Hindawi
Scientific Programming
Volume 2021, Article ID 9041150, 8 pages
https://doi.org/10.1155/2021/9041150

Research Article

Decision Support System in a Memristor-Based Mobile CIM Architecture Applied on Big Data Computation and Storage

Wei Li[1] **and Zhao Deng**[2]

[1]*Collage of Computer Science, South-Central University for Nationalities, Wuhan, China*
[2]*State Key Laboratory of Advanced Technology for Materials Synthesis and Processing, Wuhan University of Technology, Wuhan, China*

Correspondence should be addressed to Wei Li; liwei@mail.scuec.edu.cn

Received 11 September 2021; Revised 2 November 2021; Accepted 8 November 2021; Published 14 December 2021

Academic Editor: Waqas Nawaz

Data computation and storage are essential parts of developing big data applications. The memristor device technology could remove the speed and energy efficiency bottleneck in the existing data processing. The present experimental work investigates the decision support system in a new architecture, computation-in-memory (CIM) architecture, which can be utilized to store and process big data in the same physical location at a faster rate. The decision support system is used for data computation and storage, with the aims of helping memory units read, write, and erase data and supporting their decisions under big data communication ambiguities. Data communication is realized within the crossbar by the support of peripheral controller blocks. The feasibility of the CIM architecture, adaptive read, write, and erase methods, and memory accuracy were investigated. The integrated circuit emphasis (SPICE) simulation results show that the proposed CIM architecture has the potential of improving the computing efficiency, energy consumption, and performance area by at least two orders of magnitude. CIM architecture may be used to mitigate big data processing limits caused by the conventional computer architecture and complementary metal-oxide-semiconductor (CMOS) transistor process technologies.

1. Introduction

Digital datasets have been rapidly growing in size and complexity, ranging from economics and business activities to public administration and from national security to many scientific research areas. Big data applications require computing resources and storage systems that can scale to manage a massive amount of diverse data, and improvements in the energy consumption and throughput of digital processors are reaching a plateau as CMOS technology approaches the end of process scaling. Memristor material, a kind of nanoscale analog memory, due to its special features including nonvolatility, nanoscale dimensions, and low power consumption, could be the best choice today for realizing signal processing at the mobile big data scale [1, 2].

Through symmetry arguments, Chua postulated the memristor to be the fourth fundamental circuit element in 1971 [3]. However, memristor is not a physical device, so it did not receive public attention until 2008, when Williams and his research group in the HP laboratory unveiled a two-terminal TiO_2 nanoscale device that exhibited memristive hysteresis characteristics [4], thus generating a strong interest in the memristor [5–7]. Williams and his coworkers showed that memristors could realize nanoscale crossbar memory and replace the existing computer memory systems in the future while taking up a much smaller area [8].

The concept of computation-centric architecture has been attracting a lot of attention for more than 40 years. In 1969, logic-in-memory (LIM) was originally introduced as a memory accelerator. In 1992, LIM concept reappeared and was named computational RAM. In the late 1990s and early 2000s, processor-in-memory (PIM) was presented. In 2004, memory-in-logic (MIL) was proposed, which provides massive addressable memory on the processor for supercomputer systems [9]. However, most of previous works just focus on either processor speed or memory speed, and computing is currently still stuck with the von Neumann architecture, which exchanges data between the CPU and the memory (cache) [10–12]. It is unavoidable to suffer from a memory bottleneck and negative impacts on the performance. This paper attempts to obtain a CIM architecture with fewer controllable wires and higher memory capacity as compared to the existing crossbar array architecture. A memory cell structure and an adaptive read method are also introduced to restrain the sneak paths.

2. Proposed Methodology

2.1. Memristor-Based CIM Architecture. Figure 1 shows the concept of memristor-based CIM. The storage and computation are integrated together in a dense nanowire crossbar array, and memristor devices are injected at each junction. Data communication is realized within the crossbar by the support of peripheral controller blocks.

Figure 2(a) shows the realistic CIM architecture, which is arranged in massive layers. From Figure 2(b), it can be observed that each layer comprises several columns and rows that are interconnected in a stair-step shape. When the cell is set to state "off," its conductance is low, encoding "0"; when the cell is set to state "on," its conductance is high, encoding "1." In order to write to and read from such an array, some peripheral circuitries are configured. The write/read control circuitry is the system control center, which generates corresponding signals to the multiplex voltage converter according to the target cell location. The system can be reset by reset circuitry.

The control wires follow time-division multiplexing for signal transmission, i.e., the same wire sends the control signal to a memory cell vertically at one moment and sends the control signal to another memory cell horizontally at another moment. By contrast, the existing memristor-based memory array is usually fabricated in a high-density crossbar architecture that has been described in the literature; it is referred to as an independent crossbar array in this paper. The memristors are located at each junction of a vertical and a horizontal nanowire, and each memristor can also be used as a 1 bit memory cell. According to the number of control wires (m for the vertical wire and n for the horizontal wire), the memory array size is $m \times n$, where m is usually equal to n.

In order to elucidate the interactive array, we present a smaller array example, in which the number of control wires is 5, as shown in Figure 2(b). There are 10 memory cells in the interactive array, and the cell at the junction of the ith and the jth control wire is denoted by (i, j), $i = 1, \ldots, 4$ and $j = 2, \ldots, 5$, respectively. V_k $(k = 1, \ldots, 5)$ refers to the voltage potentials on these wires. The voltage across the memristor cell (i, j) is denoted by V_{ji}, and $V_{ji} = V_j - V_i$.

The memory cell is fabricated using a series connection of a threshold memristor and two diodes in parallel and reverse to each other, as shown in the right upper side of Figure 2(b). When V_{ji} is positive, switch S_{ji} is on. In the opposite case, when V_{ji} is negative, switch S_{ij} is on. The diodes in the memory cell can shut off some reverse current flowing through the undesired sneak paths. The threshold memristor has specific switch characteristics. By applying a positive voltage equal to or higher than the positive threshold value V_{OPEN}, the memristor is set to state "on"; the resistance of the memristor is denoted by R_{ON}. Similarly, by applying a negative voltage equal to or lower than the negative threshold value V_{CLOSE}, the memristor is set to state "off"; the resistance of the memristor is denoted by R_{OFF}.

Assume that we want to write "1" in the memory cell (i, j) in Figure 2(b). To this end, the voltage potentials on the control wires can be set as follows.

Let

$$\begin{aligned} &0.75V < V_{\text{OPEN}} < 1.5V \\ &-1.5V < V_{\text{CLOSE}} < -0.75V. \end{aligned} \tag{1}$$

Let

$$V_k = \begin{cases} 0 & (k = j) \\ 1.50\text{V} & (k = i) \\ 0.75\text{V} & (k \neq i, j) \end{cases}. \tag{2}$$

With these settings for the small array example, we have selected the target memory cell (1, 3). The voltage across the control wires can be deduced as follows:

$$\begin{bmatrix} V_{12} & & & \\ V_{13} & V_{23} & & \\ V_{14} & V_{24} & V_{34} & \\ V_{15} & V_{25} & V_{35} & V_{45} \end{bmatrix} = \begin{bmatrix} +0.75 & & & \\ +1.50 & +0.75 & & \\ +0.75 & 0 & -0.75 & \\ +0.75 & 0 & -0.75 & 0 \end{bmatrix} V. \tag{3}$$

Only the state of memory cell (1, 3) can be set; the states of the other cells are invariable. If we want to write "0" in the memory cell (i, j), the sign of all the above wire voltage potentials should be simply reversed.

2.2. Read, Write, and Erase Operations. The memristance of the memristor is related to the lowest possible resistance R_{ON} and the highest possible resistance R_{OFF}, as well as the boundary state constant ω/D. As can be seen in Figure 3, for simplicity, the memristor state can be defined as logic zero when $0<\omega/D < 0.5$ and logic one when $0.5<\omega/D < 1$. The corresponding limit states are $\omega/D = 0$ and $\omega/D = 1$, respectively. In reality, to account for possible noise injection, the memristor state can be defined as logic zero when $0<\omega/D < 0.4$ and logic one when $0.6<\omega/D < 1$. In other words, a confusion region in between $0.4<\omega/D < 0.6$ should be avoided for read and write. The block diagram is shown in Figure 4.

Figure 5(a) shows the read circuit, which produces several signals to implement read operation. Two sample signals, which control the conversion of current-to-voltage samples on capacitors C_1 and C_2, are controlled by switches S_1 and S_2. At the beginning of write, read, and erase operations, switch S_3 is asserted to balance the charge on both capacitors. Once the signals are sampled, then the sense-enable operation is performed by first asserting HS high and later LS high; HS and LS are shown in Figure 5(b). The

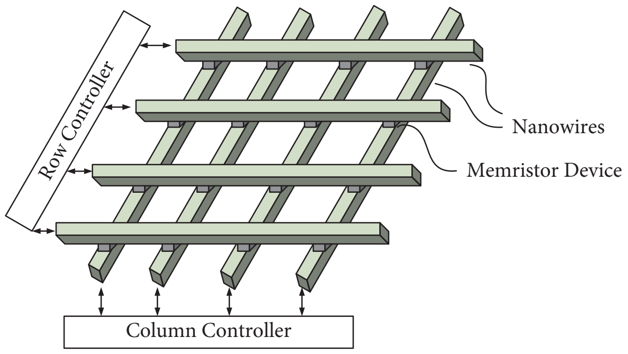

FIGURE 1: The concept of memristor-based CIM.

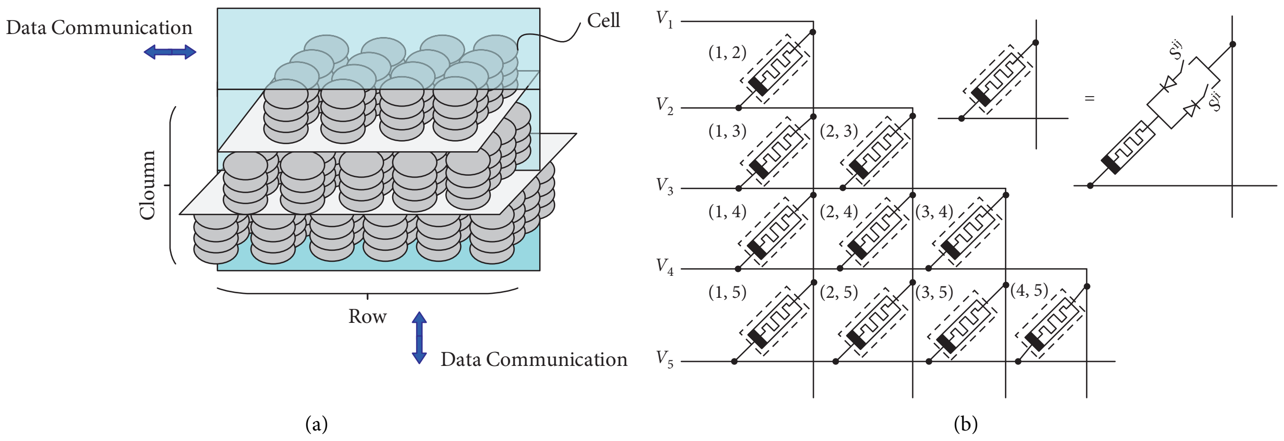

FIGURE 2: (a) A 3-layer realistic stack architecture of CIM. (b) An example of a 4×4 stair-step crossbar architecture.

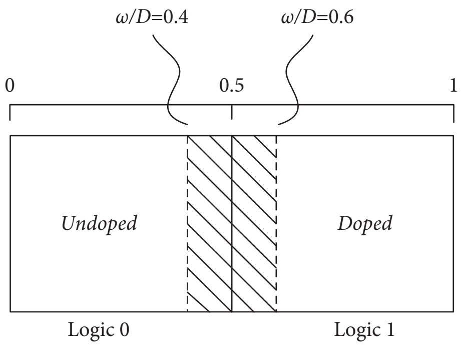

FIGURE 3: Memristor state definition.

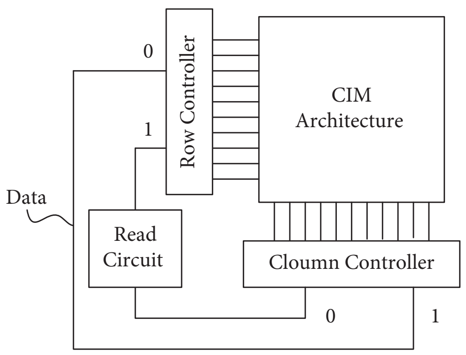

FIGURE 4: Top-level block diagram.

voltage at point V_X is purposefully measured to produce a low output of low resistance and a high output of high resistance.

The write and erase operations are the extensions of a single-cycle read operation. When the write operation is performed, the exciting source will generate the write one pulse or write zero pulse depending on the incoming data, but the write operation is not as simple as the read operation. This section delves into write operations with respect to the flow diagram in Figure 6.

To avoid the destructive signal issue in write operations, we propose to improve the memristor-based memory cell structure. The proposed structure is in Figure 7. Both read and write operations are implemented by such a structure, and the R/W-enable switch acts as an operation choice. In the write operation, R/W enable switches to the ground. In the read operation, it switches to V_X.

The comparator has two thresholds VH_1 and VH_2. It compares V_X with the two thresholds. If $V_X > VH_2$, output V_O is V_{OH} which represents logic 1. Similarly, V_O is V_{OH} too. When the value of V_X is between VH_1 and VH_2, the output of the comparator is V_{OL} which represents logic 0.

We apply an exciting source pulse on the memristor and produce a voltage signal V_X, which is further amplified into a

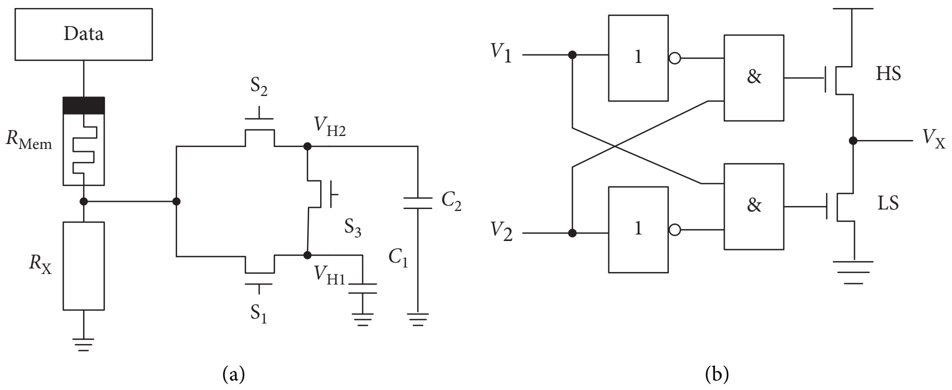

Figure 5: (a) Read sampling circuit. (b) Sense comparator that determines the resistance state.

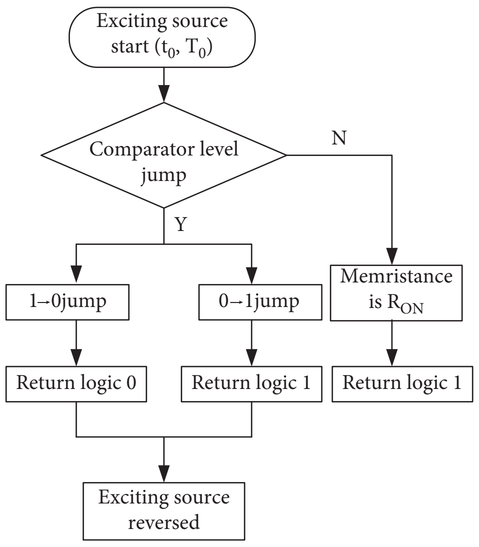

Figure 6: Write flow diagram.

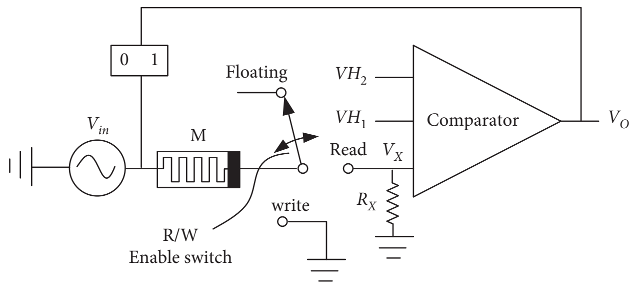

Figure 7: Circuit structure for a memristor-based memory cell.

comparator. After the pulse stage, the sense stage detects the memristor state by the output of the comparator. When the initial boundary state constant is higher than 0.6, its initial state is logic 0, and the initial output of the comparator is V_{OL}. The positive pulse is applied on the memristor from t_0, and then the initial boundary state constant becomes higher, responding to that the memristance of the memristor becomes smaller. The comparator output V_O jumps from V_{OL} to V_{OH} at t_1. Similarly, the positive pulse is applied on the memristor from T_0, and the comparator output V_O jumps from V_{OH} to V_{OL} at T_1 when the initial boundary state constant is lower than 0.4, whose initial state presents logic 0.

Figure 8 shows an example of the read scheme. When the initial boundary state constant is higher than 0.6, the comparator output V_O jumps from V_{OL} to V_{OH} at t_1. Once the jump signal is sampled by the inverter, then a negative pulse is performed on the memristor until t_2, where the time interval between t_1 and t_0 is equal to that between t_2 and t_1; this stage is convert stage. Similarly, when the initial boundary state constant is lower than 0.4, the comparator output V_O jumps from V_{OH} to V_{OL} at T_1; once the jump signal is sampled by the inverter, then a negative pulse is performed on the memristor until T_2, where the time interval between T_1 and T_0 is equal to that between T_2 and T_1, but there is some difference from the former state; when the negative pulse is performed on the memristor, the comparator output jumps from V_{OL} to V_{OH} again. The comparator jumps twice for an instant, and the worst case is that the initial boundary state constant is 1, and then the comparator output maintains the same without a jump step.

3. Experimental Results and Discussion

In order to validate the proposed stair-step array structure performance and read method, we present our simulation results in this section. We write to the memristors that can be switched to an "ON" or "OFF" state and read the states by measuring the current from memristors. The simulation is based on a SPICE model built upon TiO_2 memristive switching made in the HP laboratory. The simulation parameters are set as follows: $V_{\text{CLOSE}} = 1.0\,\text{V}$, $V_{\text{OPEN}} = -1.0\,\text{V}$, $R_{\text{OFF}} = 2.5 \times 10^6\,\Omega$, and $R_{\text{ON}} = 2.5 \times 10^2\,\Omega$. The simulation approach consists of considering different memory conditions on an $m + n = 8$ array. The memristor of interest is situated in the center of the array.

As mentioned before, the write and erase operations are the extensions of a single-cycle read operation. The signals V_{H1} and V_{H2} presented in Figure 5 are appropriately renamed as V_{RH} and V_{RL}, which helps facilitate the understanding of simulation results. The renamed logic signals denote the memristor of interest in high resistance state and low resistance state. Figure 9(a) shows the number of cycles required for a write operation, while Figure 9(b) shows the

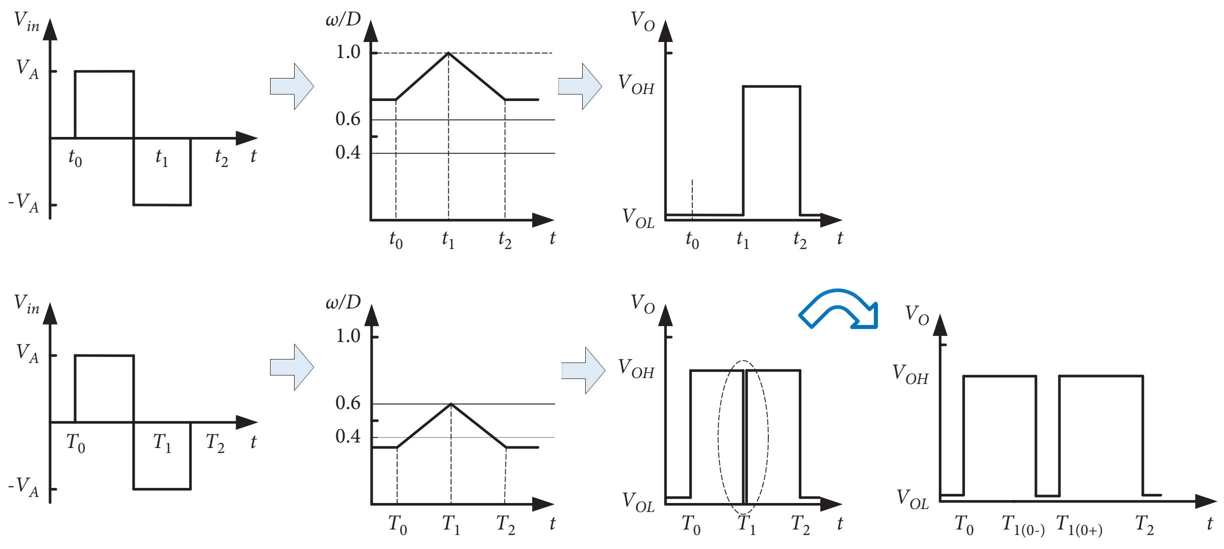

Figure 8: Example of the read scheme.

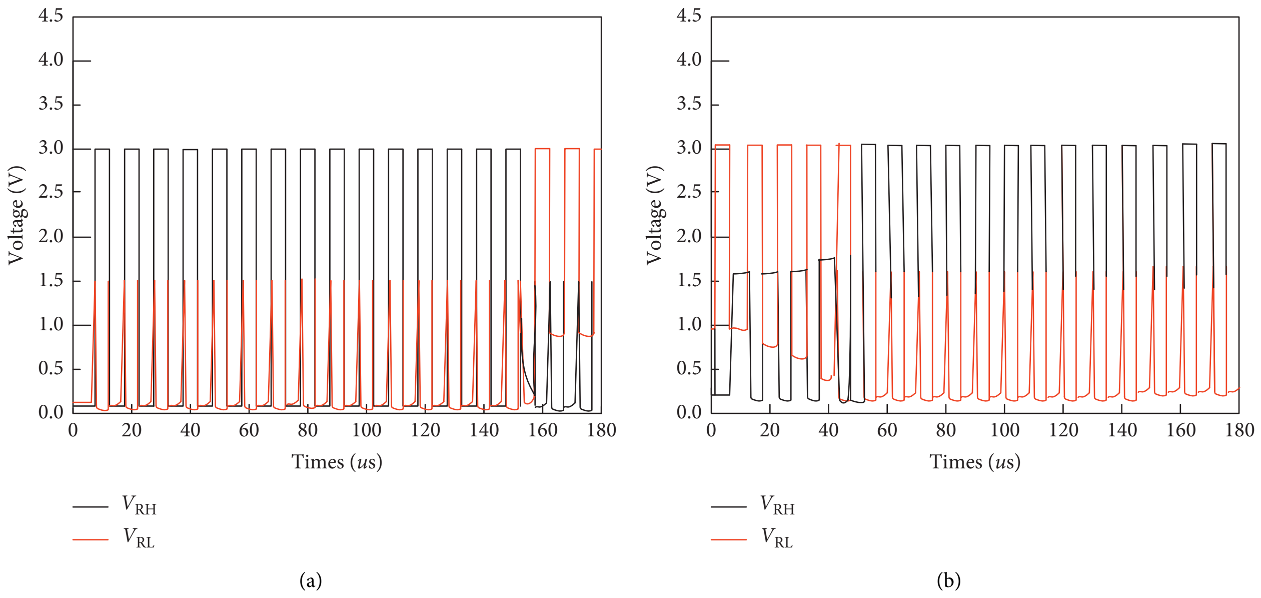

Figure 9: (a) Resistance-state signals for write. (b) Resistance-state signals for erase.

number of cycles required for an erase operation. Each read operation provides the memristor state feedback, and the memristor only changes from high resistance to low resistance when the memristor is written to its lowest level. The simulation results imply that, during memory operation, the number of read operations necessary for a write after an erase may be different. And this adaptive method will prevent any overerasing and overwriting.

Along with the read, write, and erase operations, we are also interested in the memory capacity of arrays with the same number of nanowires. Assume that both the independent crossbar array and the interactive crossbar array possess the same number of control wires; this means that the number of control wires of the latter one is $m + n$, and the number of memory cells can be calculated as

$$\begin{aligned} 1 + 2 + \cdots + (m + n - 1) &= \frac{1}{2}(m + n)(m + n - 1) \\ &= m \times n + \frac{1}{2}\left(m^2 + n^2 - m - n\right). \end{aligned} \tag{4}$$

If and only if $m = 1$, $n = 1$, and the number of memory cells is the same, then the latter one has more memory cells. The detailed comparisons between the independent crossbar array and the interactive crossbar array are given in Table 1.

The memory capacity (i.e., array size) of the interactive array can be compared to that of an independent array for the case where the control wire numbers are the same, as shown in Figure 10. It is obvious that the interactive crossbar

Table 1: Comparison between the two crossbar arrays.

Items	Independent	Interactive
Array size	$m = n = 4$	$m + n = 8$
Cell number	16	28
Array size	$m = n = 8$	$m + n = 16$
Cell number	64	120
Array size	$m = n = 16$	$m + n = 32$
Cell number	256	496

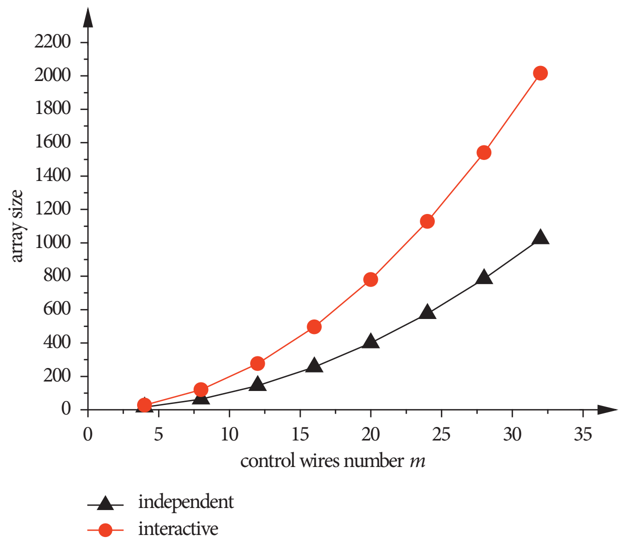

Figure 10: Array size of interactive and independent.

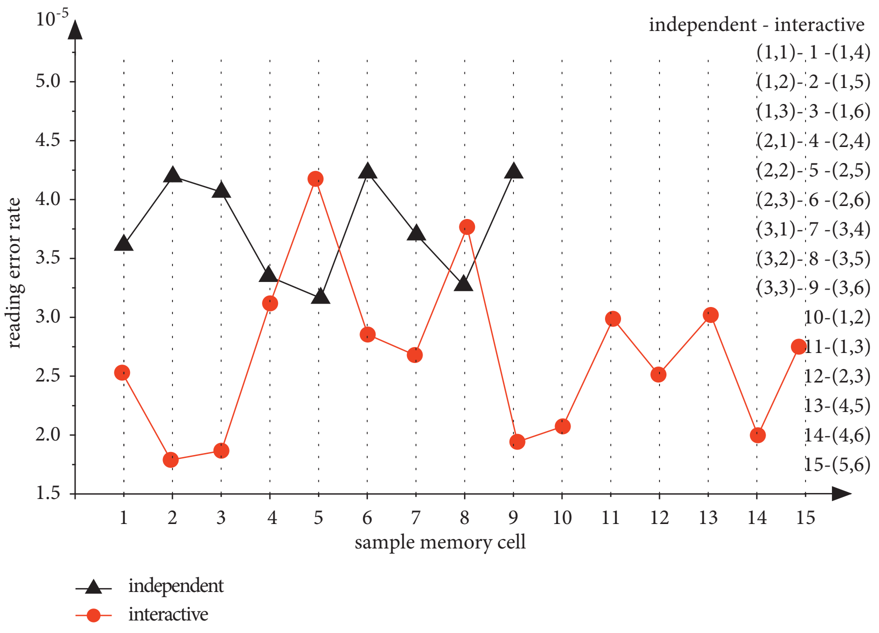

Figure 11: Read operation error rate of the interactive array and independent array.

array has a higher memory capacity, and the larger the array size, the higher the memory capacity.

Another concern besides the memory operation is the read operation error rate. From Figure 11, it can be observed that the sample memory cells form a 3×3 independent crossbar array (9 memory cells) and an interactive crossbar (15 memory cells), respectively. Both arrays are fabricated by 6 control wires and the same memory cell structure. The

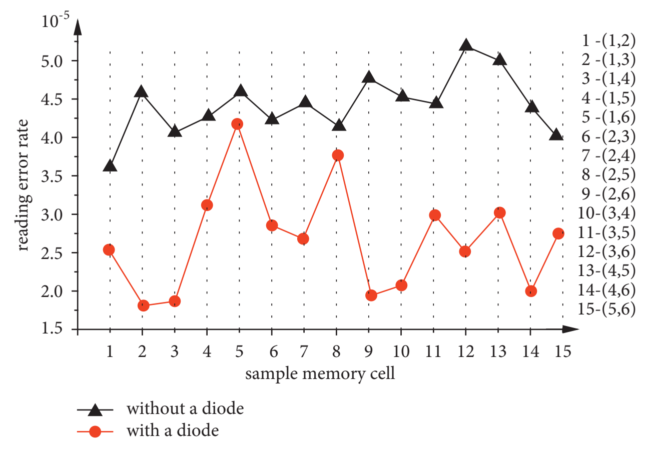

Figure 12: Read operation error rate of memory cells from two arrays.

memory cells in the former array are denoted as (m, n), where m and n are the number of control lines, and $m = 1, 2, 3$ and $n = 1, 2, 3$, respectively. The memory cells in the latter array are denoted as (i, j); we write to and read from each sample memory cell 10^5 times and count the read operation error rate. The simulation results show that the interactive crossbar array has a higher accuracy in the read operation than the independent crossbar array. This may be attributed to the fact that the interactive crossbar array has fewer current paths and, as a result, less sneak paths.

The diode in memory cells is another factor that may contribute to the restraining of sneak paths. In Figure 12, the sample memory cells are from two interactive crossbar arrays: one array with memory cells that are fabricated only by a memristor (without a diode) and another array with memory cells that are described in Section 2 (with a diode). The results show that the memory cell structure proposed in this paper is an effective solution to the sneak path problem.

4. Conclusions

This paper presented an experimental study on the CIM architecture, which can be utilized to store and process big data in the same physical location. The feasibility of the CIM architecture and adaptive read, write, and erase operation methods were investigated. The following conclusions can be drawn from this study.

The showcased CIM architecture is more effective in memory capacity than the independent crossbar array. On the one hand, the interactive crossbar CIM architecture has a higher memory capacity, and the larger the array size, the higher the memory capacity. On the other hand, CIM architecture preventsovererasing and overwriting and has fewer sneak paths. The method of achieving the memory operation relates adaptively to each memristor cell thereby allowing for the increased yield when it comes to using devices that differ from high to low (or low to high) resistance states. The CIM architecture also exhibits higher operation accuracy and lower energy consumption, which is the crux of mobile data processing. The CIM architecture emphasizes the importance of storing multibit data. This may indeed enable and prepare to process massive data in the big data era.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Conflicts of Interest

The authors declare no conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgments

The authors acknowledge the National Natural Science Foundation of China (Grant no. 61503418) and the Fundamental Research Funds for the Central Universities, South-Central University for Nationalities (Grant no. CZY19013).

References

[1] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2019.

[2] A.-Q. Gbadamosi, L. Oyedele, A.-M. Mahamadu et al., "Big data for design options repository: towards a DFMA approach for offsite construction," *Automation in Construction*, vol. 120, Article ID 103388, 2020.

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

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

[5] M. Syed Ali, R. Saravanakumar, and J. Cao, "New passivity criteria for memristor-based neutral-type stochastic bam neural networks with mixed time-varying delays," *Neurocomputing*, vol. 171, no. 1, pp. 1533–1547, 2016.

[6] Y. Shi, J. Cao, and G. Chen, "Exponential stability of complex-valued memristor-based neural networks with time-varying delays," *Applied Mathematics and Computation*, vol. 313, pp. 222–234, 2017.

[7] L. Su and L. Zhou, "Exponential synchronization of memristor-based recurrent neural networks with multi-proportional delays," *Neural Computing & Applications*, vol. 31, no. 5, pp. 7907–7920, 2019.

[8] P. O. Vontobel, W. Robinett, P. J. Kuekes, D. R. Stewart, J. Straznicky, and R. Stanley Williams, "Writing to and reading from a nano-scale crossbar memory based on memristors," *Nanotechnology*, vol. 20, no. 42, Article ID 425204, 2009.

[9] B. J. Jasionowski, K. Lay, M. Michelle, and M. Margala, "A processor-in-memory architecture for multimedia compression," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 15, no. 4, pp. 478–483, 2007.

[10] S. A. Josselyn and S. Tonegawa, "Memory engrams: recalling the past and imagining the future," *Science*, vol. 367, no. 6473, p. eaaw4325, 2020.

[11] W. Hirst, J. K. Yamashiro, and A. Coman, "Collective memory from a psychological perspective," *Trends in Cognitive Sciences*, vol. 22, no. 5, pp. 438–451, 2018.

[12] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing data where it makes sense: enabling in-memory computation," *Microprocessors and Microsystems*, vol. 67, no. 3, pp. 28–41, 2019.