Tutorial on memristor-based computing for smart edge applications

Gebregiorgis, Anteneh; Singh, Abhairaj; Yousefzadeh, Amirreza; Wouters, Dirk; Bishnoi, Rajendra; Catthoor, Francky; Hamdioui, Said

DOI
10.1016/j.memori.2023.100025

Publication date
2023

Document Version
Final published version

Published in
Memories - Materials, Devices, Circuits and Systems

Citation (APA)
Gebregiorgis, A., Singh, A., Yousefzadeh, A., Wouters, D., Bishnoi, R., Catthoor, F., & Hamdioui, S. (2023). Tutorial on memristor-based computing for smart edge applications. *Memories - Materials, Devices, Circuits and Systems*, *4*, Article 100025. https://doi.org/10.1016/j.memori.2023.100025

Contents lists available at ScienceDirect

Memories - Materials, Devices, Circuits and Systems

journal homepage: www.elsevier.com/locate/memori

Tutorial on memristor-based computing for smart edge applications[☆]

Anteneh Gebregiorgis [a,*], Abhairaj Singh [a], Amirreza Yousefzadeh [b], Dirk Wouters [c], Rajendra Bishnoi [a], Francky Catthoor [d], Said Hamdioui [a]

[a] Computer Engineering Lab, Delft University of Technology, Delft, The Netherlands
[b] IMEC, The Netherlands
[c] RWTH Aachen University, Germany
[d] IMEC, Belgium

ARTICLE INFO

Keywords:
Edge-AI
Memristor
Computation-In-Memory

ABSTRACT

Smart computing on edge-devices has demonstrated huge potential for various application sectors such as personalized healthcare and smart robotics. These devices aim at bringing smart computing close to the source where the data is generated or stored, while coping with the stringent resource budget of the edge platforms. The conventional Von-Neumann architecture fails to meet these requirements due to various limitations e.g., the memory-processor data transfer bottleneck. Memristor-based Computation-In-Memory (CIM) has the potential to realize such smart edge computing for data-dominated Artificial Intelligence (AI) applications by exploiting both the inherent properties of the architecture and the physical characteristics of the memristors. This paper discusses different aspects of CIM, including classification, working principle, CIM potentials and CIM design-flow. The design-flow is illustrated through two case studies to demonstrate the huge potential of CIM in realizing orders of magnitude improvement in energy-efficiency as compared to the conventional architectures. Finally future challenges and research directions of CIM are covered.

1. Introduction

Edge computing has emerged as a potential alternative to traditional server and cloud computing which rely on powerful computing hardware such as GPU [1]. Edge computing enables new types of applications (such as personalized healthcare, autonomous driving, etc.) with the advantage of implementing the required Artificial Intelligence (AI) solutions as close as possible to the data sources and personalized to the end-user [2]. Consequently, the market share of edge AI has increased tremendously, and it is expected to grow further [3,4]. For instance, the market value of edge computing hardware will exceed $200 billion by the year 2025 and the demand for edge computing will increase in volume and use case, creating significant opportunities in different sectors. For example, Fig. 1 shows the U.S. edge computing market size for different components (Hardware, Software, Service and Platforms). The boom in edge computing market value is driven mainly by the increase in connectivity of devices that produce gazillions of data and increasing demand for real-time processing and decision making. Thus, these two factors are increasingly pushing computing towards edge devices which ultimately drives the rise in the edge hardware market. However, the conventional Von-Neumann based architectures (such as CPU, GPU and TPU) are suffering from the three well-known architectural walls such as the so-called *memory-wall* [5]; not to mention the three technology walls CMOS technology (used to implement such architectures) is facing such as static power [6]. As a result, excessive time and energy are spent in moving massive amount of data between the memory and data paths, which makes such architectures to be extremely energy-inefficient [7–9]. These challenges are therefore hindering the widespread application of edge computing. Both architectural and technological walls are making the design of efficient edge computing engines extremely difficult. Therefore, there is a clear demand for *alternative* architectures and/or technologies to realize resource and energy-efficient computing engines for edge-AI applications.

Existing works on alternative architectures can be classified on those based on traditional CMOS [10–14], and those based on emerging device technologies such as memristors [15–18]. The first class consists of computing engines that go beyond Von-Neumann, but fully implemented using *volatile* CMOS technologies; examples are neuromorphic engines such as IBM TrueNorth [13], Intel Lohi [10,11], Morphic [19],

☆ This work was supported in part by the EU H2020 grant "DAIS" that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007273.

* Corresponding author.
E-mail addresses: A.B.Gebregiorgis@tudelft.nl (A. Gebregiorgis), A.Singh-5@tudelft.nl (A. Singh), Amirreza.Yousefzadeh@imec.nl (A. Yousefzadeh), wouters@iwe.rwth-aachen.de (D. Wouters), R.K.Bishnoi@tudelft.nl (R. Bishnoi), Francky.Catthoor@imec.be (F. Catthoor), S.Hamdioui@tudelft.nl (S. Hamdioui).

https://doi.org/10.1016/j.memori.2023.100025
Received 7 November 2022; Received in revised form 7 December 2022; Accepted 13 January 2023

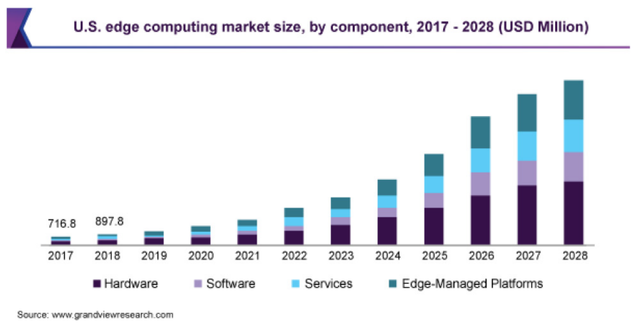


Fig. 1. Projection of edge computing market growth in U.S. divided among edge hardware, software, services and platforms from the year 2017–2028.

BrainScales [20], and SpiNNaker [12]. Hence, their architectures face common challenges [7,8,21]; e.g., they are power hungry, expensive (mainly targeting cloud and data-centres), suffer from high leakage power (due to volatile CMOS nodes), occupy large silicon area, as huge amount of memory is distributed among multiple processors (neurons) on a single chip and the SRAMs used for synapses require six or more transistors per synapse, communication with the off-chip memory consumes huge memory bandwidth, etc. All these challenges make them unsuitable for edge applications which are too demanding e.g., in term of energy-efficiency (order of ≈0.1fJ/op) and cost. On the other hand, the second class makes use of alternative architectures based on emerging non-volatile device technologies, commonly known as memristor-based Computation-In-Memory (CIM). CIM has the potential to break the aforementioned challenges (due to the inherent nature of the architecture and the devices used to realize it) and deliver energy and cost-efficient implementations of computing engines suitable for edge-AI applications [22–24].

Such memristor-based CIM architectures use non-volatile devices to store data while exploiting their inherent capability to perform computation on the stored data which enables them to circumvent the costly data movement of the conventional Von-Neumann based systems [25]. Emerging non-volatile memory technologies, such as Spin-Torque Transfer Magnetic Random Access Memory (STT-MRAM), Phase-Change Memory (PCM), and Resistive Random Access Memory (RRAM) serve as the building blocks in realizing CIM [26]. As a result, different CIM-based accelerators have been developed to target different domains, applications and kernels such as neuromorphic computing [27–31]. Although these works have demonstrated (at the small scale) the potential of CIM in realizing energy-efficient computing, achieving the maximum attainable potential of CIM is still an open question. For instance, using an appropriate holistic approach to design CIM accelerator for a dedicated application could significantly improve not only the overall energy-efficiency, but also result in cost-effective and robust hardware implementation [32]. Therefore, understanding the targeted application, identifying the critical kernels to be accelerated, selecting the best CIM microarchitecture for the critical kernels and applying proper design methodology are examples of aspects which contribute to the overall CIM efficiency.

This paper presents a holistic design-flow that is essential for the design of efficient CIM accelerator and demonstrates it on two case studies. The paper first discusses the CIM concept, including its classification and its potential in realizing energy-efficient computing. Then, a detailed discussion on CIM design-flow is presented by starting from system level design all the way down to circuit design and device technology selection. Finally, the paper presents two case studies to demonstrate the potential of CIM in accelerating data and computation intensive applications.

The remainder of the paper is organized as follows. Section 2 presents the background of CIM such as the working principle, CIM classification, device technologies for CIM, and potential CIM applications. Section 3 presents the design-flow of CIM accelerator. Sections 4 and 5 present the CIM case studies using database query and image classification applications, respectively. Finally, Section 6 discusses the challenges and future directions of CIM.

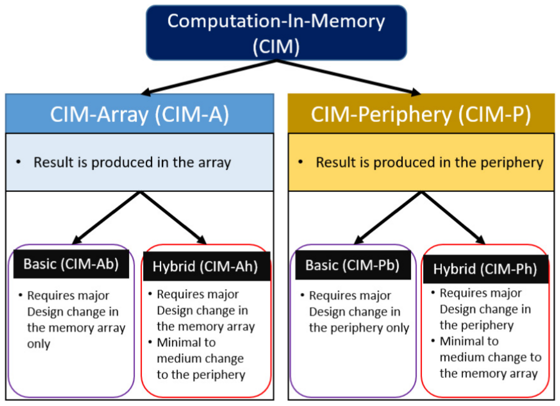

Computation-In-Memory (CIM)
CIM-Array (CIM-A)
CIM-Periphery (CIM-P)
Result is produced in the array
Result is produced in the periphery
Basic (CIM-Ab)
Requires major Design change in the memory array only
Hybrid (CIM-Ah)
Requires major Design change in the memory array
Minimal to medium change to the periphery
Basic (CIM-Pb)
Requires major Design change in the periphery only
Hybrid (CIM-Ph)
Requires major Design change in the periphery
Minimal to medium change to the memory array

Fig. 2. Classification of CIM architectures.

2. What is CIM all about?

This section first presents the basic concept and classification of CIM architecture followed by the discussion of device technologies used for CIM. Then, a high level memristor-based CIM architecture is presented together with its benefits. Finally, the potential applications of CIM are discussed.

2.1. CIM basics and classification

Computation-In-Memory (CIM) is a computing paradigm which integrates the computation and storage in the same physical location. The integration of computation and storage location enables CIM to overcome the data transfer bandwidth challenge of conventional architectures and unlock new potential for efficient computing. It is worth stressing the fact that CIM architectures perform computations *within the memory core*. As this consists of a memory array and the peripheral circuits, CIM architectures can be divided into two basic sub-classes depending on *where* the result of the computation is produced [25] as shown in Fig. 2; these sub-classes can be also combined into many hybrid combinations.

- CIM-Array (CIM-A): the computation result is produced within the memory array. CIM-A architectures require always a redesign of cells to support logic or arithmetic operations, as the conventional memory cell dimensions and their embedding in the bit- and wordline structure are optimized only for storage and typical read/write accesses. Examples of CIM-A architectures are presented in [33–35].
- CIM-Periphery (CIM-P): in CIM-P class, the computation result is produced within the peripheral circuit. CIM-P architectures require significant design changes in the memory peripheral circuits and these may typically contain dedicated circuits such as customized sense amplifiers. Typical examples of CIM-P architectures contains logical operations and vector matrix multiplications [36–38].

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 medium changes are required in the peripheral circuit (CIM-Ah) or vice versa (CIM-Ph) [39].

Table 1
Comparison of bit-cell design metrics for various CIM flavours.
Source: Data obtained from [47,48].

Metrics	CIM flavors with various memory technology					
	SRAM CIM	DRAM CIM	Flash CIM	RRAM CIM	MRAM CIM	PCM CIM
Size (F^2)	120–150	10–30	10–30	10–30	10–30	10–30
Volatility	Yes	Yes	No	No	No	No
Write energy	∼fJ	∼10 fJ	∼100 pJ	∼1 pJ	∼1 pJ	∼10 pJ
Write speed	∼1 ns	∼10 ns	0.1–1 ms	∼10 ns	∼5 ns	∼10 ns
Read speed	∼1 ns	∼3 ns	∼100 ns	∼10 ns	∼5 ns	∼10 ns
Endurance	10^{16}	10^{16}	$10^4 - 10^6$	10^7	10^{15}	10^{12}
Scalability	Medium	Medium	Medium	High	High	High

2.2. Device technologies for CIM

CIM can be realized using both conventional and emerging memory technologies. The memory technologies used for CIM can be broadly classified as charge-based memories and non-charge-based memories. In charge-based memories such as, Dynamic Random Access Memory (DRAM), Static Random Access Memory (SRAM) and Flash, information is stored through the presence of charge. Whereas, the non-charge-based memories include different types of storage elements distinguished by their physical mechanism; these includes resistive [40], magnetic memories [41,42], or even phase change memories [43]. The unique characteristics associated with each device technology for CIM are illustrated in Table 1.

2.2.1. CIM using charge-based memories

The volatile SRAM and DRAM memories can be used to build CIM architecture. An SRAM-based architecture has the benefit of speed due to faster SRAM accesses [44], while DRAM based CIM has a significant advantage in terms of density [45]. Due to their volatility, both SRAM and DRAM based CIM flavors face serious power dissipation problems. On the contrary, Flash based CIM has a non-volatile memory that uses a floating gate transistor which has a charge trapping mechanism. Flash based CIM has density benefit over DRAM based CIM as Flash uses only a single transistor. However, it requires high voltage and considerably long duration to write a value [46].

2.2.2. CIM using non-charge-based memories

CIM implementation based on non-charge-based memories commonly known as memristors. Memristor-based CIM flavors store the information in the form of resistance states, which can be in a high resistance or a low resistance states [6]. The resistance state can be changed using reset or set electrical pulses [49]. Non-charge-based memory technologies (memristor-based CIM) have several key advantages over charge-based memory technologies such as, zero leakage, non-volatility, density and scalability. However, non-charge-based memory technologies have relatively higher write energy and longer read and write latency than their charge-based counterparts [47,48].

Due to its different advantages CIM using non-charge-based memories outperforms CIM using charge-based memories, at least in terms of energy efficiency and cost [50]. Therefore, in the rest of this paper we will focus on CIM using non-charge-based memories, or the so called memristor-based CIM.

2.3. Memristor-based CIM

Fig. 3 shows a high level micro-architecture of memristor based CIM. The storage and computation are integrated together in a crossbar array structure where memristor device is 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 row decoding equipped with Digital-to-Analog

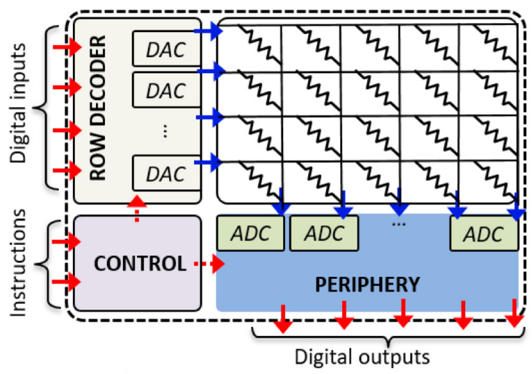


Fig. 3. CIM core architecture concept.

Conversion (DAC). Reading the memory array during the operation execution may require Analog-to-Digital Conversion (ADC) or dedicated sense amplifiers, etc. The control block is responsible for the overall control of the CIM core operation.

Memristor-based CIM has many features that make it feasible to realize ultra-low power and energy-efficient computing:

- *Practically zero leakage computing:* 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:* CIM provides high parallelism as typically all columns in a crossbar can be accessed concurrently, leading to maximal parallelism. Moreover, the scalability of memristor 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:* 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 between the memory and processing units.

2.4. Potential CIM applications

CIM has a wide application range as it overcomes the shortcomings of conventional technologies and architectures. Some representative applications that can benefit from CIM acceleration are discussed below. However, it is worth mentioning that the applications which can benefit from CIM acceleration are not limited to these applications.

- Pattern matching with automata processor: Pattern matching with automata processor has binary Vector Matrix Multiplication (VMM) as its kernel. Such application can be easily accelerated using memristive-based CIM by mapping the binary VMM kernel to a CIM crossbar array [51,52].
- Database: Database queries require bit-wise operations such as OR, AND, and XOR. These operations can be accelerated with CIM, in this case the CIM-P is the right architecture for this application as it is less demanding from design and technology point of view [51,52].
- Compressed sensing and recovery: This requires multilevel vector-matrix multiplication, which can be easily implemented and accelerated in CIM architecture [53].
- LIght Detection And Ranging (LIDAR) image enhancement using guided image filtering: Guided image filtering is an image processing technique in which an input image is smoothed based on a guidance image, while preserving edges [54]. The guidance image serves as a filter for the convolution (VMM) operation. This can be easily accelerated with CIM.

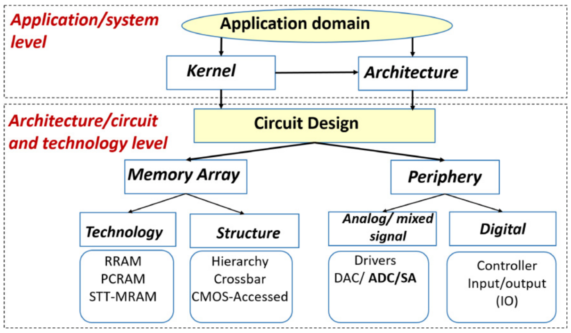


Fig. 4. Illustration of CIM design-flow.

- Image recognition and classification with deep neural networks: Image recognition and classification using deep neural networks can benefit from CIM as its kernel operation (VMM) can be mapped and accelerated using CIM crossbar array [55].
- Hyper-Dimensional Computing (HDC): HDC involves manipulation and comparison of large patterns within memory [56]. HDC is highly memory-centric application and it can be easily implemented in CIM.
- Sparse coding: Sparse coding reduces the complexity of the input signals and enables more efficient processing to improve feature extraction and pattern recognition functions [57]. Sparse coding can be mapped and accelerated by CIM-based neural network implementations.
- Associative memory: An associative memory compares input data with the data stored in the CIM array and finds the address of the data with the closest match to the input data [58]. Thus, the comparison logic operation kernel can be accelerated with CIM.
- Physically Unclonable Function (PUF): PUF can be viewed as a computational unit that returns an output response, r = f(c), where c is an input challenge and f describes the unique internal stochastic property of the PUF. A stronger PUF can be implemented using CIM crossbar array by exploiting the broad distribution of memristive resistance values [59].

3. CIM design-flow

The design of CIM accelerator is strongly application dependent; the kernels/functions that should be accelerated for one application could be different from those of other applications. Delivering an optimized and cost-effective CIM accelerator implementation requires a holistic approach in which the whole design stack in its entirety should be addressed. Fig. 4 illustrates the holistic approach for CIM accelerator design. The remainder of this section discusses CIM design-flow from system level design aspect to circuit level design of CIM architectures.

3.1. System level design

The system level design aspect of CIM design-flow involves two main steps which are crucial for developing efficient CIM architecture. These steps are discussed below.

3.1.1. Application profiling for critical kernel identification

Application profiling is the process of determining the execution speed and resource utilization of the internal functions of an application. Profiling enables the identification of the critical functions/kernels which have the most significant impact on the performance metrics (e.g., energy, latency) of an application execution. These critical kernels have to be then accelerated by CIM in order to speed-up the execution of the application and reduce the overall energy consumption. As already mentioned, different applications may require different kernels to be accelerated while minimizing the data movement. For example, the kernels of a classifier based on neural networks (NN) is Vector Matrix Multiplication (VMM) [60], the kernels of data-base application are bit-wise logic operations [52], the kernel of a matching application using an automata processor is the binary vector matrix multiplication [61], etc. Note that depending on an application, the operands needed by the kernel/function to be accelerated my reside both in memory (like it is the case of data-base query) or only one operand resides in the memory and the other one has to be fed to the CIM core through the external input (as it is the case for VMM used in NNs). Hence, the nature of the kernel contributes to the definition of the CIM (micro)architecture.

3.1.2. Accelerator configuration definition

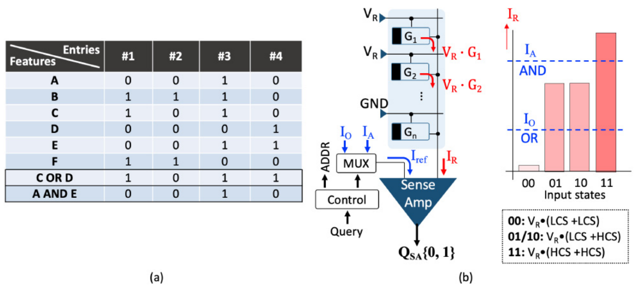
Defining appropriate CIM configuration is critical before starting the circuit design. Different configurations are possible including CIM-A versus CIM-P, on-chip (resides in the same physical chip with the main core) versus off-chip, etc. Each of these have their own pros and cons not only in terms of energy-efficiency, but also in terms of cost and complexity. Obviously, the size of the problem/ application and the kernels that need to be accelerated have a large impact on the selection of the appropriate configuration. Performing some design exploration at this stage while considering some trade-offs is quite important.

3.2. Circuit level design

Once the kernels and suitable CIM architecture are identified, the next step is designing the circuit as shown in Fig. 4. Since CIM devices are consisting of memory array and peripheral circuits (see Fig. 3), the design of both parts has to be considered. The design of the memory array has two aspects: the technology, and the structure. The technology refers to the memristive device type to be selected, which could be RRAM, PCRAM, STT-MRAM, etc. The choice of the technology depends on, but not limited to, the number of bits needed per the memory cell (e.g., PCRAM and RRAM can support multi-level storage while STT-MRAM is binary), the selected CIM architecture (e.g., CIM-A requires higher endurance than CIM-P), etc. The structure refers to the ways both the array e.g., dual bit line arrays [62], common-source line array [63], crossbar array [64] and the cell (e.g., the one-transistor-one-memristor (1T1R)) are organized. On the other hand, the peripheral CMOS circuits include two parts: analog/mixed signal and digital circuits. The analog/mixed circuits are mainly required for conversions between analog and digital domains; they might be also used for multiplexing signals in analog domain. Examples are digital-to-analog converters (DACs) and analog-to-digital converters (ADCs), customized sense amplifiers (SA), etc. The choice of analog circuits depends on, but not limited to, the kernel and the architecture; e.g., performing analog VMM with CIM-P requires at least the relatively expensive ADCs, while performing OR function with CIM-P architecture requires only customized SAs. The digital CMOS circuits are mainly required for the memory controller and temporary storage (i.e., registers).

4. CIM-based Boolean Binary Logic for database query

This section presents a database query application case study to demonstrate the importance of the CIM design-flow presented in the previous section. The section first presents a system-level profiling to identify the critical kernel of database query application for CIM acceleration. Thereafter, the circuit-level design to realize a CIM accelerator of the critical kernel is illustrated followed by simulation results.



Entries / Features	#1	#2	#3	#4
A	0	0	1	0
B	1	1	1	0
C	1	0	1	0
D	0	0	0	1
E	0	0	1	1
F	1	1	0	0
C OR D	1	0	1	1
A AND E	0	0	1	0

Fig. 5. (a) Boolean binary logic operations involved in a database query application, and (b) its implementation using scouting logic design [65].

4.1. System-level profiling

4.1.1. Binarized database query

Query operations can be performed on databases that are structured as collections of features, associated with different entries. A query is considered to be satisfied if the features associated with the selected set of entries met the predefined constraints. It is possible to formulate the queries on feature vectors as bulk *Boolean Binary Logic* (BBL) operations if the databases are represented in a bitmap representation; i.e., vectors of the logical "0" and "1" as shown in Fig. 5a. This binary representation of data favors the utilization of memristor-based CIM architecture for accelerating bulk BBL operations as the key kernel to solve database queries.

4.1.2. Kernel identification for acceleration

Many database applications involve queries with series of BBL operations. Note that any query can be expressed as the sum of products (SOP) (for example $(a_1 \cdot b_1)+(a_2 \cdot b_2)$), or the product of sums (POS) (for example $(a_1+b_1)\cdot(a_2+b_2)$) where sum and product operators correspond to logical OR and logical AND operations, respectively. However, their occurrence is instructed by the query and is not necessarily alternated OR and AND operations. Hence, an arbitrary query function $F(A)$ can be expressed as a combination of POS and SOP as given by:

$$F(A) = F(S_1) * F(S_2) * \dots * F(S_p) \tag{1}$$

where $F(S_i) = a_i * b_i$, $*$ is an OR or AND operator, and p depends on the query length.

Database applications usually involve several query functions, where each query function contains multiple logical OR and/or logical AND (BBL) operations. The amount of BBL operation per database increase exponentially with increase in a query, incurring significant energy and latency overhead. Therefore, BBL operation is a critical kernel of database applications which can be benefited from CIM acceleration.

4.1.3. Mapping of the database kernel into CIM

The underlying principle of solving database query using CIM architecture is to store the database entries in arrays of memristive devices using their conductance as the logic state variable. Therefore, we exploit the non-volatile binary storage capability of the memristive devices and the inherent parallelism that the CIM provides. For example, 0 and 1 values can be represented by the memristive devices Low Conductance State (LCS) and High Conductance State (HCS), respectively. The operands of a product or a sum (e.g., a_i and b_i) are aligned within a single column; hence, data alignment is needed. It is clear that BBL operation for database query operation, such as the example shown in Fig. 5a, can be easy mapped to CIM crossbar.

4.2. Circuit-level design

4.2.1. PCM-based 1R bitcell design

PCM is one of the most attractive resistive memory technology [66] and is being massively explored for CIM implementations [67]. Recently proposed projected PCM device [65] based on Sb_2Te_3 consists of a segment of a non-insulating projection material, in parallel to the phase-change material segment. Programming such a device requires adequate power to melt the phase-change material and sufficiently abrupt to quench the molten volume to the low conducting amorphous phase (RESET operation) or a low-amplitude slow melting to switch to the high conducting crystalline phase (SET operation). The projected layer takes advantage of the highly non-linear current–voltage (I-V) characteristics of amorphous phase change materials to decouple the read and write processes. The low sheet resistance of the projection material allows the current to bypass the highly resistive amorphous phase during the low-field read operation and flows through it only during the high-field write operation. In this manner, the read signal becomes negligibly affected by the non-ideal electronic properties of the amorphous phase, which makes the projected PCM device remarkably immune to conductance variations arising from structural relaxation, $1/f$ noise and temperature variations [65]. Each cross-point in the CIM memory unit consists of a selector-less projected PCM device (1R) with a series resistance for regulating abrupt current fluctuations. Although this 1R structure with a crossbar without selectors suffers inherently from current sneak-paths [68], special biasing schemes are used for each operation in [52] to limit this effect and make the selector-less crossbars functional.

4.2.2. CIM-based logic design

Since the database systems store data-bits in the memory unit, non-stateful-based CIM-P class of logic designs can be used to perform bulk BBL operations. Efficient CIM-P based read-assisted logic scheme such as scouting logic [38] is deployed where the logical operands are stored as conductance values in the crossbar, while the result of the logical operation is produced in the peripheral circuit (through current sensing). The operands are only read during the operation and the devices does not to be (re)programmed during the evaluation of the logical operation; hence less impact on the device endurance.

Fig. 5b shows the realization of BBL using scouting logic. By simultaneously activating multiple rows,it is possible to implement logical operations such as AND and OR using customized sense amplifiers (SAs) providing appropriate choice of reference currents during the operation execution. Hence, enabling the execution of the basic building blocks of the database query operations. Example queries are shown in Fig. 5a that determines the entries satisfying "C" OR "D" and "A" AND "E". The features are activated by biasing simultaneously the associated crossbar rows corresponding with a read voltage V_R. In a CIM core, the resulting read current I_R along each column is the summed conductance of the memristive devices according to Kirchhoff's laws. The logical output is obtained by comparing the resulting current with predefined reference currents via a SA per column. The query is performed on the entire CIM core, executing simultaneously up to C BBL operations where C is the number of column in the crossbar; thus, achieving high parallelism with O(1) time complexity.

4.2.3. Cascaded logic design for a series of BBL operations

Real-world database queries consist of a multitude of sub-queries with associated logical operations rather than a single query. Solving such a query with a scouting logic shown in Fig. 5 could yield an inefficient system as it requires an additional memory unit for temporary storage of intermediate results, additional expensive programming cycle and subsequently fetching for further processing along with the next set of logical outputs. Recently proposed cascaded logic computing system [52] performs a logical operation both in-memory and near-memory simultaneously, thus avoiding temporary storage,

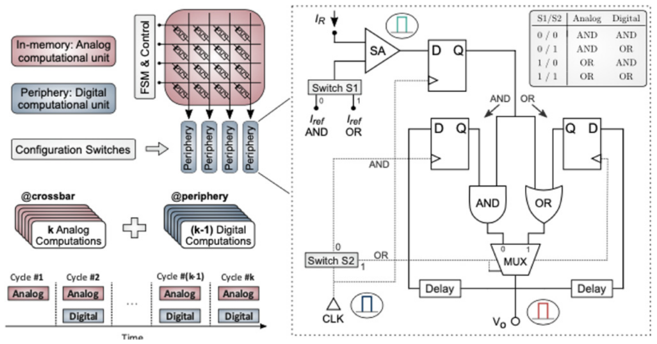


Fig. 6. Cascaded logic design to perform series of bulk BBL operations for high throughput [52].

Table 2
Design parameters.

Parameters	Specifications
PCM device	Sb_2Te_3 [65]
R_{off}	1 MΩ
R_{on}	20 kΩ
Read voltage	0.1 V with ±10% variations
CMOS technology	65 nm TSMC
Database size	41 × 303
Simulation environment	Synopsis HSPICE

programming operation and subsequent fetching. While an in-memory analog computation using scouting logic is executed, a near-memory digital logic operation is carried out at the periphery of the memory array using conventional CMOS-based gates (see Fig. 6). Since these analog and digital computing steps are performed in parallel, a 2X throughput gain is achieved.

4.3. Results

4.3.1. Simulation setup

Simulations are performed using Sb_2Te_3 based Phase-Change Memory (PCM) device [65] that is assembled in a 1R crossbar structure. The two stable high and low resistive states chosen for our simulations are 1 MΩ and 20 kΩ, with R_{off}/R_{on} of 50. Table 2 presents the simulation setup used to evaluate the performance efficiency of CIM implementation of database query.

4.3.2. Simulation results

In order to demonstrate the efficiency of the BBL operations mapped to CIM crossbar, an example query comprising of 11 OR and AND operations is chosen, out of which 6 logical operations are performed in the analog domain (using scouting logic), while the remaining 5 logical operations are performed in the digital domain (using cascading logic). The simulation results are presented using a database of size 41 × 303, where a maximum of 2 operations are performed at each clock period (6 ns) which means that the total time required for a single query with 11 logical operations is 36 ns. The total power consumption of the system is 558 μW and the total required energy for the fully cascaded query is 20 pJ (3.3 pJ/cycle). These numbers refer explicitly to the core components, which means that the control unit and any post-processing circuits are excluded from the calculation. The achieved throughput was 92.6 GOPS and the energy-efficiency reached 166 TOPS/W. In other words, each logic operation consumes less than 10fJ of energy.

5. CIM-based Binary Neural Network (BNN) for image classification

This section presents an image classification using CIM implemented Binary Neural Network (BNN) case study to demonstrate the applicability of CIM design-flow. The section first presents the system-level design

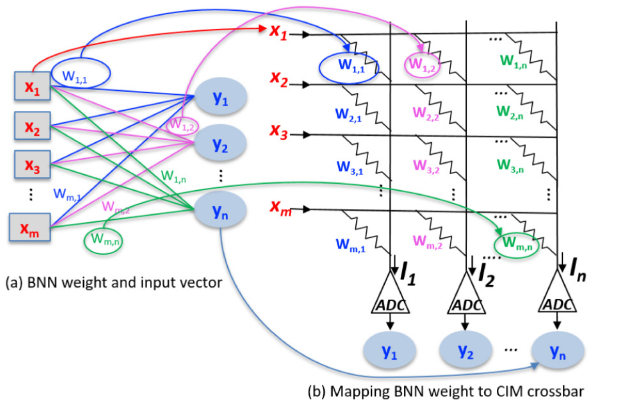


Fig. 7. Mapping of BNN weights to crossbar array illustrating weight mapping and MAC operation for each neuron.

aspect by presenting the system-level profiling for kernel identification. Then, the section discusses the circuit design aspect to realize the CIM acceleration of the critical kernel. Finally, the simulation results are presented.

5.1. System-level profiling

Binary Neural Network (BNN) is a simplified version of Artificial Neural Networks (ANN) which reduces memory and computation cost of ANN, and enables to deploy deep models of ANN on resource-constrained platforms such as edge-platforms [69]. The concept behind BNN is simply to represent each weight value and input using binary values +1 and −1 so that the storage and computation can be performed in 1-bit instead of full precision [69,70]. Therefore, BNNs can perform various tasks such as image recognition and classification in a resource-efficient manner than their ANN counterparts. Due to their binary storage and computation, BNNs can be easily realized with CIM.

5.1.1. BNN working principle

BNNs operate similar to the full-precision ANNs, where the neurons in every layer calculate the weighted sum of the binary input vector and binary weights, and then pass it through an activation function to get the output as shown in Eq. (2).

$$O_i = f(\sum W_{i,j} \cdot X_i) \tag{2}$$

where O_i is the output, f is the activation function (*Sigmoid* function in this case), $\sum W_{ij} \cdot X_i$ is the weighted sum of the binary input (X_i) and binary weight ($W_{i,j}$).

Unlike ANNs, BNNs need to binarize the floating point weights of the network. Therefore, the floating-point weight values are converted to binary values using the signum (Sign) function as shown in Eq. (3).

$$W_b = Sign(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{3}$$

where W_b is the binary weight, Sign is the signum function and x is the floating-point weight.

5.1.2. BNN critical kernel identification

To accelerate BNNs on crossbar array the core operation needs to be identified. In this case the weighted sum, Multiply Accumulate (MAC), operation given in Eq. (2) is the costly operation. Thus, the MAC operation can be easily identified as the kernel operation of BNNs for two main reasons: (i) the size of the MAC (Vector matrix operation (VMM)) per neuron increases linearly with the increase in the number of inputs. (ii) every neuron in the network have to perform the operation and increasing the number of neurons in a BNN increases

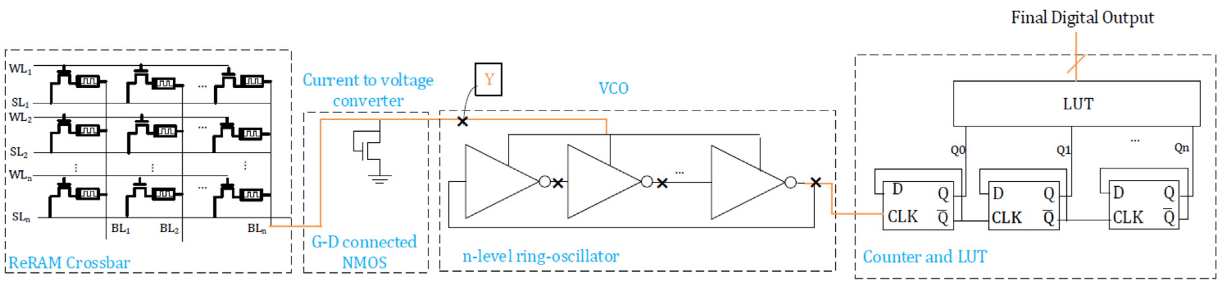


Fig. 8. Schematic of ReRAM array and output current sensing circuit based on a VCO-based ADC [71].

the required MAC operations significantly. For example, a simple two layer fully-connected BNN with 10 neurons in each layer and 10 input pixels needs 20 MAC units performing in total 380 operations (200 multiplications and 180 additions). Therefore, the MAC operation is indeed the critical kernel which should be be accelerated using CIM.

5.1.3. Mapping BNN kernel to CIM crossbar

Mapping the BNN kernel operation, MAC/VMM, into a CIM crossbar array is straightforward. The binary weights of the BNN can be encoded as resistance states of the memristors where HRS and LRS represent −1 and +1 weights, respectively. Similarly, the binary inputs of the neurons can be represented by binary voltages. For a 1T1R (1 transistor and 1 resistor) based crossbar, the binary voltages can be used as an input directly without the need to have Digital to Analog Converter (DAC), as the transistor can be easily turned on/off according to the input voltage. Fig. 7 shows the mapping of a single BNN layer into a 1T1R based CIM crossbar. The binary weights of the layer shown in Fig. 7(a) are mapped as resistance states of the memristors in the crossbar. As shown in Fig. 7(b), each column in the CIM crossbar corresponds to a single neuron and the number of columns is equivalent to the number of neurons, while the number of rows of the crossbar is equal to the number of inputs to the layer.

5.2. Circuit-level design

It is worth mentioning that Analog to Digital Converter (ADC) is not always necessary to map BNNs into a CIM crossbar. For a BNN where the neurons are implemented as a *threshold logic*, ADC is not required as the neuron output can be easily determined by comparing the column output current with a reference value. Since the BNN given in Fig. 7 is based on *Sigmoid* function neurons, ADC is required to convert the output currents of the crossbar columns into digital values. Therefore, efficient 1T1R bit-cell and ADC circuit designs are needed to implement the CIM crossbar structure presented in Fig. 7(b).

5.2.1. 1T1R bit-cell design

A RRAM device is fabricated by sandwiching a metallic oxide (commonly HfO_x, or TiO_x) between the two regions, i.e., the doped top electrode and undoped bottom electrode. RRAM operation is based on the reversible formation or disruption of the Conductive Filament (CF) in a resistive layer leading to a high or low resistance state. When a sufficiently high positive voltage (higher than the set threshold voltage, V_{set}) is applied, a CF is formed and the device will have a low resistance state. On the contrary, when a negative voltage ($<V_{reset}$) is applied, the CF breaks down which leads to high resistance state. In the 1T1R structure the transistor (1T) is mainly required to program and verify read of the individual conductance values in each cell. Moreover, it solves the *sneak paths* problem which is common in a 1R bit-cell based crossbar [72]. Sneak path leads to deviation of results, especially, in the analog computation. Besides solving the sneak path problem, 1T1R structures enable analog programming for the memristors with low standard deviation [73], as it enables accurately measurement of the programmed values.

Table 3
Measurement and simulation setup.

Parameters	Specifications
RRAM device	ZrO_2/Ta [74]
R_{HRS}	30 kΩ
R_{on}	3 kΩ
Read voltage	0.9 V with ±10% variations
CMOS technology	28 nm TSMC
Workload application	Binary Neural Network (BNN)
Behavioral simulation	Python
Circuit level simulation	SPICE

5.2.2. ADC design

ADCs are used whenever an analog signal is needed as an input for digital modules. In this work a Voltage Controlled Oscillation (VCO) based ADC [71] (see Fig. 8) is adopted to convert the analog MAC result of the crossbar into digital values, which will be an input to the neurons. The VCO-based ADC presented in Fig. 8 has three stages. The first stage transforms the analog bit-line current into an analog voltage. The analog voltage is then transformed into pulses with the help of the VCO at the second stage. Finally, the third stage counts the generated pulses with a counter and mapped it to the corresponding digital signal with the help of Lookup Table (LUT). The output of the third stage is equivalent to a digital signal and it can be processed by digital modules.

The VCO-based ADC is chosen as it have various advantages over the state-of-the-art ADCs. These advantages include: (i) its compact design consumes less area and allows to assign one ADC per column for higher resolution. (ii) satisfies the RRAM devices requirements such as variability in their resistive states and the restriction on the maximum voltage across the device.

5.3. Measurement and simulation results

5.3.1. Measurement results

Before mapping the BNN to a RRAM crossbar array, we investigated the factors that determine the output current with an experimental study for a single column with 8 rows of RRAM devices fabricated based on the setup given in Table 3. In the experiment, the different (mean) output current levels are determined by the values of the input vector and the conductance values of the 8 RRAM devices in the column. Fig. 9, shows the current measurement results of the column where 4 devices are programmed to LRS, while the other 4 devices are programmed to HRS. The measured output current is plotted for all 256 possible input vector combinations ranging from (0 0 0 0 0 0 0 0) to (1 1 1 1 1 1 1 1). They are grouped in 5 different "groups"; each group corresponds to the same number of selected LRS and HRS devices; for example, the group indicated with "1 LRS device" presents the 32 measurements of the 32 (from 256) possible inputs for which the number of selected cells with LRS is just 1. The variability in the observed currents is due to the number of selected HRS devices per input combination (indicated as 1 in Fig. 9), and to the selection of one LRS from the four LRS devices (indicated as 2 in Fig. 9). Most LRS resistances are found to be lower than the target value, resulting in an increasing vertical deviation from the expected current (indicated as 3 in Fig. 9).

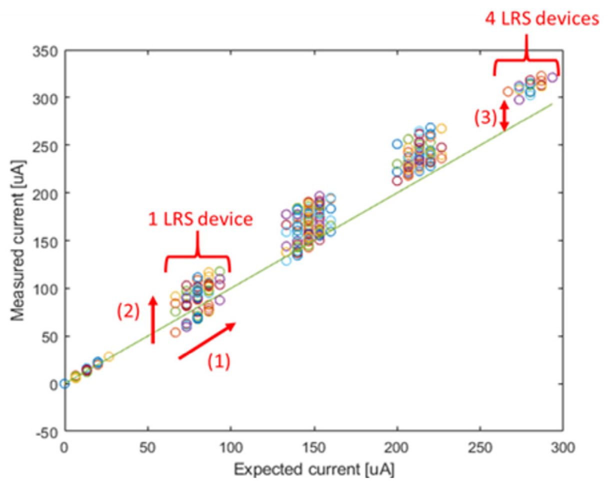


Fig. 9. VMM using RRAM crossbar array: Current measurements for a single column in a 8 × 8 array.

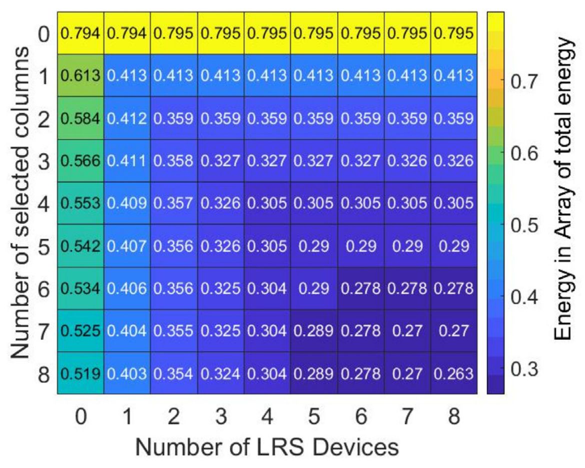


Fig. 10. Energy consumed in an 8 × 8 array during MAC operation, as function of number of selected columns and LRS devices.

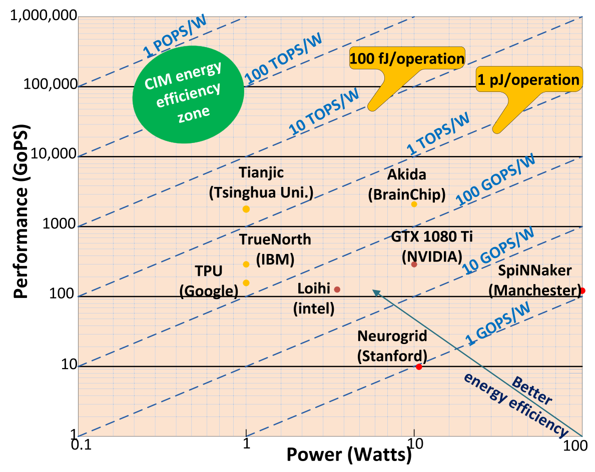


Fig. 11. Illustration of energy efficiency for various hardware and CIM energy-efficiency potential [75].

5.3.2. Simulation results

The performance metrics such as energy and latency of the MAC operation in RRAM crossbar are investigated by mapping BNN wights to 8 by 8 crossbar array simulated based on the setup given in Table 3. Besides the memristor array, the current sensing ADC plays critical role for the performance metrics. The investigation uses the VCO-based ADC (shown in Fig. 8) to convert the MAC operation result into a digital value [71]. The energy consumption of a MAC operation shown in Fig. 10 is extracted using the setup circuitry shown in Fig. 8. The energy results shown in Fig. 10 confirm the major role of the ADC output circuit, as the energy dissipation in the array may be as low as 30%.

6. Discussion and future directions

Edge devices need energy-efficient, compact and reliable hardware for the execution of NN applications. Fig. 11 shows the energy efficiency with respect to their performance for various hardware architectures and energy efficiency potential of CIM. NNs require a lot of multiple and accumulate (MAC) operations, which is resource and energy intensive. In recent years, specialized hardware (like Tensor processor units) have been developed to accelerate AI computations which can do more calculations per second than GPUs, while consuming the same amount of power, as shown in the figure. However, these specialized hardware accelerators are costly to be deployed in resource-constrained edge platforms as they are not energy-efficient.

Clearly, the demand of energy-efficiency cannot be full-filled by the existing hardware for the edge applications. CIM-based computation architectures using memristive devices have potential to improve the performance, energy and area efficiency compared to these traditional computing systems. This is due to the fact that these memristive devices have several advantageous features such as non-volatility, scalability, high density, CMOS compatibility, etc. Additionally, the nature of memristor state dynamics, make CIM more suitable for edge applications by addressing the architectural and technological walls of the conventional architectures.

Despite many advantages of memristor-based CIM architectures, they face several challenges at various abstraction levels, starting from materials and device-level to all the way to the architecture and compiler-level.

- Material/Technology: These devices use new materials and are subjected to new technology related issues. For instance, memristor devices can have several non-ideal characteristics such as variations, defects, limited low to high resistance ratios, stochastic write behavior etc. These non-ideality issues can influence the overall accuracy of the architecture [49,76–80]. Additionally, memristors have limited endurance and require high programming current to program them.
- Circuit/Architecture: In addition to the device level non-idealities, CIM architectures also face circuit-level non-idealities. For instance, non-ideal Digital-to-Analog Converters (DAC) and Analog-to-Digital Converters (ADC), which are the two main components of CIM architecture, can significantly impact the accuracy [81,82]. Moreover, inaccuracies due to driver resistances, wire resistances and other parasites can contribute to accuracy reduction.
- Tools/Compiler: Previous challenges can add complications for profiling and tool development activities, that can impact the kernel identification as well as CIM accelerations tasks. This will influence the design space exploration, that is essential to guide optimal design options as well as exploration of various trade-off scenarios.

Therefore, addressing the aforementioned challenges plays an instrumental role in harnessing the full potential of CIM for realizing AI application on resource-constrained edge platforms. In spite of these challenges, CIM implementations are demonstrating their promising potential towards achieving targeted energy-efficiency for edge computing (fJ/op). Nevertheless, more research is needed to address the aforementioned challenges and unlock the full potential of CIM.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] M. Satyanarayanan, The emergence of edge computing, Computer (2017).
[2] W. Shi, S. Dustdar, The promise of edge computing, Computer (2016).
[3] J. Chabas, et al., New Demand, New Markets: What Edge Computing Means for Hardware Companies, Tech. Rep, McKinsey & Company, New York, NY, USA, 2018.
[4] Edge Computing Market Share & Trends Report, 2021–2028, 2020, https://www.grandviewresearch.com/industry-analysis/edge-computing-market, accessed: 2020-11-29.
[5] D.A. Patterson, Future of computer architecture, in: Berkeley EECS Annual Research Symposium (BEARS), College of Engineering, UC Berkeley, US, 2006.
[6] S. Hamdioui, et al., Memristor for computing: Myth or reality? in: DATE, 2017.
[7] H. Amrouch, et al., Towards reliable in-memory computing: From emerging devices to post-von-Neumann architectures, in: VLSI-SoC, 2021.
[8] S. Diware, et al., Accurate and energy-efficient bit-slicing for RRAM-based neural networks, TETCI (2022).
[9] A. Singh, et al., Cim-based robust logic accelerator using 28 nm stt-mram characterization chip tape-out, in: AICAS, 2022.
[10] A. Lines, et al., Loihi asynchronous neuromorphic research chip, in: IEEE ASYNC, 2018.
[11] M. Davies, Taking neuromorphic computing to the next level with loihi 2, 2021.
[12] E. Painkras, et al., SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation, IEEE J. Solid-State Circuits (2013).
[13] M.V. DeBole, et al., TrueNorth: Accelerating from zero to 64 million neurons in 10 years, Computer (2019).
[14] K. Rocki, et al., Fast stencil-code computation on a wafer-scale processor, in: SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, 2020.
[15] I. Kataeva, et al., Towards the development of analog neuromorphic chip prototype with 2.4 M integrated memristors, in: ISCAS, 2019.
[16] F. Cai, et al., A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations, Nat. Electron. (2019).
[17] W.-H. Chen, et al., CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors, Nat. Electron. (2019).
[18] C.-X. Xue, et al., A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices, Nat. Electron. (2021).
[19] C. Frenkel, et al., MorphIC: A 65-nm 738k-synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning, Trans. Biomed. Circuits Syst. (2019).
[20] J. Schemmel, et al., Live demonstration: A scaled-down version of the brainscales wafer-scale neuromorphic system, in: ISCAS, 2012.
[21] A. Yousefzadeh, et al., Energy-efficient in-memory address calculation, Trans. Archit. Code Optim. (TACO) (2022).
[22] Hsu, et al., AI edge devices using computing-in-memory and processing-in-sensor: From system to device, in: IEDM, 2019.
[23] Z. Zhou, et al., Edge intelligence: Paving the last mile of artificial intelligence with edge computing, Proc. IEEE (2019).
[24] S. Hamdioui, et al., Applications of computation-in-memory architectures based on memristive devices, in: DATE, 2019.
[25] A. Gebregiorgis, et al., A survey on memory-centric computer architectures, JETC (2022).
[26] S. Rai, et al., Perspectives on emerging computation-in-memory paradigms, in: DATE, 2021.
[27] A. Shafiee, et al., ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars, ACM SIGARCH Comput. Archit. News (2016).
[28] L. Song, et al., Pipelayer: A pipelined reram-based accelerator for deep learning, in: International Symposium on High Performance Computer Architecture, HPCA, 2017.
[29] X. Qiao, et al., Atomlayer: a universal reram-based cnn accelerator with atomic layer computation, in: DAC, 2018.
[30] S. Gupta, et al., Nnpim: A processing in-memory architecture for neural network acceleration, IEEE Trans. Comput. (2019).
[31] F. Chen, et al., Regan: A pipelined reram-based accelerator for generative adversarial networks, in: ASP-DAC, 2018.
[32] H.A.D. Nguyen, et al., A computation-in-memory accelerator based on resistive devices, in: Proceedings of the International Symposium on Memory Systems, 2019.
[33] J. Borghetti, et al., 'Memristive'switches enable 'stateful'logic operations via material implication, Nature (2010).
[34] S. Kvatinsky, et al., Memristor-based IMPLY logic design procedure, in: ICCD, 2011.
[35] Kvatinsky, et al., MAGIC—Memristor-aided logic, TCAS II (2014).
[36] M. Hu, et al., Hardware realization of BSB recall function using memristor crossbar arrays, in: DAC, 2012.
[37] S. Li, et al., Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories, in: DAC, 2016.
[38] L. Xie, et al., Scouting logic: A novel memristor-based logic design for resistive computing, in: ISVLSI, 2017.
[39] A. Singh, et al., Low-power memristor-based computing for edge-AI applications, in: ISCAS, 2021.
[40] G.S. Sandhu, Emerging memories technology landscape, in: Non-Volatile Memory Technology Symposium (NVMTS), 2013 13th, 2013.
[41] M. Radosavljević, et al., Nonvolatile molecular memory elements based on ambipolar nanotube field effect transistors, Nano Lett. (2002).
[42] A. Gebregiorgis, et al., Spintronic normally-off heterogeneous system-on-chip design, in: DATE, 2018.
[43] M. Le Gallo, A. Sebastian, An overview of phase-change memory device physics, J. Phys. D: Appl. Phys. (2020).
[44] M. Ali, et al., IMAC: In-memory multi-bit multiplication and accumulation in 6T sram array, IEEE TCAS I (2020).
[45] S. Li, et al., Drisa: A dram-based reconfigurable in-situ accelerator, in: 50th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO, 2017.
[46] S.K. Gonugondla, et al., Energy-efficient deep in-memory architecture for NAND flash memories, in: ISCAS, 2018.
[47] S. Salahuddin, et al., The era of hyper-scaling in electronics, Nat. Electron. (2018).
[48] F. Oboril, et al., Evaluation of hybrid memory technologies using SOT-MRAM for on-chip cache hierarchy, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2015).
[49] R. Bishnoi, et al., Special session–emerging memristor based memory and CIM architecture: Test, repair and yield analysis, in: 2020 IEEE 38th VLSI Test Symposium, VTS, 2020.
[50] E.C. Apollos, et al., Memristor-based CiM architecture for big data era, in: International Conference on Electronics, Computer and Computation, ICECCO, 2019.
[51] M. Imani, et al., NVQuery: Efficient query processing in nonvolatile memory, IEEE TCAD (2018).
[52] I. Giannopoulos, et al., In-memory database query, Adv. Intell. Syst. (2020).
[53] A. Sebastian, et al., Temporal correlation detection using computational phase-change memory, Nature Commun. (2017).
[54] K. He, et al., Guided image filtering, IEEE Trans. Pattern Anal. Mach. Intell. (2012).
[55] M. Komar, et al., Deep neural network for image recognition based on the Caffe framework, in: International Conference on Data Stream Mining & Processing, DSMP, 2018.
[56] P. Kanerva, Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors, Cogn. Comput. (2009).
[57] P.M. Sheridan, et al., Sparse coding with memristor networks, Nature Nanotechnol. (2017).
[58] R. Karam, et al., Emerging trends in design and applications of memory-based computing and content-addressable memories, Proc. IEEE (2015).
[59] P. Koeberl, et al., Memristor PUFs: a new generation of memory-based physically unclonable functions, in: DATE, 2013.
[60] A. Haron, et al., Parallel matrix multiplication on memristor-based computation-in-memory architecture, in: International Conference on High Performance Computing & Simulation, HPCS, 2016.
[61] J. Yu, o. Du Nguyen, Memristive devices for computation-in-memory, in: DATE, 2018.
[62] X. Dong, et al., Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2012).
[63] Y. Sato, et al., Sub-100-μA reset current of nickel oxide resistive memory through control of filamentary conductance by current limit of MOSFET, IEEE Trans. Electron Devices (2008).
[64] L. Zhao, et al., Constructing fast and energy efficient 1tnr based reram crossbar memory, in: 2017 18th International Symposium on Quality Electronic Design (ISQED), 2017.
[65] I. Giannopoulos, et al., 8-bit precision in-memory multiplication with projected phase-change memory, in: IEDM, 2018.
[66] G. Burr, et al., Recent progress in phase-change memory technology, IEEE J. Emerg. Sel. Top. Circuits Syst. (2016).
[67] A. Sebastian, et al., Computational phase-change memory: Beyond von Neumann computing, J. Phys. D: Appl. Phys. (2019).
[68] H.D. Lee, et al., Integration of 4F2 selector-less crossbar array 2Mb ReRAM based on transition metal oxides for high density memory applications, in: 2012 Symposium on VLSI Technology, VLSIT, IEEE, 2012.
[69] H. Qin, et al., Binary neural networks: A survey, Pattern Recognit. (2020).
[70] X. Sun, et al., XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks, in: DATE, 2018.

[71] M. Mayahinia, et al., A voltage controlled oscillation based ADC design for computation-in-memory architectures using emerging ReRAMs, JETC (2021).
[72] H. Zheng, et al., Reducing forming voltage by applying bipolar incremental step pulse programming in a 1T1R structure resistance random access memory, IEEE Electron Device Lett. (2018).
[73] X. Sheng, et al., Low-conductance and multilevel CMOS-integrated nanoscale oxide memristors, Adv. Electron. Mater. (2019).
[74] A. Hardtdegen, et al., Improved switching stability and the effect of an internal series resistor in HfO 2/TiO x Bilayer ReRAM cells, IEEE Trans. Electron Devices (2018).
[75] A. Gebregiorgis, et al., Dealing with non-idealities in memristor based computation-in-memory designs, in: VLSI-SoC, 2022.
[76] I. Chakraborty, et al., Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks, in: DAC, 2020.
[77] S. Jain, et al., RxNN: A framework for evaluating deep neural networks on resistive crossbars, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2020).
[78] S. Diware, et al., Unbalanced bit-slicing scheme for accurate memristor-based neural network architecture, in: AICAS, 2021.
[79] M. Fieback, et al., Testing scouting logic-based computation-in-memory architectures, in: IEEE European Test Symposium, ETS, 2020.
[80] T. Ketkar, S. Sahay, Impact of non-idealities in RRAMs on hardware spiking neural networks, in: IEEE Electron Devices Technology & Manufacturing Conference, EDTM, 2021.
[81] A. Singh, et al., SRIF: Scalable and reliable integrate and fire circuit adc for memristor-based cim architectures, TCAS I (2021).
[82] C. Münch, et al., A novel oscillation-based reconfigurable in-memory computing scheme with error correction, IEEE Trans. Magn. (2020).