive stateful logic devices are certainly attractive; in principle, they can perform complete in-memory computing. With proper error correction, their regime expands to near-memory. However, the memristor’s switching speed is much slower than that of a CMOS logic device; it typically ranges from 50 ns to 100 μs due to various practical limitations such as charging effects of the measurement setup, although some pioneering works showed that it could be as fast as a picosecond regime.\textsuperscript{117,118} Therefore, the stateful logic device is not a substitute for performance-oriented conventional computers and applications. Although it seems unlikely that stateful logic devices will displace the conventional von Neumann computer, they are superior to the conventional von Neumann computer architecture in terms of simplicity. And they are well suited to new positions in the emerging future, where low-power operation and efficient cost-computing capability will be in high demand. The advantages of stateful logic devices coincide well with these future requirements.

Until then, further innovations are expected in applications of stateful logic technology. Stateful logic technology itself is already highly mature. The next innovations will involve synergies with other emerging technologies. Two such potential synergies are expected.

The first synergy involves combining the present approach with other memristive computing technologies. Various types of memristive computing are known in the near-memory regime, based on the forms of input and output.\textsuperscript{119} In stateful logic, they are all resistances and thus can be defined with $R-R$ logic.\textsuperscript{120} Alternately, $V-R$ logic\textsuperscript{121–126} and $V-V$ logic\textsuperscript{127–129} are
also possible. As the latter methods belong to the near-memory category, their potential by themselves is limited. However, they can be used to support and enhance stateful logic efficiency. Second, stateful logic can be merged with other applications that utilize the memristive crossbar array platform, such as neuromorphic accelerators or hypercomputing. Those potential opportunities will further develop the feasibility of stateful logic and eventually lead to new perspectives on computing that conventional computers cannot supply.

Acknowledgements

Y.S.K. and M.Y.S. contributed equally to this work. This research was supported by the Ministry of Trade, Industry & Energy (MOTIE) (grant numbers 20003655 and 20003789), the Korea Semiconductor Research Consortium (K SRC) support program for the development of the future semiconductor device, and the 2020 UP Research Project of KAIST.

Conflict of Interest

The authors declare no conflict of interest.

Keywords

in-memory computing, Internet of things, memristors, neuromorphic, stateful logic

Received: December 16, 2020
Revised: January 27, 2021
Published online: April 1, 2021

[1] G. E. Moore, Electronics 1965, 38, 114.
[2] C. G. Hwang, in IEEE Int. Solid-State Circuits Conf., Digest of Technical Papers, IEEE, San Francisco, CA, USA 2002, pp. 24–27.
[3] M. M. Waldrop, Nature 2016, 530, 144.
[4] W. Arden, M. Brillouet, P. Cogez, M. Graef, B. Huizing, R. Mahnkopf, More-Than-Moore, White Paper, 2010.
[5] Q. Shi, B. Dong, T. He, Z. Sun, J. Zhu, Z. Zhang, C. Lee, InfoMat 2020, 2, 1131.
[6] S. Park, Y. Jeong, H. J. Jin, J. Park, H. Jang, S. Lee, W. Huh, H. Cho, H. G. Shin, K. Kim, C. H. Lee, S. Choi, S. Im, ACS Nano 2020, 14, 12064.
[7] M. Zhu, T. He, C. Lee, Appl. Phys. Res. 2020, 7, https://doi.org/10.1063/5.0016485.
[8] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, IEEE Internet Things J. 2016, 3, 637.
[9] M. Satyanarayanan, Computer 2017, 50, 30.
[10] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, IEEE Internet Things J. 2018, 5, 450.
[11] J. Von Neumann, M. D. Godfrey, IEEE Ann. Hist. Comput. 1993, 15, 27.
[12] W. H. Kautz, IEEE Trans. Comput. 1969, C-18, 719.
[13] H. S. Stone, IEEE Trans. Comput. 1970, C-19, 73.
[14] S. Ghose, A. Boroumand, J. S. Kim, J. Gómez-Luna, O. Mutlu, IBM J. Res. Dev. 2019, 63, 1.
[15] L. O. Chua, IEEE Trans. Circuit Theory 1971, 18, 507.
[16] L. O. Chua, S. M. Kang, Proc. IEEE 1976, 64, 209.
[17] D. B. Strukov, G. S. Snider, D. R. Stewart, R. S. Williams, Nature 2008, 453, 80.
[18] I. Abraham, Sci. Rep. 2018, 8, 1.
[19] S. Vongehr, X. Meng, Sci. Rep. 2015, 5, 1.
[20] F. Z. Wang, L. Li, L. Shi, H. Wu, L. O. Chua, J. Appl. Phys. 2019, 125.
[21] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, R. S. Williams, Nature 2010, 464, 873.
[22] H. Wu, P. Yao, B. Gao, H. Qian, Nat. Electron. 2018, 1, 8.
[23] M. A. Zidan, J. P. Strachan, W. D. Lu, Nat. Electron. 2018, 1, 22.
[24] F. Hui, E. Grustan-Gutierrez, S. Long, Q. Liu, A. K. Ott, A. C. Ferrari, M. Lanza, Adv. Electron. Mater. 2017, 3, 1.
[25] M. Lanza, H. S. P. Wong, E. Pop, D. Ielmini, D. Strukov, B. C. Regan, L. Racher, M. A. Villena, J. J. Yang, L. Goux, A. Belmonte, Y. Yang, F. M. Puglisi, J. Kang, B. Magvari-Köpe, E. Yalon, A. Kenyon, M. Buckwell, A. Mehonic, A. Shluger, H. Li, T. H. Hou, B. Hudec, A. Akinwande, R. Ge, S. Ambrogi, J. B. Roldan, E. Miranda, J. Suhe, K. L. Pey, et al., Adv. Electron. Mater. 2019, 5, 1.
[26] G. Snider, Appl. Phys. A 2005, 80, 1165.
[27] P. Kuekes, in Memristor Memristive Syst. Symp., UC Berkeley, Berkeley, CA 2008.
[28] K. Kim, S. Shin, S. M. Kang, in Proc. IEEE Int. Symp. Circuits Systems IEE, Rio de Janeiro 2011, pp. 2497–2500.
[29] K. Kim, S. Shin, S. M. Kang, IEEE Trans. Comput. Des. Integr. Circuits Syst. 2011, 30, 1800.
[30] S. Shin, K. Kim, S. M. Kang, IEEE Trans. Circuits Syst. II Express Briefs 2011, 58, 442.
[31] E. Lehtonen, J. H. Poikonen, M. Laiho, presented at Int. Workshop on Cellular Nanoscale Networks and their Applications, IEEE, Turin 2012, pp. 1–6.
[32] X. Zhu, X. Yang, C. Wu, N. Xiao, J. Wu, X. Yi, IEEE Trans. Circuits Syst. II Express Briefs 2013, 60, 682.
[33] S. Kvatinsky, D. Belousov, S. Liman, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, U. C. Weiser, IEEE Trans. Circuits Syst. II Express Briefs 2014, 61, 895.
[34] 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.
[35] Y. S. Kim, M. W. Son, H. Song, J. Park, J. An, J. B. Jeon, G. Y. Kim, S. Son, K. M. Kim, Adv. Intell. Syst. 2020, 2, 1900156.
[36] E. Lehtonen, J. H. Poikonen, M. Laiho, in Memristor Networks (Eds: A. Adamatzky, L. Chua), Springer, Berlin, 2014, pp. 603–623.
[37] B. Chen, F. Cai, J. Zhou, W. Ma, P. Sheridan, W. D. Lu, in IEEE Int. Electron Devices Meeting IEEE, Washington, DC 2015, p. 17.5.1–17.5.4.
[38] H. Li, Z. Chen, W. Ma, B. Gao, P. Huang, L. Liu, X. Liu, J. Kang, IEEE Electron Device Lett. 2015, 36, 1142.
[39] L. Xie, H. A. Du Nguyen, M. Taouil, S. Hamdioui, K. Bertels, in 33rd IEEE Int. Conf. on Computer Design, IEEE, New York, NY 2015, pp. 335–342.
[40] P. Huang, J. Kang, Y. Zhao, S. Chen, R. Han, Z. Zhou, Z. Chen, W. Ma, M. Li, L. Liu, X. Liu, Adv. Mater. 2016, 28, 9758.
[41] N. Wald, S. Kvatinsky, in Int. Conf. on the Science of Electrical Engineering, IEEE, Elat 2016, pp. 1–5.
[42] Q. Chen, X. Wang, H. Wan, R. Yang, IEEE Trans. Comput. Des. Integr. Circuits Syst. 2017, 36, 279.
[43] H. Li, T. F. Wu, S. Mitra, H. S. P. Wong, IEEE Trans. Circuits Syst. I Regul. Pap. 2017, 64, 2263.
[44] Z. Sun, E. Ambrosi, A. Bricalli, D. Ielmini, Adv. Mater. 2018, 30, 1802554.
[45] B. C. Jang, Y. Nam, B. J. Koo, J. Choi, S. G. Im, S. H. K. Park, S. Y. Choi, Adv. Funct. Mater. 2018, 28, 1.
[46] S. Gupta, M. Imani, T. Rosing, IEEE/ACM Int. Conf. Computer Design, IEEE, San Diego, CA 2018, 1–7.
[47] K. M. Kim, R. S. Williams, IEEE Trans. Circuits Syst. I Regul. Pap. 2019, 11.
Young Seok Kim is a Ph.D. student in the Department of Materials Science and Engineering at the Korea Advanced Institute of Science and Technology (KAIST). He received his B.S. degree in the Department of Materials Science and Engineering at Korea University in 2012 and M.S. degree in the Department of Materials Science and Engineering at KAIST in 2018. His current research interests include memristive stateful logic and material design for better operation of stateful logic.

Myeong Won Son is an M.S. student in the Department of Materials Science and Engineering at the Korea Advanced Institute of Science and Technology (KAIST). He received his B.S. degree from Ulsan National Institute of Science and Technology (UNIST). His research interests are advanced semiconductor materials and their applications to improve the performance of the overall semiconductor system.

Kyung Min Kim is an associate professor in the Department of Materials Science and Engineering at the Korea Advanced Institute of Science and Technology (KAIST) since 2017. He received a B.S. degree in 2003 and a Ph.D. degree in 2008 from Seoul National University, Seoul, Korea. From 2011 to 2013, he worked at Samsung Electronics in Korea. From 2014 to 2017, he worked at Hewlett Packard Labs of Hewlett Packard Enterprise at Palo Alto, California, USA. His research interests include memristor applications for future computing, such as in-memory computing devices and neuromorphic devices.