Low-power memristor-based computing for edge-AI applications

Singh, Abhairaj; Diware, Sumit; Gebregiorgis, Anteneh; Bishnoi, Rajendra; Catthoor, Francky; Joshi, Rajiv V.; Hamdioui, Said

DOI
10.1109/ISCAS51556.2021.9401226

Publication date
2021

Document Version
Final published version

Published in
2021 IEEE International Symposium on Circuits and Systems (ISCAS)

2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401226

Low-power Memristor-based Computing for Edge-AI Applications

Abhairaj Singh*, Sumit Diware*, Anteneh Gebregiorgis*, Rajendra Bishnoi*, Francky Catthoor†,
Rajiv V. Joshi‡ and Said Hamdioui*
*Computer Engineering Lab, Delft University of Technology, Delft, The Netherlands
{A.Singh-5, S.S.Diware, A.B.Gebregiorgis, R.K.Bishnoi, S.Hamdioui}@tudelft.nl
†IMEC, Belgium, {Francky.Catthoor}@imec.be
‡IBM Research Division Yorktown Heights, NY, US, {rvjoshi}@us.ibm.com

Abstract—**With the rise of the Internet of Things (IoT), a huge market for so-called smart edge-devices is foreseen for millions of applications, like personalized healthcare and smart robotics. These devices have to bring smart computing directly where the data is generated, while coping with the limited energy budget. Conventional von-Neumann architecture fail to meet these requirements due to e.g., memory-processor data transfer bottleneck. Memristor-based computation-in-memory (CIM) has the potential to realize smart local computing for highly parallel data-dominated AI applications by exploiting the inherent properties of the architecture and the physical characteristics of the memristors. This paper provides a broad overview of CIM architecture highlighting its potential and unique properties in enabling smart local computing. Moreover, it discusses design considerations of such architectures including both crossbar array as well as peripheral circuits; special attention is given to analog-to-digital converter (ADC), as it is the most critical unit of analog-based CIM operation e.g., vector-matrix multiplication (VMM). Finally, the paper outlines the potential future directions for CIM-based edge smart computing.**

I. Introduction

The breakthrough in Artificial Intelligence (AI) has led to a booming increase in AI-based applications and services [1]. Although existing AI applications have achieved state-of-the-art performance in various fields, their dependency on cloud servers presents strict resource requirements such as memory and network bandwidth [2]–[4]. Edge computing (aka edge-AI), is a promising solution to overcome those barriers by performing local computing (on the edge-devices) [5]–[7]. The main advantages of edge-AI over traditional AI applications can be summarized as: (*i*) Energy-efficiency: In comparison to cloud computing, edge-AI offers a more energy-efficient approach by processing only relevant data; (*ii*) Real-time response: Due to local computation, edge-AI saves substantial amount of time, which enables a nearly real-time response; (*iii*) Storage-efficiency: edge-AI reduces the amount of data stored in centralized systems by local processing and decision-making. However, edge-AI has stringent requirements that must be dealt with in order to harness its full potential; edge-AI hardware must be fast, compact and extremely energy-efficient, as edge-devices have limited resource such as battery lifetime or harvested energy [5], [8], [9].

The existing AI processing architectures based on the conventional von-Neumann architecture (such as CPU, GPU and TPU) are severely constrained by the so-called ”memory wall” [10], [11]. As a result, excessive time and energy are spent in moving massive amounts of data between the memory and data paths [12]. This challenge is imperiling the deployment of AI on edge-devices. Therefore, a paradigm shift is paramount to unlock the full potential of edge-AI. In this regard, memristor-based computation-in-memory (CIM) has the potential to break the aforementioned challenge (due to the nature of the architecture and the devices used to realize it) and deliver energy efficient implementations of hardware edge-AI [13]–[15]. Such memristor-based CIM architecture uses non-volatile devices to store the data while exploiting their inherent capability to perform computation on the stored data and hence, circumventing the costly data movement of von-Neumann based systems [16].

This paper highlights memristor-based CIM architecture as a potential candidate for edge-AI applications, and discusses the low-power design aspects of such an architecture. It provides a broad overview of different memristor-based CIM architectures, memristor device technologies, crossbar configurations and related periphery circuits. The paper gives special attention to the design of ADCs as they are the most critical units of analog-based CIM for e.g., multiply-and-accumulate (MAC) operations.

II. Computation-in-Memory (CIM)

A. CIM Basics and Classification

CIM refers to the computing paradigm where the computation (i.e., execution) of an operation is performed within the memory core. Referring to Fig. 1 [16], depending on *where* the result is generated, CIM can be classified into i) CIM-Array (aka CIM-A) in which the results are produced in the crossbar array (marked as circled 1) [17]–[19]; and ii) CIM-Periphery (aka CIM-P) where the results are produced in periphery (marked as circled 2) [20]–[22]. Moreover, both classes can be further classified into: (1) basic architectures requiring design changes *only* inside the memory array (CIM-Ab) or *only* in the periphery (CIM-Pb), and (2) hybrid where in addition to *major* changes in the memory array minimal to

This research is supported by EC Horizon 2020 Research and Innovation through MNEMOSENE project under Grant 780215.

978-1-7281-9201-7/21/$31.00 ©2021 IEEE

medium changes are required in the peripheral circuit (CIM-Ah) or vice versa (CIM-Ph) [16].

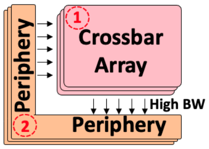


Fig. 1. CIM core architecture and its classification [16].

CIM aims at utilizing non-volatile, CMOS-compatible and highly scalable memristor devices whose inherent nature enables both storage and computing capabilities. This makes it possible for CIM to exploit maximum bandwidth and massive reduction of data movement between the memory and external processing unit, resulting into extremely low power operations. Therefore, CIM is a promising candidate for fast, low-power budget edge-AI applications.

B. CIM Architectural Units

1) Crossbar Array: We will use CIM-Ph primitive computational unit for illustration; it is extensively explored for matrix-matrix multiplication (MMM) with large operand sizes for edge-AI applications [23], [24]. Fig. 2a shows a subset of MMM; i.e., vector-matrix multiplication (VMM) performing several multiply-accumulate (MAC) operations that encompasses the most fundamental computational unit in different domains such as complex neural networks.

VMM is performed by applying a voltage vector $V{=}V_j$ (where $j{\in}\{1,m\}$) to memristor-crossbar matrix of conductance values $G{=}G_{ij}$ (where $i{\in}\{1,n\}$, $j{\in}\{1,m\}$). At any instance, each column performs a vector-vector multiplication (VVM) or a MAC operation, with the output current vector I, in which each element is $I_i{=}\Sigma V_j \cdot G_{ij}$. Note that all n MAC operations are performed with O(1) time complexity.

2) Periphery: A CIM core can be inherited from standard well-established memory cores such as SRAMs and DRAMs, but with some major modifications to accommodate analog-based computing, as shown in Fig. 2b. Firstly, the CMOS-based bitcell comprising the memory unit is replaced by memristor-based bitcell configured in a compact crossbar array, as described previously. The circuit blocks comprising the periphery that supports the bitcell array are significantly modified depending on the operations CIM should accommodate. E.g., for MMM operations, the following is needed: 1) Row-decoder becomes complex as CIM involves enabling several rows in a single computation cycle. Also, *1-bit* row or word-line drivers are now replaced by digital-to-analog converters (DACs) that convert multi-bit VMM operands into an array of analog voltages. 2) Column periphery circuits performing read operations (i.e., *1-bit* sense-amplifiers) are now replaced by analog-to-digital converters (ADCs) to quantify currents as digital bit-streams. Post-processing circuits such as shift-and-add are required for MMM 3) Control block needs to deal with complex instructions such as handling intricacies of multi-operand VMM operations as opposed a simple read or write memory operation. The crossbar and periphery complexities are further discussed in subsequent sections.

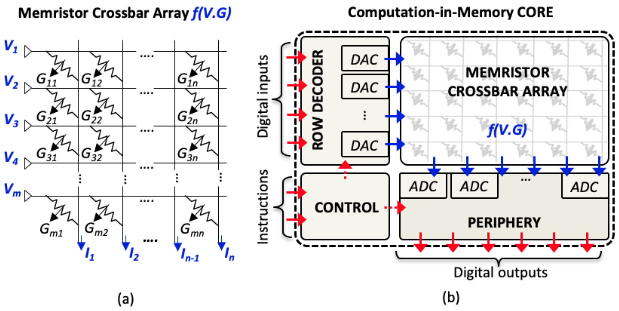


Fig. 2. (a) Scaling CIM-P primitive to a large-scale crossbar array. (b) Detailed CIM-P core [25].

C. CIM Potential

CIM can provide fast and efficient computing for a wide range of computation kernels. Some examples of such kernels along with their applications are: binary arithmetic (temporal correlation [26]), bitwise logic operations (database [27], encryption [28]), linear algebra (image processing, deep learning for edge-AI) [29].

Recent published work based on circuit simulation and small scale prototypes has shown that CIM is very promising.

Simulation based work reported that CIM architecture provides two to three orders of magnitude improvement in energy-delay product and energy spent per operation compared to conventional von-Neumann architecture [30], and around 10 fJ/arithmetic operation (1 MAC = 255 arithmetic operations) can be realized [24]. Small scale prototype work considering database query applications demonstrated that CIM architecture can achieve 6 fJ/logic operation. All these examples highlight the tremendous potential of CIM over von-Neumann architecture.

III. CIM-Crossbar Technology and Configurations

A. Device Technologies

CIM crossbar can be implemented using different device technologies; e.g., oxide-based resistive RAM (RRAM), phase change memory (PCM) and spin-transfer torque RAM (STT-RAM). Table I provides a comparison of the key attributes and applicability of these devices from CIM perspective.

Technology	**RRAM**	**PCM**	**STT-RAM**
Cell area [31] (F^2)	4-12	4-30	6-50
No. of bits $(per\ device)$	1-6 [32]	1-8 [33]	1 [34]
R_{off}/R_{on} $(unitless)$	10 [32]	100 [33]	2.8 [34]
Endurance [35] $(cycles)$	10^8-10^{12}	10^8-10^{15}	$>10^{15}$
Read latency [31] (ns)	<10	<10	<10
Write latency [31] (ns)	<10	~50	<10
Write energy [31] (pJ)	~0.1	~10	~0.1
Application kernels for edge-AI	VMM, MMM etc.	VMM, MMM etc	Digital logic, arithmetic etc.

TABLE I
Device technologies for CIM.

B. Resistive Memory Non-idealities

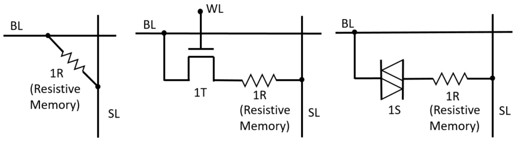
The underlying physics and fabrication process of a resistive memory device can lead to various non-idealities, causing deviation from its ideal behaviour as a programmable resistor. Here, we exclude the discussion regarding well-known CMOS device variations. Some of the major non-idealities are:

- **Device-to-device variation**: Due to fabrication imperfections, different resistive memory devices show different resistance characteristics under identical programming conditions [36]–[38].
- **Cycle-to-cycle variation**: Owing to the stochastic nature of underlying physics (e.g. filament formation/rupture in RRAM, crystallization/amorphization in PCM) the same resistive memory device shows different resistance characteristics under identical programming conditions at different points in time [36]–[38].
- **Resistance drift**: The accumulated effect of large number of read operations can lead to significant resistance change (drift) of the resistance state [39]–[41].
- **Nonlinear I-V characteristics**: Due to non-linear I-V characteristics of resistive memories, variation in a read voltage can lead to different effective resistance ratios [42]–[44], causing functional errors.

C. Bit-cell Configurations and Crossbar-level Issues

Bit-cell configuration refers to the structure of the basic building block (bit-cell) connected at each intersection of bit line (BL) and source line (SL). Fig. 3 shows the three main bit-cell configurations:

- **1 Resistor (1R) bit-cell** (Fig. 3a): This bit-cell configuration can provide the least crossbar area and energy consumption. However, the unwanted current paths that exist while programming or reading a single cell (e.g. during write-verify) in a crossbar with 1R bit-cell, referred as sneak paths; these can disturb the resistance states of other cells, reduce read margin and increase energy consumption [45], [46]. V/2 and V/3 biasing schemes [47] can prevent sneak paths for crossbar with 1R bit-cell, but at the cost of increased energy and half-select cell disturbs.
- **1-Transistor 1-Resistor (1T1R) bit-cell** (Fig. 3b): In this case the transistor overcomes the sneak paths issue and its gate can be used for programming [32]. However, such a transistor increases the crossbar area [48] and energy consumption. It acts as a nonlinear variable resistor in series with memristor, which can lead to errors in the crossbar VMM output.
- **1-Selector 1-Resistor (1S1R) bit-cell** (Fig. 3c): The selector suppresses the sneak paths. Note that 1S1R can provide better area footprint than 1T1R as selectors can be fabricated in back-end-of-line [49]. However, there is no perfect two-terminal selector possessing all the desired characteristics [50]. Turn-on time of the selector introduces delay in the crossbar operation [51]. Selectors also contribute non-linearity which can result in errors in the crossbar output.



(a) 1R (b) 1T1R (c) 1S1R

Fig. 3. Bit-cell Configurations.

Apart from sneak paths, another important issue at the crossbar level is the IR drop problem [52]. Due to the finite parasitic resistance of interconnect wires, cells connected to bit lines farther from the input port receive degraded voltages, which may lead to errors in the crossbar operation output.

For a CIM architecture running a practical edge-AI application, such as a neural network-based image recognition for autonomous vehicles, the aforementioned device-level non-idealities (e.g., device variations) and crossbar-level issues (e.g., IR drop) manifest as errors in vector-matrix multiplication, resulting in degraded accuracy. Hence, it is necessary to mitigate their impact [53], [54].

IV. CIM Periphery

Among the design blocks discussed in Section II, conversions performed by ADCs are very critical and challenging due to 1) Analog signals have low noise margin and hence, can lead to erroneous output [55]; 2) Analog computation heavily relies on memristors and CMOS selectors strength (e.g., for 1T1R), therefore these variations induce variation in output current [51], [52]; 3) Quantization error in ADCs increases as the the number of activation levels increases for higher resolution/accuracy [56]; In addition, the area/power increases drastically at higher accuracy, while speed reduces [56]. Fig. 4 shows that the ADC alone typically dominates CIM die area (>90%) and power consumption (>65%); this highlights the importance of ADC implementation for CIM architectures which could target e.g., machine learning algorithms such as Conventional Neural Networks (CNN) or Deep Neural Networks (DNNs)

A. ADC Design Methodology

A typical ADC design can be broken down into three design stages, as shown the Fig. 5. At first, the analog input (here, column current) is received by the *sensing stage*. As the first level of conversion, the analog input is converted to a voltage or to a time equivalent quantity; it can also quantified as current, and thus remains in the analog domain. Next is the *conversion stage*. Here, the obtained quantified quantity (from the sensing stage) is converted to discrete digital data, either as number of pulses or interim bit-streams of *0*s and *1*s. Finally is the *decision stage*. Here the number of pulses are quantified as digital bit-streams or the above interim bit-streams are

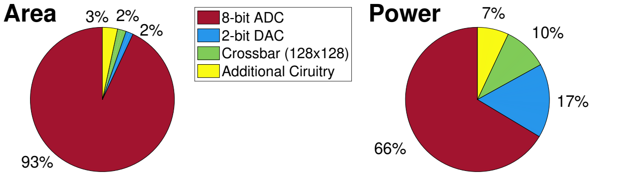


Fig. 4. Area and Power share of CIM design blocks [24].

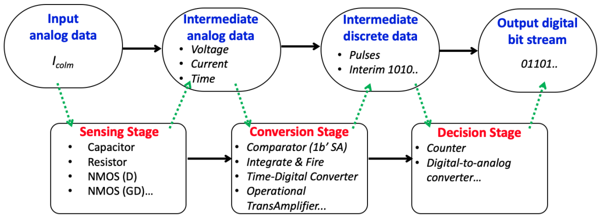


Fig. 5. Typical stages, physical parameters and circuit designs involved in an ADC design.

finalized. For the completeness, the circuit components used at each stage of conversion are also shown in Fig. 5.

B. ADC Classification

ADC designs can be classified based on the physical quantities used during the conversions. As it can be seen in Fig. 5, the first conversion is from current to voltage/current/time and second conversion is to discrete pulses/interim bit-streams before the final bit-streams are obtained; this results into six ADC classes; these are: Voltage-Pulse (VP), 2) Current-Pulse (CP), 3) Time-Pulse (TP), 4) Voltage-Interim-Digits (VD), 5) Current-Interim-Digits (CD), and 6) Time-Interim-Digits (TD).

C. Existing ADC Comparison

Fig. 6 maps the existing ADCs (based on the previous classification) while considering efficiency metrics in terms of latency, power, energy, area and supported resolution. These are derived from [25], [56]–[63]; the number of ADC designs considered to derive the metrics average values for each class are given in brackets (e.g., 2 designs for VP class). The metrics are normalized and quantified with level 1 (lowest) to 5 (highest) in the y-axis.

Analyzing Fig. 6 reveals some important information. First, ADCs that belong to the ”pulses” class consume relatively less power at the cost of speed and low resolution; cumulatively, energy is relatively low. Second, ADCs that belongs to the VD class are faster and support high resolution at the cost of power, energy and area. Third, ADCs in the TP class are typically inefficient in terms of latency, power and area as they need large sized block such as time-digital converters (TDCs) and phase locked loops (PLLs); these offer lower resolution compared to current and voltage-based ADCs. Hence, the designs that involve time-domain inter-conversions are least explored. Finally, the classes CD and TD are not explored yet; this may due to the complexity of such designs. In conclusion, VD, CP, and VP classes are mostly explored in literature, which is understandable as the merits and demerits of these two classes generally complement each other.

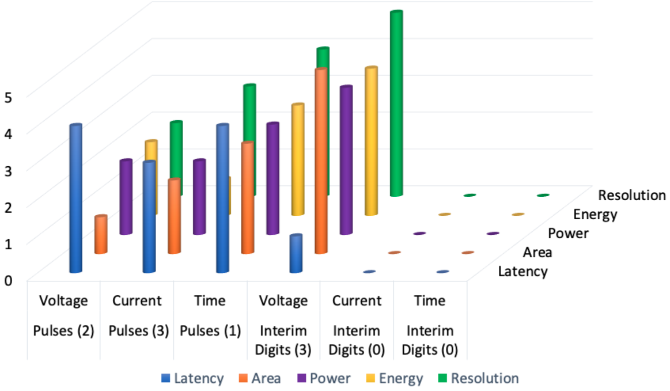


Fig. 6. Qualitative comparison of ADC designs involving different physical parameters [25], [56]–[63].

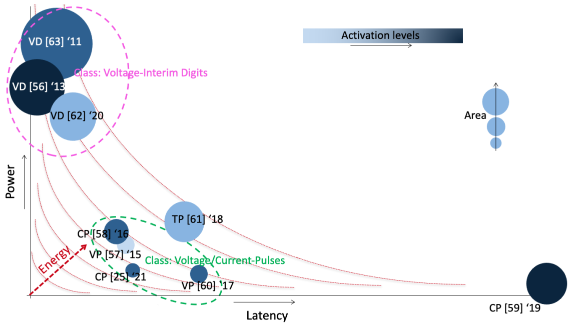


Fig. 7. Quantitative comparison of state-of-the-art ADC designs for CIM represented by class, reference and year of publication. [25], [56]–[63]. Larger area implies more columns shared per ADC and higher resolution implies higher maximum operand size for MAC operations. Red lines represents iso-energy contour lines.

Fig. 7 combines the most critical efficiency metrics together for these existing ADC in literature. It can be clearly seen that the speed and/or resolution are improved at the expense of power and/or area, and vice-versa. ADC designs in [25], [57], [58], [60] fall under VP/CP classes, and are compact, consume less power at the expense of speed and resolution. On the other hand, VD class ADC designs such as [56], [62], [63] are fast converters with high resolution; however, they suffer from high power and area consumption. Owing to this fact, ADC designs falling in the CP and VP (VD) classes are typically utilized in simpler (complex) networks supporting small (large) operand size targeting approximate (accurate) edge-AI computing.

V. Conclusion and Future Prospect

If successful, CIM will be able to significantly increase energy-efficiency by orders of magnitude; this may enable new power-constrained computing paradigms at the edge (e.g., neuromorphic computing, artificial and bio-inspired neural networks) which could fuel many application domains (e.g., wearable devices, wireless sensors, automotive). However, research on CIM is still in its infancy stage, and the challenges are substantial at all levels, including material/technology, circuit and architecture, and tools and compilers.

At the technology level, there are still many open questions; examples are device endurance, high resistance ratio between the off and on state of the devices, multi-level storage, precision of analog weight representation, etc. At the circuit and architecture levels, many challenges have to be still worked out; examples are high precision programming of memory elements, complexity of signal conversion circuits, accuracy of measuring (e.g., the current as a metric of the output), scalability of the crossbars and their impact on the accuracy of computing, the partitioning across crossbars and the corresponding intra- and inter-communication, etc. At the tools/compilers level, issues related to e.g., profiling, simulation and design tools are still open.

References

[1] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXiv preprint 1502.00873*, 2015.

[2] H. A. Alhaija *et al.*, "Augmented reality meets deep learning for car instance segmentation in urban scenes," in *British MVC*, 2017.

[3] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: deep belief networks with multitask learning," *Transactions on Intelligent Transportation Systems*, 2014.

[4] C. Potes, S. Parvaneh, A. Rahman, and B. Conroy, "Ensemble of feature-based and deep learning-based classifiers for detection of abnormal heart sounds," in *Computing in Cardiology Conference (CinC)*, 2016.

[5] T. Rausch, W. Hummer, V. Muthusamy, A. Rashed, and S. Dustdar, "Towards a serverless platform for edge {AI}," in *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[6] Y.-L. Lee, P.-K. Tsung, and M. Wu, "Techology trend of edge ai," in *Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2018.

[7] P. G. Lopez *et al.*, "Edge-centric computing: Vision and challenges," 2015.

[8] X. Xu *et al.*, "Scaling for edge inference of deep neural networks," *Nature Electronics*, 2018.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, 2016.

[10] H. A. Du Nguyen, J. Yu, L. Xie, M. Taouil, S. Hamdioui, and D. Fey, "Memristive devices for computing: Beyond cmos and beyond von neumann," in *Conference on Very Large Scale Integration*, 2017.

[11] S. Hamdioui *et al.*, "Memristor for computing: Myth or reality?" in *DATE*, 2017.

[12] ——, "Memristor based computation-in-memory architecture for data-intensive applications," in *DATE*, 2015.

[13] Hsu *et al.*, "Ai edge devices using computing-in-memory and processing-in-sensor: From system to device," in *IEDM*, 2019.

[14] Z. Zhou *et al.*, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, 2019.

[15] S. Hamdioui *et al.*, "Applications of computation-in-memory architectures based on memristive devices," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 486–491.

[16] H. A. D. Nguyen *et al.*, "A classification of memory-centric computing," *JETC*, 2020.

[17] J. Borghetti *et al.*, "'memristive'switches enable 'stateful'logic operations via material implication," *Nature*, 2010.

[18] S. Kvatinsky *et al.*, "Memristor-based imply logic design procedure," in *ICCD*, 2011.

[19] Kvatinsky *et al.*, "Magic—memristor-aided logic," *TCAS II: Express Briefs*, 2014.

[20] M. Hu *et al.*, "Hardware realization of bsb recall function using memristor crossbar arrays," in *DAC*, 2012.

[21] S. Li *et al.*, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *DAC*, 2016.

[22] L. Xie *et al.*, "Scouting logic: A novel memristor-based logic design for resistive computing," in *ISVLSI*, 2017.

[23] A. Velasquez and S. K. Jha, "Parallel boolean matrix multiplication in linear time using rectifying memristors," in *ISCAS*, 2016.

[24] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *ISCA*, 2016.

[25] A. Singh *et al.*, "SRIF: Scalable and reliable integrate and fire circuit adc for memristor-based cim architectures." *TCAS 1: Regular Papers*, 2021.

[26] A. Sebastian *et al.*, "Temporal correlation detection using computational phase-change memory," *Nature Communications*, 2017.

[27] I. Giannopoulos, A. Singh *et al.*, "In-memory database query," *Advanced Intelligent Systems*, 2020.

[28] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *MICRO*, 2017.

[29] A. Sebastian *et al.*, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, 2020.

[30] S. Hamdioui *et al.*, "Memristor based computation-in-memory architecture for data-intensive applications," in *DATE*, 2015.

[31] S. Yu and P. Chen, "Emerging memory technologies: Recent trends and prospects," *Solid-State Circuits Magazine*, 2016.

[32] M. Hu *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Advanced Materials*, 2018.

[33] I. Giannopoulos *et al.*, "8-bit precision in-memory multiplication with projected phase-change memory," in *IEDM*, 2018.

[34] W. Skowroński *et al.*, "Understanding stability diagram of perpendicular magnetic tunnel junctions," *Nature Scientific Reports*, 2017.

[35] B. Li, B. Yan, and H. Li, "An overview of in-memory processing with emerging non-volatile memory for data-intensive applications," in *GLSVLSI*, 2019.

[36] J.-H. Lee *et al.*, "Exploring cycle-to-cycle and device-to-device variation tolerance in MLC storage-based neural network training," *Transactions on Electron Devices*, 2019.

[37] G. Charan *et al.*, "Accurate inference with inaccurate RRAM devices: A joint algorithm-design solution," *Journal on Exploratory Solid-State Computational Devices and Circuits*, 2020.

[38] Y. Zhang *et al.*, "STT-RAM cell optimization considering MTJ and CMOS variations," *TMAG*, 2011.

[39] W. Shim, Y. Luo, J.-S. Seo, and S. Yu, "Investigation of read disturb and bipolar read scheme on multilevel rram-based deep learning inference engine," *Transactions on Electron Devices*, 2020.

[40] E. I. Vatajelu, P. Prinetto, M. Taouil, and S. Hamdioui, "Challenges and solutions in emerging memory testing," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 3, pp. 493–506, 2017.

[41] R. Bishnoi *et al.*, "Read disturb fault detection in STT-MRAM," in *ITC*, 2014.

[42] W. Zhang *et al.*, "Design guidelines of rram based neural-processing-unit: A joint device-circuit-algorithm analysis," in *DAC*, 2019.

[43] V. Karpov *et al.*, "Nucleation switching in phase change memory," *Applied Physics Letters*, 2007.

[44] M. Moinuddin *et al.*, "Low-current-density magnetic tunnel junctions for STT-RAM application using MgO$_x$N$_{1-x}$ $(x = 0.57$) tunnel barrier," *TED*, 2020.

[45] M. A. Lastras-Montaño and K. Cheng, "Resistive random-access memory based on ratioed memristors," *Nature Electronics*, 2018.

[46] Y. Ben-Hur and Y. Cassuto, "Detection and coding schemes for sneak-path interference in resistive memory arrays," *Communications*, 2019.

[47] L. Shi *et al.*, "Research progress on solutions to the sneak path issue in memristor crossbar arrays," *Nanoscale Advances*, 2020.

[48] C. Li *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, 2018.

[49] R. Midya *et al.*, "Anatomy of Ag/Hafnia-based selectors with 10^{10} nonlinearity," *Advanced Materials*, 2017.

[50] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Materials*, 2019.

[51] M. A. Zidan *et al.*, "Memristor based memory: The sneak paths problem and solutions," *Microelectronics Journal*, 2013.

[52] Y. Jeong *et al.*, "Parasitic effect analysis in memristor-array-based neuromorphic systems," *IEEE Transactions on Nanotechnology*, 2018.

[53] W. Shim, Y. Luo, J. Seo, and S. Yu, "Investigation of read disturb and bipolar read scheme on multilevel rram-based deep learning inference engine," *Transactions on Electron Devices*, 2020.

[54] Y. Jeong, M. A. Zidan, and W. D. Lu, "Parasitic effect analysis in memristor-array-based neuromorphic systems," *Transactions on Nanotechnology*, 2018.

[55] C. Liu *et al.*, "A memristor-based neuromorphic engine with a current sensing scheme for artificial neural network applications," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 647–652.

[56] Kull *et al.*, "A 3.1 mw 8b 1.2 gs/s single-channel asynchronous sar adc with alternate comparators for enhanced speed in 32 nm digital soi cmos," *JSSC*, 2013.

[57] Liu, "A spiking neuromorphic design with resistive crossbar," in *DAC*, 2015.

[58] Liu *et al.*, "A memristor crossbar based computing engine optimized for high speed and accuracy," in *ISVLSI*, 2016.

[59] Yan *et al.*, "Rram-based spiking nonvolatile computing-in-memory processing engine with precision-configurable in situ nonlinear activation," in *Symposium on VLSI Technology*, 2019.

[60] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *HPCA*, 2017.

[61] Jiang *et al.*, "Pulse-width modulation based dot-product engine for neuromorphic computing system using memristor crossbar array," in *ISCAS*, 2018.

[62] S. Yin, X. Sun, S. Yu, and J.-s. Seo, "High-throughput in-memory computing for binary deep neural networks with monolithically integrated rram and 90-nm cmos," *Transactions on Electron Devices*, 2020.

[63] P. Nuzzo *et al.*, "A 6-bit 50-ms/s threshold configuring sar adc in 90-nm digital cmos," *TCAS I: Regular Papers*, 2011.