Devices and Architectures for Efficient Computing In-Memory (CIM) Design

Bengel, Christopher; Gebregiorgis, Anteneh; Menzel, Stephan; Waser, Rainer; Gaydadjiev, Georgi; Hamdioui, Said

DOI
10.1007/978-3-031-46077-7_29

Publication date
2023

Document Version
Final published version

Published in
Embedded Computer Systems

Citation (APA)
Bengel, C., Gebregiorgis, A., Menzel, S., Waser, R., Gaydadjiev, G., & Hamdioui, S. (2023). Devices and Architectures for Efficient Computing In-Memory (CIM) Design. In C. Silvano, M. Reichenbach, & C. Pilato (Eds.), *Embedded Computer Systems: Architectures, Modeling, and Simulation - 23rd International Conference, SAMOS 2023, Proceedings* (pp. 437-450). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 14385 LNCS). Springer. https://doi.org/10.1007/978-3-031-46077-7_29

Devices and Architectures for Efficient Computing In-Memory (CIM) Design

Christopher Bengel[1], Anteneh Gebregiorgis[2(✉)], Stephan Menzel[3], Rainer Waser[1,3], Georgi Gaydadjiev[2], and Said Hamdioui[2]

[1] Institut für Werkstoffe der Elektrotechnik II, RWTH Aachen, Aachen, Germany

[2] Delft University of Technology, Delft, The Netherlands
a.b.gebregiorgis@tudelft.nl

[3] Peter Grünberg Institute, Forschungszentrum Jülich, Jülich, Germany

Abstract. Smart computing has demonstrated huge potential for various application sectors such as personalized healthcare and smart robotics. Smart computing aims bringing computing close to the source where the data is generated or stored. Memristor-based Computation-In-Memory (CIM) has the potential to realize such smart computing for data and computation intensive applications. This paper presents an overview and design present of CIM, covering from the architecture and circuit level down to the device level. On the circuit and device level, accelerators for machine learning will be presented and discussed, focusing on variability and reliability effects. We will discuss these aspects for Redox-based Resistive Random Access Memories (ReRAM) based on the Valence Change Mechanism (VCM) by employing the compact model JART VCM v1b.

Keywords: CIM · architectures · memristive devices · memristors

1 Introduction

The conventional von Neumann architectures (such as CPU, GPU and TPU) are suffering from the three well-known architectural walls such as the so-called *memory-wall* [1]; not to mention the three technology walls CMOS technology (used to implement such architectures) is facing such as static power [2]. As a result, excessive time and energy are spent on moving massive amounts of data between the memory and data paths, which makes such architectures extremely energy-inefficient [3–5]. The explosion of data-intensive applications and their unprecedented demand for energy efficiency, from data centers to

This work was funded in part by EU's Horizon Europe research and innovation programme under grant agreement No. 101070374, in part by the Deutsche Forschungsgemeinschaft (SFB 917), and in part by the Federal Ministry of Education and Research (BMBF, Germany) in the project NEUROTEC II (project numbers 16ME0398K and 16ME0399).

C. Silvano et al. (Eds.): SAMOS 2023, LNCS 14385, pp. 437–450, 2023.
https://doi.org/10.1007/978-3-031-46077-7_29

energy-constrained edge devices, further exacerbate the challenges [6]. To overcome these challenges and significantly improve the efficiency, beyond von-Neumann Computation-In-Memory (CIM), in which computation and storage are integrated in the same physical location, has become a potential alternative for efficient computing mainly for edge devices [3,6].

Thus, CIM architectures based on memristive devices store the data while exploiting their inherent capability to perform computation on the stored data circumvents the costly data movement of von-Neumann based systems [5]. Memristive devices are a promising and relatively new type of device for CIM. They offer interesting opportunities, making them a viable addition to current applications such as machine learning. In addition, they also significantly improve new computing paradigms such as neuromorphic computing which represents a special case of CIM [7,8]. Enhanced hybrid systems based on the combination of memristive devices and complementary metal oxide semiconductor (CMOS) devices can offer significant benefits over conventional CMOS systems via the co-location of memory and computing. There exists a range of resistive switching devices that are considered for CIM such as Phase Change Memory (PCM) where the switching is based on changing the internal device structure between an amorphous phase and a crystalline phase [9] and Magnetoresistive RAM (MRAM) devices in which the resistive switching is based on the change of the magnetization direction in a ferromagnetic film [10]. Also, we have ReRAM devices which can be further classified as Electrochemical Metallization Memory (ECM), also called Conductive-Bridge RAM (CBRAM) and Valence Change Memory (VCM), also called Oxide-based RAM (OxRAM). For ReRAM devices the switching is based on local redox reactions. Due to the specific physical functionality of these devices they have to be individually considered for CIM. The device type considered in more detail in this work is non-volatile, bipolar and filamentary switching VCM devices [11,12].

This paper provides a broad overview of CIM architectures, circuits and devices highlighting state-of-the-art research in CIM. Particularly, the paper investigates memristive devices and their widespread application in neuromorphic computing. In this regard, ReRAMs are introduced in terms of their potential for novel computing paradigms. Moreover, we address design and non-ideality challenges of CIM. The rest of the paper is organized as follows: In Sect. 2 the relevant fundamentals on CIM and VCM devices is explained. Section 3 explains commonly investigated architectures for CIM. Section 4 then details circuit and device level considerations for CIM based on VCM devices followed by the discussion of design and non-ideality challenges in Sect. 5. Finally, the conclusion and future directions are presented in Sect. 6.

2 Background

2.1 CIM Basics

CIM is a computing paradigm where the operation execution happens within the memory where the data resides. Figure 1 shows a high-level micro-architecture

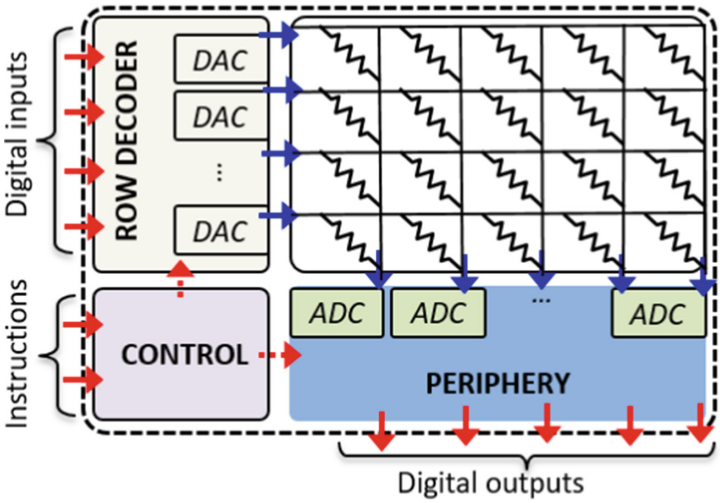

Fig. 1. CIM core architecture concept.

of a CIM crossbar, where memristive devices such as ReRAM devices are used at each crossbar junction. The communication to the crossbar is realized with the support of peripheral circuits which perform different functions depending on the targeted CIM architecture; for example input/output data format conversion may require Digital-to-Analog Conversion (DAC) in the row decoding part or Analog-to-Digital Conversion (ADC), dedicated sense amplifiers in the read path. The control block is responsible for the overall control of the CIM core operation.

2.2 CIM Benefits

Memristive CIM has many features that make it feasible to realize ultra-low power and energy-efficient computing [6]:

- **Practically zero leakage computing** [13]: The non-volatile nature of the resistive devices enables CIM to maintain the stored values in a leakage-free manner when it is not operating, which solves the leakage bottleneck of SRAM-based architectures.
- **Massive parallelism** [6]: CIM provides high parallelism as typically all columns in a crossbar can be accessed concurrently, leading to maximal parallelism. Moreover, the scalability of memristive device technology enables to increase the number of columns per crossbar, which in turn increases the degree of parallelism CIM can offer.
- **Near zero data bandwidth requirement** [14]: Integration of storage and computation in the same physical location circumvents the bandwidth bottleneck associated with the traditional computation-centered systems, which need significant data movement.

– **Extremely energy-efficient computing** [13]: The combination of non-volatility (near zero leakage), parallelism and near zero bandwidth requirement enables CIM to offer extremely energy-efficient computing.

2.3 VCM Devices and Circuits for CIM

Filamentary VCM switching is observed for two terminal devices consisting of a stack with a metal oxide like ZrO_2, Ta_2O_5 or HfO_2 which is sandwiched between two different metal electrodes [15]. One of the electrodes has a high work function and low oxygen affinity and therefore forms a Schottky contact with the oxide. As the main resistance change happens at this electrode it is called electronically active electrode (AE). The other electrode has a low work function and high oxygen affinity and forms an Ohmic contact with the oxide. It is therefore called ohmic electrode (OE). Underlying the resistive switching in VCM cells is the movement of charged oxygen vacancies inside the oxide due to an applied electrical field. An increase of the number of oxygen vacancies near the AE interface leads to a resistance reduction and is termed a SET operation, while a reduction of the number of oxygen vacancies near the AE increases the resistance and is called a RESET process. The cell state after the SET process is called the low resistance state (LRS) and the state after the RESET process is called the high resistance state (HRS) [11,12]. Before the VCM cells can be repeatedly switched they have to be electroformed, as the fabricated oxide is initially highly insulating. During this electroforming process, the oxide layer is locally reduced and oxygen vacancies are generated, decreasing the resistance of the devices. Today, forming is mostly carried out in the SET direction with relatively slow voltage sweeps (V/s) at voltages between 2 V–4 V [16,17].

The variability of VCM devices arises from the stochastic nature of the switching process [18,19]. It has consequences on the circuit and architectural level design of CIM applications based on these devices [20–22]. Observed variability in experiments or simulations can be classified as switching variability or read variability, depending on whether it is observed during a switching or a reading process. In addition, it can be classified as device-to-device (d2d) or cycle-to-cycle (c2c) variability, depending on whether it was observed between multiple devices or in the same device during multiple switching cycles [23]. Read variability, sometimes also called read noise or random telegraph noise (RTN) [24,25] describes the effect that during the read operation the current in VCM cell shows random fluctuations and jumps with different jump heights. The different current jumps were associated with oxygen vacancies jumping at different positions in the plug or disc region [24]. Switching and read variability are critical effects influencing the performance of VCM cells in computing applications. However, different computing applications are effected differently as will be discussed in Sect. 4 in detail. These differences concern first of all which type of variability is relevant for a certain application and then secondly, which amount of variability can be tolerated.

Circuit-level compact models are used for the investigation of computing applications. For the results shown in this work the Jülich Aachen Resistive

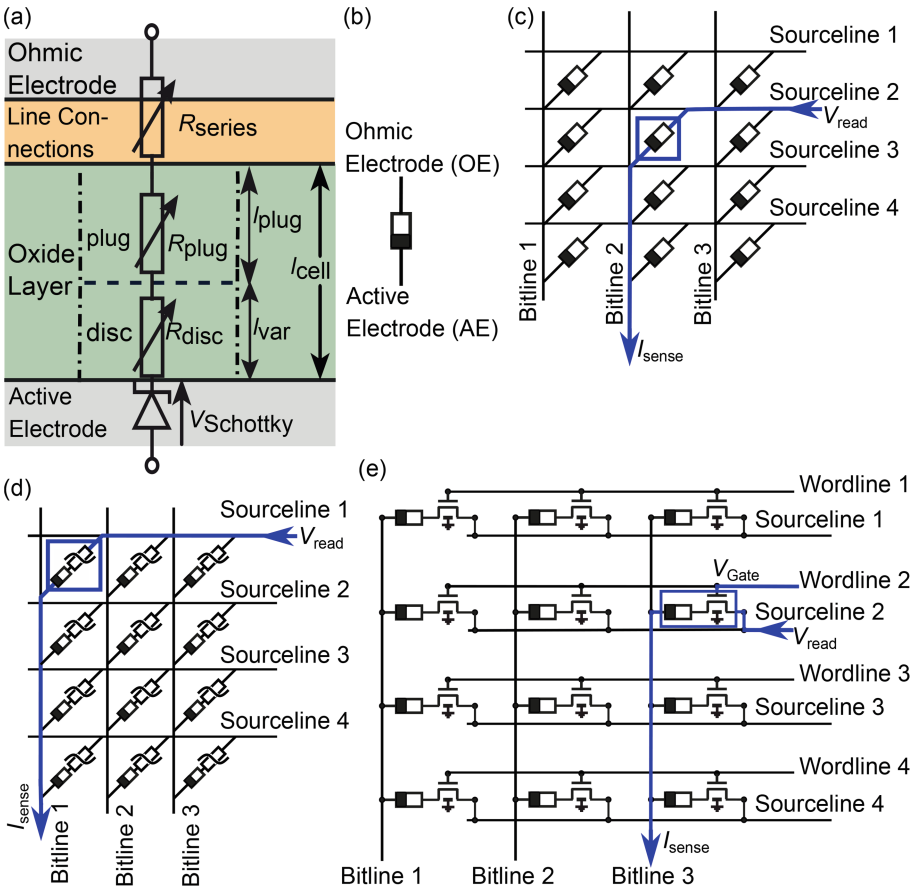

Fig. 2. (a) shows the Equivalent Circuit diagram (ECD) of the JART VCM v1b compact model with the circuit symbol shown in (b). A passive 1R crossbar array (c) is composed of horizontal Sourcelines and vertical Bitlines with a VCM cell at each crossing point. The 1S1R array (d) has an additional selector element in series with the VCM cell at each crossing point of Sourceline and Bitline. In the 1T1R array each VCM cell is connected in series with a transistor (usually an n-type field effect transistor (NMOS) due to the higher charge carrier mobility) (e). To access the elements of the 1T1R array an additional Wordline is required that sets the voltage at the gate of the transistors. Exemplary readout schemes for individual cells are highlighted in blue. (Color figure online)

Switching Tools (JART) VCM v1b compact model is used which is a sophisticated and physically motivated model for filamentary switching VCM cells. In the past, it has been used to describe several key properties of VCM devices such as the highly nonlinear SET and RESET switching kinetic [26,27], the multilevel switching in the RESET direction [28] and nonideality effects like read noise [24]. Its equivalent circuit diagram (ECD) corresponds to the general

metal-oxide-metal structure of a VCM cell and is shown in Fig. 2 (a). The more commonly used circuit symbol is shown in Fig. 2 (b).

VCM cells are often organised in array structures or crossbars. Passive arrays were proposed such as 1R (1 resistive element) arrays as shown in Fig. 2 (c). Another passive array structure are 1 Selector 1 resistive element (1S1R) arrays Fig. 2 (d). While passive arrays allow for the highest possible integration density of $4\,F^2$ (where F denotes the minimum feature size of the used technology), they suffer from issues such as sneak paths and programming difficulties (1R) or limited multilevel capabilities (1R and 1S1R) [29,30]. Therefore, most works focus on active 1 Transistor 1 resistive element (1T1R) arrays [31,32]. A 1T1R array structure is shown in Fig. 2 (e).

3 CIM Architectures

3.1 CIM Architecture Units

As shown in Fig. 3(b), a CIM core has two main architectural units: (1) Memory array commonly known as crossbar array unit and periphery unit. The crossbar array stores the data, and can perform any logic or arithmetic operation. Similarly, the periphery unit converts input/output data formats between analog and digital. Moreover, the periphery unit can also be used to perform basic logical and arithmetic operations.

Crossbar Array: Different applications use primitive computational units such as multiply and accumulate (MAC) extensively to perform matrix-matrix multiplication (MMM) with large operand sizes [33,34]. Such primitive units can be easily mapped into a memristive crossbar array and perform their operation

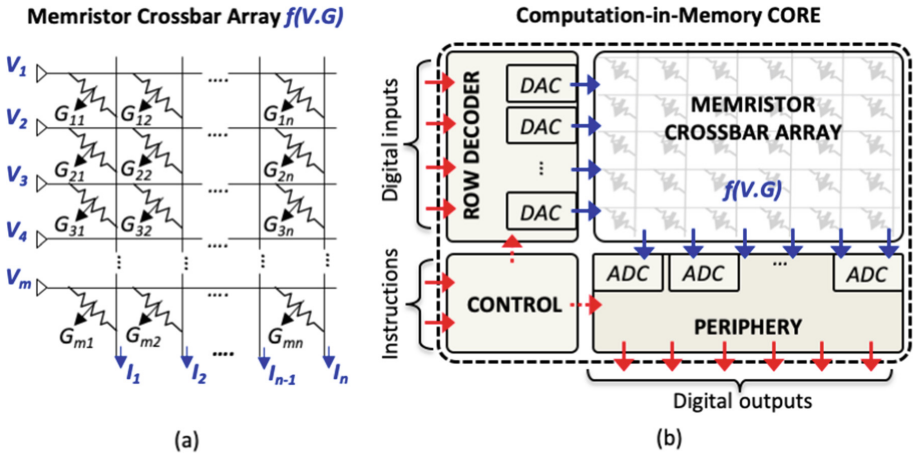

Fig. 3. CIM architecture (a) ReRAM based crossbar operation demo (b) CIM core architecture i.e., Periphery + crossbar array

e.g., MMM in the crossbar unit of a CIM. Figure 3(a), shows a subset of MMM operation *i.e.*, vector-matrix multiplication (VMM) using CIM crossbar array. From Fig. 3(a) it can be observed that the VMM is performed by applying a voltage vector $V = V_j$ (where $j \in \{1, m\}$) to a memristive-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.

Periphery: A CIM core needs some major modifications to accommodate analog-based computing, as shown in Fig. 3(b). The circuit blocks comprising the periphery that supports the bitcell array need to be modified to support CIM operations. For example, the following is needed to perform VMM operation in CIM: 1) Row-decoder becomes complex as it involves enabling several rows in parallel. 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 need to be replaced by analog-to-digital converters (ADCs). 3) Control block needs to deal with complex instructions such as handling intricacies of multi-operand VMM operations.

3.2 Potential CIM Applications

CIM architectures can be applied in different application segments which have extreme demand in terms of storage, energy and computation efficiency. This subsection presents some of the application domains in which CIM can be applied [35].

Neuromorphic Computing. Neuromorphic computing is one of the application domains which can significantly benefit from CIM architecture. The main reason for this is the fact that the main operation employed by neuromorphic systems involves intensive Matrix-Matrix Multiplication (MMM) or Vector-Matrix Multiplication (VMM). Since both MMM and VMM kernels can be easily accelerated using CIM architecture, neuromorphic computing can achieve substantial improvement in energy efficiency and alleviate data movement problems by employing CIM.

Sparse Coding. Sparse coding of information is a powerful means to perform feature extraction on high dimensional data and it is of vital importance for a wide range of application segments such as object recognition, computer vision, signal processing and etc. Sparse coding can be used to implement energy-efficient bio-inspired neuromorphic applications as well. Since sparse coding mainly relies on bulky matrix-vector multiplication operation, it can directly benefit from CIM to accelerate the matrix-vector multiplication operation efficiently.

Threshold Logic. Threshold logic is a basic operation that uses a threshold gate which takes n inputs (x_1,x_2, . . ., x_n) and generates single output y. A threshold logic has a threshold θ and each input x_i is associated with a weight w_i. Since weighted sum operation is the core operation involved in threshold logic, it can be easily accelerated using CIM.

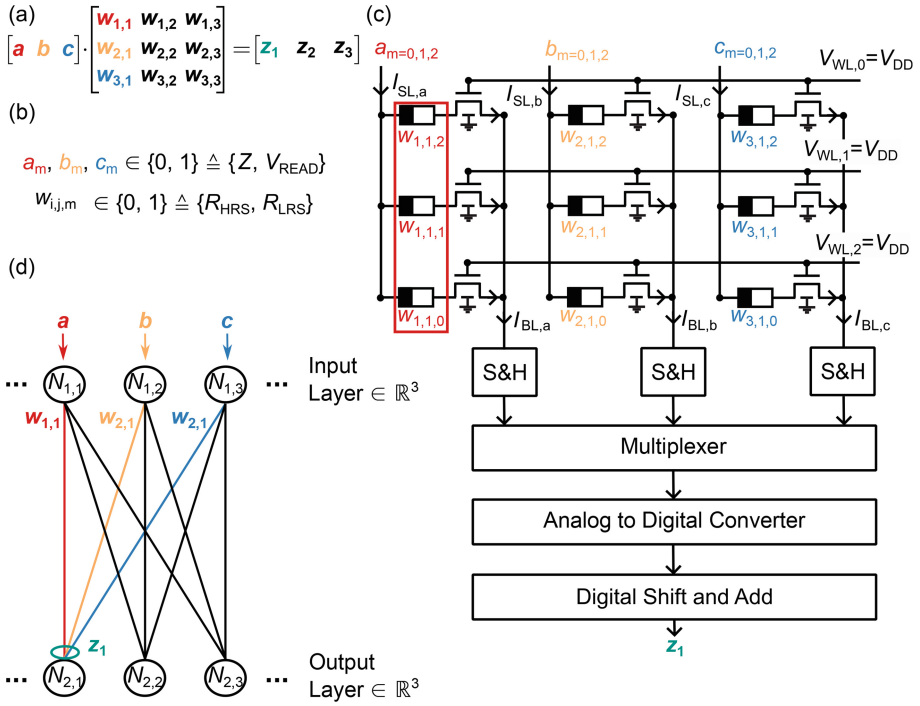

Fig. 4. (a) shows the Vector-Matrix-Multiplication of a 1×3 vector with a 3×3 matrix. In (b) the input data signals are assigned their physical correspondence with a '1' being encoded as V_{READ} and a '0' being encoded by setting the input lines (Sourcelines) high ohmic. A '1' in the weight matrix is encoded by a device in the LRS and a '0' is represented by a device in the HRS. (c) shows an exemplary circuit-level architecture of the 1T1R crossbar. Each element of the input vector and the weight matrix is assumed to consist of three bits. The input vector is applied via the Sourcelines over three time cycles and the weight matrix is stored column-wise with the first three columns encoding the first column of the weight matrix. The results of the dot product operation is encoded in the Bitline current (I_{BL} and temporarily stored in a Sample & Hold element before it is multiplexed to the ADC and then shifted and added to align the dot products for the correct bit position in the multiplication. In (d) an exemplary artificial neural network structure is shown, that can be mapped by the VMM in (a).

4 Circuits and Devices for CIM

4.1 Vector-Matrix-Multiplication Accelerators

Machine learning has been a rapidly growing field in the last decade, driven by improvements in the algorithms and network architectures as well as by improving the underlying hardware [36]. It can be used to extract information out of large amounts of data generated in contexts such as the internet of things or self-driving cars [37]. Machine learning algorithms are trained to perform certain tasks through exposure to previous examples of how to perform a task correctly. During this training phase, they adapt their internal parameters or weights according to a teacher signal with labeled data. This procedure is called supervised learning [38]. When the training is finished, the network is able to classify or respond to unseen data input with a high-accuracy answer. This second phase is called the inference phase. During the training and the inference phase, a common type of operations are Vector-Matrix (VMM) or Matrix-Matrix Multiplication (MMM) constructed from multiply-accumulate (MAC) operations [39,40]. While in conventional computer architectures these operations are associated with heavy data transfer between the memory and the CPU or GPU, CIM using VCM cells reduces the data transfer by allowing the memory arrays to perform both inference and training in the same physical location [41].

Figure 4 explains the mapping of a VMM operation to a 1T1R crossbar array and a neural network. Figure 4 (a) shows the 1×3 input vector multiplied with a 3×3 matrix resulting in a 1×3 output vector. Each element of the input vector and input matrix consists of three bits. Each element of the output vector then contains six bits to map all possible input combinations. The bits of the input vector are converted to high ohmic to represent a '0' and to V_{READ} to represent a '1'. The weight bits are represented by a device in the HRS state for a '0' and by a LRS device for a '1' as shown in Fig. 4 (b). Figure 4 (c) shows the circuit of the 1T1R crossbar array, representing the first column of the weight matrix. The input vector is applied over three time cycles represented by the indices '0', '1' and '2' of the input vector components. During the operation the Wordlines connecting to the transistor gates have to be activated by applying V_{DD} to them. In each cycle an output current $I_{BL,i}$ is produced based on the result of the MAC operation between input vector bit and weight. This current is stored on a Sample & Hold element (S&H). This intermediate storage is required because in many cases it is not possible to provide one Analog to Digital Converter (ADC) per column of the array. In that case the MAC results of the different columns have to be temporarily stored and multiplexed to the available ADCs [42,43]. After the ADC stage the result might also require shifting and adding to align the result at the correct bit position. The result of the first time cycle does not have to be shifted, the results of the second cycle have to be shifted by one and the results of the third cycle have to be shifted by two.

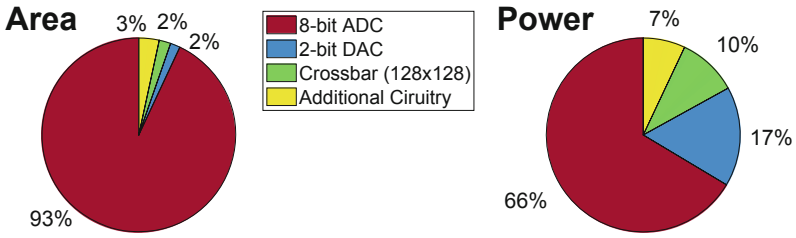

Fig. 5. Area and Power share of CIM design blocks [34].

5 CIM Challenges

5.1 Design Challenges

In CIM architectures, the operations are performed in an analog manner as shown in Fig. 3, and the result is converted to a digital signal using Analog-to-Digital Converter (ADC) at the periphery of the CIM architecture. However, the conversion performed by ADC is very critical and challenging due to 1) Analog signals have low noise margin and hence, can lead to erroneous output; 2) Analog computation heavily relies on the device strenghts of the memristive and CMOS devices along the column, therefore their variations induce variation in output current; 3) Quantization error in ADC increases as we increase the number of levels or reduce the resolution. In addition, area/power increases drastically as we do so and speed reduces along with accuracy. For instance, substantiating the importance of ADC design in CIM-based implementation of machine learning algorithms such as CNN and DNN, Fig. 5 shows that the ADC alone typically dominates CIM die area (>90%) and power consumption (>65%). Thus, efficient ADC design is imperative to efficiently deploy CIM architecture in different resource-constrained systems.

5.2 Non-ideality Challenges

While CIM using VCM cells is a promising new field, there remain several challenges for industry-level adoption. Those challenges depend on the type of application as each application will put different requirements on the devices. In the case of machine learning accelerators or VMM accelerators, the VCM cells are initially programmed during the training phase and then read out over a long time scale during the inference [22,44]. The programming is achieved via program-verify algorithms [45–47]. These algorithms adapt the resistance of individual 1T1R cells by repeatedly performing SET and RESET operations to bring the resistance into a previously specified range. After the programming the resistances should be constant over time under read stress or without any voltage applied. The readout operation is then affected by the read disturb effect and the read noise effect. Read disturb is a directed and time-dependent accumulative effect. It describes the change of the device state due to the applied read

voltage. This resistance change happens towards lower resistances if the read voltage is applied in the SET direction or towards higher resistances if the read voltage is applied in the RESET direction. It is an accumulative process, therefore, over many read operations it will increase in magnitude. Additionally, it is more pronounced at higher voltages. Through our detailed experimental and theoretical analysis it was found that the effect is stronger pronounced in the SET direction, where even an information loss due to an abrupt switching from the HRS to the LRS is possible. If the readout is done in the RESET direction, the resistance change is more gradual and weaker at comparable voltages. From a long-term stability point of view it is therefore more favorable to read in the RESET direction to prevent the negative effects of read disturb [44]. Read noise or RTN is an undirected and not accumulative process whereby the read-out current fluctuates over time. The effect is stronger at higher resistances, giving them a higher inaccuracy [24,25]. The occurrence of read noise depends on the dominant electron conduction mechanism of the VCM cells. While there also exist VCM switching devices without read noise those devices are often based on less industrial fab compatible material systems like $SrTiO_3$ or TiO_x [48]. The typical materials considered for industrial applications like HfO_2 or Ta_2O_5 both show read noise. Reducing the impact of read noise is then only possible by using lower ohmic devices which however also has negative effects like a higher energy consumption. It should also be noted that read disturb and read noise are not correlated between different devices.

6 Conclusion and Future Direction

CIM has the potential for a computing paradigm shift from the traditional von-Neumann architecture based computing. This paper presented the overview and cross-layer design aspects of memristive-based CIM designs. The paper first discussed devices and circuits for CIM design followed by the discussion on CIM architectures. The paper also highlighted different design and non-ideality challenges which are roadblocks for the widespread applicability of CIM designs. Therefore, addressing those design and non-ideality challenges is prime importance to harness the full potential of CIM and its widespread applicability.

References

1. Patterson, D.A.: "Future of computer architecture," in Berkeley EECS Annual Research Symposium (BEARS). College of Engineering, UC Berkeley, US (2006)
2. Hamdioui, S., et al.: Memristor for computing: myth or reality?. In: DATE (2017) *DATE*, 2017.
3. Gebregiorgis, A., et al.: Tutorial on memristor-based computing for smart edge applications. Memories-Mater. Devices Circ. Syst. **4**, 100025 (2023)
4. Diware, S., et al.: Accurate and energy-efficient bit-slicing for RRAM-based neural networks. TETCI **7**(1), 164–177 (2022)
5. Gebregiorgis, A., et al.: A survey on memory-centric computer architectures. JETC **18**(4), 1–50 (2022)

6. Singh, A., et al.: Low-power memristor-based computing for edge-AI applications. In: ISCAS (2021)
7. Zidan, M.A., Strachan, J.P., Lu, W.D.: The future of electronics based on memristive systems. Nat. Electron. **1**, 22–29 (2018)
8. Shalf, J.: The future of computing beyond Moore's Law. Phil. Trans. R. Soc. A **378**, 20190061 (2020)
9. Wuttig, M., Yamada, N.: Phase change materials for rewriteable data storage. Nat. Mater. **6**, 824 (2007)
10. Apalkov, D., Dieny, B., Slaughter, J.M.: Magnetoresistive random access memory. Proc. IEEE **104**, 1796–1830 (2016)
11. Dittmann, R., Menzel, S., Waser, R.: Nanoionic memristive phenomena in metal oxides: the valence change mechanism. Adv. Phys. **70**(2), 155–349 (2022)
12. Waser, R., Dittmann, R., Staikov, G., Szot, K.: Redox-based resistive switching memories - nanoionic mechanisms, prospects, and challenges. Adv. Mater. **21**(25–26), 2632–2663 (2009)
13. Yu, J., et al.: The power of computation-in-memory based on memristive devices. In: ASP-DAC (2020)
14. Kanerva, P.: Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors. Cogn. Comput. **1**, 139–159 (2009)
15. Yang, J.J., et al.: Memristive switching mechanism for metal/oxide/metal nanodevices. Nat. Nanotechnol. **3**(7), 429–433 (2008)
16. Hardtdegen, A., Torre, C.L., Cüppers, F., Menzel, S., Waser, R., Hoffmann-Eifert, S.: Improved switching stability and the effect of an internal series resistor in HfO_2/TiO_x bilayer ReRAM cells. IEEE Trans. Electron Devices **65**(8), 3229–3236 (2018)
17. Wiefels, S., von Witzleben, M., Hüttemann, M., Böttger, U., Waser, R., Menzel, S.: Impact of the ohmic electrode on the endurance of oxide based resistive switching memory. IEEE Trans. Electron Devices **68**(3), 1024–1030 (2021)
18. Rieck, J.L., Hensling, F.V., Dittmann, R.: Trade-off between variability and retention of memristive epitaxial $SrTiO_3$ devices. APL Mater. **9**(2), 21110/1-7 (2021)
19. Kopperberg, N., Wiefels, S., Liberda, S., Waser, R., Menzel, S.: A consistent model for short-term instability and long-term retention in filamentary oxide-based memristive devices. ACS Appl. Mater. Interfaces. **13**(48), 58066–58075 (2021)
20. Kim, T., et al.: Spiking neural network (snn) with memristor synapses having non-linear weight update. Front. Comput. Neurosci. **15**, 646125 (2021)
21. Quesada, E.P., et al.: Experimental assessment of multilevel RRAM-based vector-matrix multiplication operations for in-memory computing. IEEE Trans. Electron Devices **70**, 2009–2014 (2023)
22. Bengel, C., Dixius, L., Waser, R., Wouters, D.J., Menzel, S.: Bit slicing approaches for variability aware ReRAM CIM macros. IT - Inf. Technol. **65**, 3–12 (2023)
23. Wiefels, S.; Reliability aspects in resistively switching valence change memory cells. PhD thesis (2021)
24. Wiefels, S., Bengel, C., Kopperberg, N., Zhang, K., Waser, R., Menzel, S.: HRS instability in oxide based bipolar resistive switching cells. IEEE Trans. Electron Devices **67**(10), 4208–4215 (2020)
25. Puglisi, F.M., Zagni, N., Larcher, L., Pavan, P.: Random telegraph noise in resistive random access memories: compact modeling and advanced circuit design. IEEE Trans. Electron Devices **65**(7), 2964–2972 (2018)
26. Cüppers, F., et al.: Exploiting the switching dynamics of HfO_2-based ReRAM devices for reliable analog memristive behavior. APL Mater **7**(9), 91105/1-9 (2019)

27. Bengel, C., et al.: Variability-aware modeling of filamentary oxide based bipolar resistive switching cells using SPICE level compact models. IEEE Trans. Circ. Syst. I: Regul. Pap. (TCAS-1) **67**(12), 4618–4630 (2020)
28. Bengel, C., Siemon, A., Rana, V., Menzel, S.: Implementation of multinary Lukasiewicz logic using memristive devices. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Korea, 22–28 May 2021. IEEE (2021)
29. Bayat, F.M., Prezioso, M., Chakrabarti, B., Nili, H., Kataeva, I., Strukov, D.: Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. Nat. Commun. **9**(1), 2331 (2018)
30. Bae, W., Yoon, K.J.: Comprehensive read margin and BER analysis of one selector-one memristor crossbar array considering thermal noise of memristor with noise-aware device model. IEEE Trans. Nanotechnol. **19**, 553–564 (2020)
31. Kiani, F., Yin, J., Wang, Z., Yang, J.J., Xia, Q.: A fully hardware-based memristive multilayer neural network. Sci. Adv. **7**(48), eabj4801/1-8 (2021)
32. Sahay, S., Bavandpour, M., Mahmoodi, M.R., Strukov, D.: Energy-efficient moderate precision time-domain mixed-signal vector-by-matrix multiplier exploiting 1T–1R arrays. IEEE J. Exploratory Solid-State Comput. **6**, 18–26 (2020)
33. Velasquez, A., et al.: Parallel Boolean matrix multiplication in linear time using rectifying memristors. In: ISCAS (2016)
34. Shafiee, A., et al.: ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. ISCAS **44**(3), 14–26 (2016)
35. Hamdioui, S., et al.: Applications of computation-in-memory architectures based on memristive devices. In: DATE (2019)
36. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015)
37. Sze, V., Chen, Y.H., Emer, J., Suleiman, A., Zhang, Z.: Hardware for machine learning: challenges and opportunities. In: 2018 IEEE Custom Integrated Circuits Conference (CICC), pp. 1–8 (2018)
38. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin (2006)
39. Zahedi, M., Mayahinia, M., Lebdeh, M.A., Wong, S., Hamdioui, S.: Efficient organization of digital periphery to support integer datatype for memristor-based CIM. In: 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 216–221 (2020)
40. Feinberg, B., Vengalam, U.K.R., Whitehair, N., Wang, S., Ipek, E.: Enabling scientific computing on memristive accelerators. In: 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), pp. 367–382 (2018)
41. Amirsoleimani, A., et al.: In-memory vector-matrix multiplication in monolithic complementary metal-oxide-semiconductor-memristor integrated circuits: design choices, challenges, and perspectives. Adv. Intell. Syst. **2**, 2000115 (2020)
42. Shafiee, A., et al.: ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pp. 14–26 (2016)
43. Li, C., et al.: CMOS-integrated nanoscale memristive crossbars for CNN and optimization acceleration. In: 2020 IEEE International Memory Workshop (IMW), pp. 1–4 (2020)
44. Bengel, C., et al.: Reliability aspects of binary vector-matrix-multiplications using ReRAM devices. Neuromorphic Comput. Eng. **2**(3), 034001 (2022)
45. Le, B.Q., et al.: Radar: a fast and energy-efficient programming technique for multiple bits-per-cell RRAM arrays. IEEE Trans. Electron Devices **68**(9), 4397–4403 (2021)

46. Milo, V., et al.: Accurate program/verify schemes of resistive switching memory (RRAM) for in-memory neural network circuits. IEEE Trans. Electron Devices **68**, 3832–3837 (2021)
47. Perez, E., Mahadevaiah, M.K., Quesada, E.P., Wenger, C.: Variability and energy consumption tradeoffs in multilevel programming of RRAM arrays. IEEE Trans. Electron Devices **68**, 2693–2698 (2021)
48. Schnieders, K., et al.: Effect of electron conduction on the read noise characteristics in ReRAM devices. APL Mater. **10**(10), 101114 (2022)