International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 5, sep-Oct-2026, ISSN (Online): 2395-566X

Memristor-Based In-Memory Computing for Artificial Intelligence Acceleration: A Review of Devices, Crossbar Architectures, Benefits and Challenges

Randive Abhijeet Santosh[1] Sadaphal Harshal Ravindra[2] Shinde Prashant Shantaram[3]
Prof. S. E. Shinde[4] Prof. M. S. Nalawade[5] Dr. V. M. Dhede[6]
[1,2,3] Department Of Electronics And Telecommunication Jaihind College Of Engineering Kuran, Pune, Maharashtra, India
[4]Seminar Guide Department Of Electronics And Telecommunication Jaihind College Of Engineering Kuran, Pune, Maharashtra, India;
[5]Seminar Coordinator Department Of Electronics And Telecommunication Jaihind College Of Engineering Kuran, Pune, Maharashtra, India;
[6]Head Of Department Department Of Electronics And Telecommunication Jaihind College Of Engineering Kuran, Pune, Maharashtra, India

Abstract—Artificial intelligence workloads are dominated by multiply-and-accumulate operations on very large weight matrices. In conventional processors these weights must be moved repeatedly between memory and the arithmetic units, and this data movement, rather than the arithmetic itself, consumes most of the energy and time. Memristor-based in-memory computing (IMC) addresses this limitation by performing computation inside the memory array. A memristor is a two-terminal resistive switching device whose conductance can be programmed and retained, so it can store a weight. When memristors are arranged in a crossbar, Ohm's law and Kirchhoff's current law cause the column currents to equal the product of an input voltage vector and the conductance matrix, so a complete vector–matrix multiplication is obtained in a single analog step. This review explains the memristor concept and its physical realisations, the working principle of crossbar arrays, and the peripheral circuits and system architecture needed to build an accelerator. Representative device-level and system-level studies are compared. The benefits for AI inference and training are assessed against non-idealities such as device variability, limited precision, conductance drift, wire resistance and the cost of analog-to-digital conversion. Applications and research directions are identified.

Index Terms—Memristor, in-memory computing, resistive switching memory, RRAM, crossbar array, vector–matrix multiplication, AI accelerator, von Neumann bottleneck.

I. INTRODUCTION

The past decade has seen remarkable growth in artificial intelligence (AI), driven by deep neural networks for vision, speech and language. These networks contain millions to billions of parameters, and both training and inference involve repeated multiplication of input vectors by weight matrices. Conventional hardware follows the von Neumann model, in which memory and processing are separate units connected by a bus. Every operation therefore requires weights and activations to be fetched from memory and results to be written back. This separation creates the so-called memory wall. Wulf and McKee observed decades ago that processor speed was improving faster than memory speed, so that memory access would eventually limit performance [1]. For energy, the imbalance is even larger: Horowitz showed that reading a value from off-chip DRAM costs orders of magnitude more energy than performing an arithmetic operation on it [2]. For data-intensive AI workloads, data movement therefore dominates both latency and energy.

In-memory computing (IMC) reduces this movement by carrying out computation where the data are stored [3], [4]. Among the candidate technologies, memristors, also called resistive switching devices, are especially attractive. Their conductance can be set to many levels and retained without power, and they can be packed densely in crossbar arrays. When voltages are applied to the rows of such an array, the currents collected on the columns directly represent a vector–matrix product [5].This paper reviews memristor-based IMC with a focus on AI acceleration. Its contributions are:

- An explanation of the memristor concept, its physical implementations and its switching behaviour.
- A description of the crossbar working principle and the peripheral circuits needed for computation.
- A review and comparison of representative device and system studies, including complete accelerator architectures.
- An assessment of advantages, limitations, applications and future research directions.

Section II explains the background, Section III describes crossbar-based computing, Section IV presents the system architecture, Section V reviews and compares the literature, Section VI discusses advantages, disadvantages and

© 2026 IJSRET

applications, Section VII presents challenges, and Section VIII concludes.

II. BACKGROUND

2.1 The Memristor Concept

In 1971, Chua argued from symmetry that, in addition to the resistor, capacitor and inductor, there should be a fourth fundamental two-terminal element relating charge and magnetic flux, which he called the memristor [6]. Its defining property is a resistance that depends on the history of current that has passed through it. In 2008, Strukov *et al.* at HP Labs showed that a thin titanium dioxide film between platinum electrodes behaves as a memristor, because the movement of oxygen vacancies under an applied field changes the boundary between a conductive and an insulating region [7].

2.2 Resistive Switching Devices

Many devices with history-dependent resistance are now grouped under the memristor label [8]. Metal-oxide resistive random-access memory (RRAM) switches by forming and partially dissolving a conductive filament of oxygen vacancies in an oxide such as HfO_x or TaO_x [9]. Conductive-bridge memory uses a metallic filament, and phase-change memory (PCM) switches between amorphous and crystalline phases. In all cases, a SET operation increases conductance and a RESET operation decreases it, and with careful programming a device can hold several intermediate conductance levels [3].

2.3 Device Requirements For Computing

For computing, the important properties are the number of distinguishable conductance levels, the linearity and symmetry of conductance updates, the variation between devices and between cycles, retention over time, endurance under repeated programming, and the ratio between high and low resistance states [10], [11].

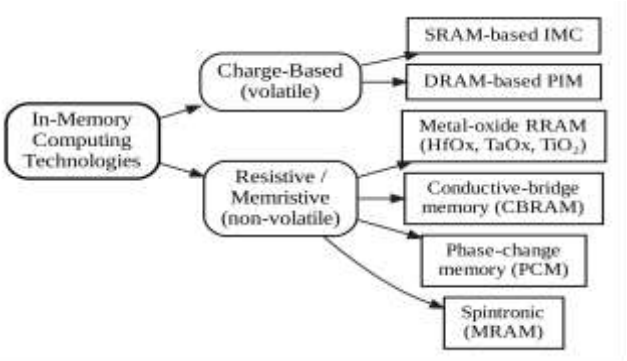


Fig. 1. Classification of memory technologies used for in-memory computing.

III. CROSSBAR-BASED COMPUTING

3.1 Crossbar Structure

A crossbar consists of parallel horizontal word lines crossing parallel vertical bit lines, with a memristor at every intersection. To reduce leakage through unselected cells, each memristor is often paired with a transistor in a one-transistor–one-resistor (1T1R) cell [5]. Fig. 2 illustrates the principle.

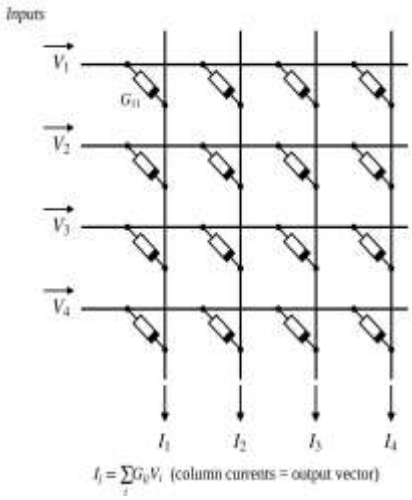


Fig. 2. Memristor crossbar performing analog vector–matrix multiplication; input voltages V_i drive the rows and each column current I_j sums the products G_{ij} V_i.

3.2 Analog Vector–Matrix Multiplication

If the conductance of the device at row *i* and column *j* is G_{ij} and a voltage V_i is applied to row *i*, Ohm's law gives a current $G_{ij}V_i$ through that device. Kirchhoff's current law then sums the currents flowing into column *j*:

$$I_j = \Sigma_i G_{ij} \cdot V_i \qquad (1)$$

The full vector of column currents is therefore the product of the input voltage vector and the conductance matrix, computed in one step regardless of matrix size [3], [5]. Because weights stay in place and only the inputs and outputs move, data movement is greatly reduced.

3.3 Representing Signed Weights And Precision

Conductance is always positive, so a signed weight is usually represented by the difference between two devices, $W = G^+ - G^-$. Higher precision can be obtained by splitting a weight across several devices with different significance, or by applying inputs bit by bit and combining partial results with shift-and-add circuits [12].

© 2026 IJSRET

3.4 Peripheral Circuits

Digital inputs must be converted to voltages by digital-to-analog converters (DACs) or applied as pulses, and column currents must be converted back to digital values by analog-to-digital converters (ADCs) after sample-and-hold. Shafiee *et al.* found that ADCs could dominate the power and area of a crossbar accelerator, which makes their design a central concern [12].

IV. SYSTEM ARCHITECTURE

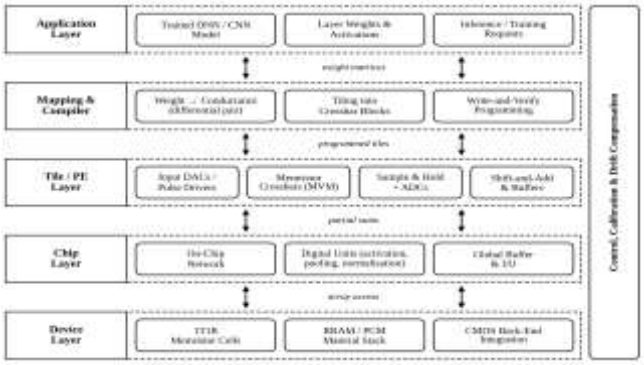


Fig. 3 shows a layered architecture for a memristor-based AI accelerator, derived from the reviewed designs [12], [13].

At the **application layer**, a trained neural network provides layers and weights. The **mapping layer** converts weights into conductance targets, splits large matrices into crossbar-sized tiles and programs the devices with write-and-verify loops. The **tile layer** contains processing elements, each with several crossbars, input DACs or pulse drivers, ADCs, shift-and-add units and local buffers. The **chip layer** connects tiles through an on-chip network and provides digital units for activation functions, pooling and normalisation, which are not easily performed in analog form. The **device layer** consists of the 1T1R memristor arrays themselves. A control and calibration function spans all layers to compensate for drift and variability.

V. LITERATURE REVIEW AND COMPARISON

Prezioso *et al.* trained and operated a small neural network on a 12 × 12 crossbar of metal-oxide memristors that classified simple patterns, demonstrating learning directly in hardware [14]. Hu *et al.* presented the dot-product engine, programming a 1T1M crossbar to perform matrix–vector multiplication with accuracy suitable for neural network inference [5]. At the system level, ISAAC by Shafiee *et al.* proposed a pipelined CNN accelerator using crossbars for in-situ analog arithmetic and reported large improvements in throughput and energy over a digital baseline in simulation [12]. PRIME by Chi *et al.* integrated computation into ReRAM main memory so that part of the memory array could be reconfigured as a neural network accelerator [13]. Ambrogio *et al.* combined phase-change memory with capacitor-based weights and reached software-equivalent accuracy in training on standard datasets [15]. Yao *et al.* built a fully hardware-implemented memristor convolutional neural network with multiple arrays and proposed hybrid training to tolerate device imperfections [16].

Table I Summary Of Representative Studies On Memristor-Based In-Memory Computing

Ref.	Level	Contribution	Key Result	Limitation
[7]	Device	Physical TiO2 memristor	Links resistive switching to Chua's theory	Single-device study
[14]	Array	On-chip trained network on 12 × 12 crossbar	Learning demonstrated in hardware	Very small network
[5]	Array	Dot-product engine with 1T1M cells	Accurate analog matrix–vector multiply	Precision limited by devices
[12]	System	ISAAC pipelined CNN accelerator	Large throughput and energy gains (simulated)	ADC overhead dominates
[13]	System	PRIME processing in ReRAM main memory	Reuses memory as accelerator	Simulation-based evaluation
[15]	Array/system	PCM with capacitor weights for training	Software-equivalent training accuracy	Complex weight cell
[16]	System	Fully hardware memristor CNN	Multi-array CNN with hybrid training	Moderate network size

© 2026 IJSRET

Table II compares memristor-based IMC with other computing approaches for AI.

Criterion	CPU	GPU / Digital Accelerator	SRAM-Based IMC	Memristor-Based IMC
Data movement	Very high	High	Low	Very low
Weight storage	External DRAM	DRAM / HBM	Volatile SRAM	Non-volatile, dense
Precision	Full digital	Full / reduced digital	Moderate	Limited, analog
Energy per MAC	High	Medium	Low	Very low (potential)
Maturity	Mature	Mature	Emerging products	Research prototypes
Main issue	Memory wall	Power, bandwidth	Area, volatility	Variability, ADC cost

VI. ADVANTAGES, DISADVANTAGES AND APPLICATIONS

6.1 Advantages

The principal advantage is reduced data movement, since weights remain stationary in the array [4]. Parallel analog computation completes a full vector–matrix product in one step, giving high throughput. Memristors are non-volatile, so weights are retained without power, which suits devices that are frequently switched off. Their small cell size allows high storage density, and fabrication is compatible with the back end of standard CMOS processes [11].

6.2 Disadvantages

Analog computation is imprecise. Device-to-device and cycle-to-cycle variation, limited numbers of conductance levels, non-linear and asymmetric updates, conductance drift over time and noise all reduce accuracy [10]. Wire resistance causes voltage drops along long lines, and sneak currents flow through unselected cells in passive arrays. ADCs and DACs consume considerable area and power [12]. Endurance limits make frequent re-programming during training difficult.

6.3 Applications

The most immediate application is energy-efficient inference at the edge, in cameras, wearables and sensors, where non-volatility and low power are valuable. Other applications include acceleration in data centres, on-chip learning, neuromorphic computing that mimics synapses, signal processing, and solving linear equations and optimisation problems with analog arrays [3], [11].

VII. CHALLENGES AND FUTURE DIRECTIONS

Several research directions are active. At the device level, materials and programming schemes that give more stable, linear and reproducible conductance levels are needed [9]. At the circuit level, low-power ADCs, shared converters and current-domain or time-domain readout aim to reduce peripheral cost [12]. At the algorithm level, hardware-aware training adds noise and quantisation during training so that the network tolerates device imperfections, and periodic calibration compensates for drift [16]. At the system level, compilers that map different neural network layers onto crossbars and digital units, and standard benchmarks for fair comparison with digital accelerators, are still developing. Large-scale integration and reliable manufacturing remain the main steps before memristor IMC can complement digital accelerators in commercial products.

VIII. CONCLUSION

Memristor-based in-memory computing offers a direct answer to the data-movement problem that limits conventional hardware for AI. By storing weights as conductances and using the physics of crossbar arrays to compute vector–matrix products in one analog step, it promises large gains in energy efficiency and throughput. This review has explained the device concept, crossbar principle, peripheral circuits and accelerator architecture, and has compared representative studies from single devices to complete systems. Device variability, limited precision, drift and converter overhead remain significant challenges. Progress in devices, circuits, hardware-aware algorithms and design tools is needed before the approach can be widely deployed, but the field connects semiconductor devices, analog circuits, computer architecture and AI hardware in a way that makes it an important topic for future research.

REFERENCES

1. W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," ACM SIGARCH Computer Architecture News, vol. 23, no. 1, pp. 20–24, 1995.

© 2026 IJSRET

2. M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC), 2014, pp. 10–14.
3. D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," Nature Electronics, vol. 1, no. 6, pp. 333–343, 2018.
4. A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," Nature Nanotechnology, vol. 15, no. 7, pp. 529–544, 2020.
5. M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in Proc. 53rd ACM/EDAC/IEEE Design Automation Conf. (DAC), 2016, Art. no. 19.
6. L. O. Chua, "Memristor—The missing circuit element," IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507–519, 1971.
7. 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.
8. J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," Nature Nanotechnology, vol. 8, no. 1, pp. 13–24, 2013.
9. H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide RRAM," Proceedings of the IEEE, vol. 100, no. 6, pp. 1951–1970, 2012.
10. S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," Proceedings of the IEEE, vol. 106, no. 2, pp. 260–285, 2018.
11. M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," Nature Electronics, vol. 1, no. 1, pp. 22–29, 2018.
12. A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in Proc. 43rd ACM/IEEE Int. Symp. Computer Architecture (ISCA), 2016, pp. 14–26.
13. P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in Proc. 43rd ACM/IEEE Int. Symp. Computer Architecture (ISCA), 2016, pp. 27–39.
14. M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," Nature, vol. 521, no. 7550, pp. 61–64, 2015.
15. S. Ambrogio et al., "Equivalent-accuracy accelerated neural-network training using analogue memory," Nature, vol. 558, no. 7708, pp. 60–67, 2018.
16. P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," Nature, vol. 577, no. 7792, pp. 641–646, 2020.

© 2026 IJSRET