A full spectrum of computing-in-memory technologies

Zhong Sun[1*], Shahar Kvatinsky[2], Xin Si[3], Adnan Mehonic[4], Yimao Cai[1] and Ru Huang[1]

[1]Institute for Artificial Intelligence, School of Integrated Circuits, Peking University, Beijing Advanced Innovation Center for Integrated Circuits, Beijing 100871, China

[2]The Andrew and Erna Viterbi Faculty of Electrical and Computer Engineering, Technion—Israel Institute of Technology, Haifa, Israel

[3]Department of Electronic Engineering, Southeast University, Nanjing 210000, China

[4]Department of Electronic & Electrical Engineering, UCL, Torrington Place, London, WC1E 7JE UK

*e-mail: zhong.sun@pku.edu.cn

To overcome the dreaded von Neumann bottleneck and to provide sustainable improvement of throughput and energy efficiency, computing-in-memory (CIM) has been extensively investigated across the full computing stack. Underlying the proliferation of various CIM schemes is to implement two kinds of computing primitives: logic gate or multiply-accumulate (MAC) operation. By observing the input and output in either operation, CIM technologies differ regarding how memory cells participate in the computation process. This divergence has led to conceptual complexity and vagueness that prevent a clear overview of the prevalent CIM schemes, each under intensive study in different stack levels such as semiconductor device, circuit design, architecture and system. Here, by identifying the degree of memory cells fused in the computation as inputs and/or output, we propose a full-spectrum classification of all CIM technologies, which is agnostic to the memory devices that could be mature or emerging, volatile or non-volatile, capacitive or resistive. Detailed principles are elucidated for standard CIM technologies across the spectrum. It provides a platform for comparing the advantages and disadvantages, evaluating the challenges, and conducting benchmarking of various CIM technologies. Additionally, such a taxonomy should inspire more CIM schemes by applying the spectrum to different memory devices and computing primitives.

Introduction

Modern computers have been very successful, thanks to their fundamentals including the universal Boolean logic gates, the continuous down-scaling of transistors, and the classic architecture that separates the processing and storage of data, thus allowing for the dedicated upgrades of each part. During the past decades, driven by Moore's law, the performance of processors has been dramatically improved. As the volume of data rapidly increases, as well as the adoption of data-centric computing (*e.g.* machine learning), the transfer of data between the two physically-separated units becomes highly costly, which dominates the overall latency and energy consumption[1]. On the other hand, despite the universality and robustness, computation with traditional logic gates is considered inefficient, consuming many resources for arithmetic calculations, such as multiplication, addition, and non-linear functions[2]. It becomes even resource-intensive to increase the computational parallelism by building many cores at the underlying hardware level.

To solve the communication bottleneck issue, in recent years, computing-in-memory (CIM) technologies have been actively investigated[3-5]. CIM is closely related to other concepts that include in-memory computing and processing-in-memory[6], and a sub-field is sometimes termed logic-in-memory[7]. The basic idea of CIM is to move data computations to the memory unit where they are stored, thus realizing *in situ* computing and eliminating the bandwidth limitation, and the data movement cost. It usually exploits physical laws, such as Kirchhoff's current law (KCL) and charge sharing in the memory array for analog computation, demonstrating efficient computing primitives including logic gate and MAC. Furthermore, the crosspoint random-access memory (RAM) architecture allows natural fan-outs that facilitate massive computational parallelism. These advantages of CIM have opened multiple novel research directions to pursue computer performance improvement in the post-Moore era and to build computing accelerators for prevailing applications such as artificial intelligence[8] (AI).

Research on CIM occurs at various levels along different dimensions, ranging from fundamental electronic devices to high-level architectures and large-scale systems, from mature silicon-based memories to emerging resistive memories. Despite bearing the same name, the

underlying principles of CIM technologies vary significantly in essence, depending on (1) whether all or part of the input operands are provided *in situ* by the memory cells, (2) whether the computation is finalized with the re-storage of the output *in situ* in the memory cell, (3) whether the input/output data are volatile or non-volatile, and (4) whether the input/output data are represented in the same physical manner. Such disparities hinder an inclusive but comprehensive insight into CIM technologies. In this Review, we present a full-spectrum classification of the prevalent CIM technologies. By abstracting the computing primitives as a two-input (X and Y), one-output (Z) operation, and according to the degree of memory cells participating in the computation, a full spectrum of CIM technologies is established. Memory cells may provide the inputs, re-store the output, and even perform the non-linear activation. Consequently, the spectrum ranges from the XYZ-type where all operands reside in the memory cells, to the O-type, where no memory cell is involved in the computation. This Review provides a unified view for assessing all CIM technologies as a continuum to analyze the advantages and disadvantages of each, thus supporting the capability exploration and development of CIM without ambiguity.

Computing primitives

MAC is the atomic operation of computer arithmetic, and it is related to the foundational Boolean logic gate in a way explained in Fig. 1, where we propose a diagram illustrating their relationship through the concept of artificial neural network (ANN) based on vector/matrix arithmetic, in the sense of both traditional computing and CIM. In von Neumann computer, all operations rely on the functionally complete set of logic gates, which in turn are built with complementary metal-oxide-semiconductor (CMOS) transistors. The logic gates are then used to build processing cores for arithmetic computations, among which the most important is the MAC operation. The scalar MAC operation is then extended to carry out vector/matrix arithmetic by sequential processing or parallelization with multiple cores, thanks to the regular form of matrix algebra. Finally, matrix lays the cornerstones for a plethora of algorithms, among which ANNs (and deep learning) are the highly concerned ones that draw the most attention nowadays. Conversely, in the case of CIM, the idea starts from the MAC operation with the embedded circuit physics, which lays the basis for logic gates, through the concept of ANN-

based threshold logic, which, in turn, is achieved through vector operations. Either parallel MAC or logic gate may be performed along one column in the memory array (Fig. 1b), although the former constitutes the basis for the latter in this context.

Fig. 1b illustrates a typical CIM architecture, which contains multiple memory banks, each of which, in turn, is composed of several memory array tiles (MATs). The memory array is organized as the random-access architecture, the workplace of most memory technologies across the traditional memory hierarchy, and almost all emerging memory technologies[9]. It is constructed by intersecting horizontal wordlines (WLs) and vertical bitlines (BLs). A memory cell is placed at each crosspoint position for data storage, which could be a single-bit or multi-bit value. In Fig. 1c, the common memory devices considered for CIM are illustrated, and categorized as volatile memory (VM) and non-volatile memory (NVM). Dynamic RAM (DRAM) and static RAM (SRAM) are mainstream memory products in modern computers[10,11]. They use charges in the constitutive capacitor or the parasitic gate capacitor for information storage and thus are volatile. Most emerging NVM memory concepts can be recognized as resistive memory, as they use the resistance attribute of the device for data storage[12], including resistive RAM (RRAM), phase change memory (PCM), magnetoresistive RAM (MRAM), and ferroelectric tunnel junction (FTJ). They are two-terminal (passive 1R) devices and can be naturally placed in the so-called crossbar array architecture. However, for reliable device operations in the integrated array, such as precise writing and mitigating sneak current path, a common solution is to connect a transistor in series to each device, forming the 1T1R cell structure. Flash memory is conventionally regarded as a charge-based device. However, in CIM, Flash works as a resistive memory whose resistance is determined by the stored charge (thus the threshold voltage) and the externally applied gate voltage, thus contributing to the physical MAC and vector/matrix arithmetic with the NOR type[13]. The situation is the same for the ferroelectric field-effect transistor (FeFET)[14], where the polarized charges in the ferroelectric layer affect the channel conductivity, thus determining the device resistance.

While allowing for random access to any memory cell in the array, this architecture is also beneficial for CIM, which is made possible by simultaneously activating multiple WLs and

BLs. For this purpose, besides the conventional WL/BL peripheral circuits for memory operations, additional modules are required for CIM, typically including instructions for simultaneous activation of multiple WLs, WL drivers for inputting analog voltages, and BL sensing circuits for the readout and conversion of analog outputs. The preliminary of CIM is based on parallel MAC associated with specific physical laws in the array. Upon activating multiple WLs, the current or potential on one BL is the dot product of vectors resulting from the interaction between the WL voltages and the memory cells, which implements parallel MAC operations. It is easily generalized to multiple columns to perform matrix operations, *e.g.*, matrix-vector multiplication (MVM)[15]. The implementation of parallel MAC provides a workhorse for accelerating some important algorithms, where neural networks are dominant[16-23], and other problems include Ising machine[24,25] *etc*. The same principle may also be used for content-addressable memory applications[26-30]. On the other hand, the parallel MAC result on the BL could be an intermediate result, which will be combined with a non-linear activation function to perform a logic gate, namely threshold logic. The non-linear function can also be met by hardware components, such as a CMOS inverter[31], a latch circuit, and an abrupt resistive switch[32]. The implementation of logic gates in CIM is about to develop a functionally complete logic set, based on which computational blocks, *e.g.*, adder and multiplier, are built to support general-purpose arithmetic operations[33-38]. Compared to the CMOS logic gates, the benefits of CIM counterparts include the capability of fusing computation in the memory array, and the massive computing parallelism offered by the crosspoint RAM architecture. Since CIM logic gates rely on analog computing with physical laws, any linearly separable logic functions may be carried out in one operation. As a result, complicated logic functions, *e.g.*, 1-bit full adder, can be conveniently achieved with reduced number of operations and hardware cost.

A full spectrum of CIM technologies

The two computing primitives can be abstracted as an equation $Z=X \bullet Y$, where the dot symbolizes the algebraic function. For dot product, X and Y represent the stored weight vector and the input vector, respectively, and Z is the scalar output. For logic gates, X and Y are two input operands, and Z is the logic output. Based on this assumption, a spectrum of CIM technologies is proposed. According to whether X and Y are provided by memory cells, and

whether the output *Z* is re-stored in a memory cell at the end of the computation, CIM is identified as six categories. Fig. 2 shows such a full-spectrum classification, together with the typical memory technologies that have been reported for each CIM type, the solved computing primitives, and the targeted applications.

(1) XYZ-CIM: both *X* and *Y* are provided by the memory cells in the array and the output *Z* is also re-stored in a memory cell. The computation relies on the implicit readout of *X* and *Y*, thus modifying the BL potential, which eventually rewrites the output cell. XYZ-CIM is typical for Boolean logic operations, which have been implemented with single-bit non-volatile RRAM[32,39-41], PCM[42], MRAM[43,44], and volatile DRAM[45-47].

(2) XZ-CIM: only one input operand is residing in a memory cell during computation. The other input is encoded by the externally applied voltage, and the output *Z* is re-stored as a single-bit cell state at the end. XZ-CIM only applies to NVM-based logic operations, and typical memory technologies include RRAM[48] and MRAM[49].

(3) Z-CIM: only the output *Z* is stored in the memory cell, and the inputs are provided through the BL and WL. By considering all possible combinations of BL and WL voltages, the resulting single-bit cell states constitute a logic gate. Z-CIM has been implemented with NVMs, such as RRAM[50-52], MRAM[53], and PCM[54].

(4) XY-CIM: both input operands *X* and *Y* are provided by memory cells, while the output *Z* is obtained at the BL sense amplifier (SA). It applies to logic operation as well, and the memory technologies could be resistive NVMs[55,56] or SRAM[57,58]. It works on based on parallel readout of two single-bit memory cells and the result is sensed and discretized to a binary output.

(5) X-CIM: only input *X* is provided by memory cells along one column in the array, *Y* is represented by the external voltages applied to WLs, and the output *Z* is obtained at the BL periphery. Different from the above-mentioned types of CIM, X-CIM usually aims to perform the dot product of two vectors in a highly parallel manner. It has been implemented

with all memory technologies[16-23], including single-bit or multi-bit NVMs, and single-bit VMs, which forms positive feedback to backward flourish its research.

(6) O-CIM: there is no interaction of memory cells here, rather conventional logic gates or computational blocks are located close to memory cells or arrays for carrying out computations. O-CIM is usually designed with mature memory technologies[59-61], including SRAM and DRAM. It resembles the earlier concept of computing-near-memory but advances to further cut down the memory-processor distance.

The spectrum based on this taxonomy should cover all CIM technologies, thanks to the clarity of a comprehensive identification of the sources of inputs and the orientation of output. Beyond the end of the spectrum, it contacts the conventional von Neumann paradigm. Across the range, a given memory technology may have been used in multiple types of CIM, but with different principles. On the other hand, some kinds of CIM may only be technically possible or worthy of interest with specific memory devices. Additionally, the computing primitives are related to the CIM types and eventually the memory vehicles. Except for the O-CIM, all other CIM types rely on analog multiplication, addition, and non-linear activation with physical laws in the circuit. Combining the former two operations results in the dot product for parallel MAC operations, and combining all three dictates Boolean logic gates.

Principles of CIM technologies across the spectrum

XYZ-CIM, XZ-CIM, and Z-CIM are common in the sense that the output *Z* is *in situ* stored in the memory cell. They all perform logic operations, mainly using emerging NVMs. Since the emerging NVMs are generally resistance-based memory, they are considered as a generic two-terminal resistive switching (RS) device, as shown in Fig. 3a. Typically, when the voltage across the device is sufficiently large with positive or negative polarities, it is switched to the high conductance state (HCS) by 'set' or the low conductance state (LCS) by 'reset', respectively. This description holds for RRAM, MRAM, and FTJ. Since only one switching polarity is usually used for CIM, the unipolar switching PCM can also be included in this model. The two conductance states encode the binary '1' and '0' as in conventional memory applications. For logic gates, the computation relies on the conditional switching of the device, as a function of

the states of other devices and the applied voltages. Such a non-linear characteristic can be viewed as an activation function in ANNs[32]. Consequently, it is possible that any RS-based NVM device could be employed for the three types of CIM.

NVM stateful logic of XYZ-CIM. Among the XYZ-CIM proposals, a prominent approach is based on the so-called stateful logic[39], achieved with NVM devices, typically RRAM. The implication (IMP) gate was originally proposed for stateful logic operations, as shown in Fig. 3b. The conductance state of one RRAM cell encodes the input operand *X*, while the other cell represents both input *Y* and output *Z* before and after the operation. The resistor's conductance is set approximately in the middle of the logarithmic values of LCS and HCS. The two WLs are applied with V_p (*e.g.* $V_w/2$) and V_w, respectively, where V_w is sufficiently large for a set transition but V_p is not, while the BL resistor is grounded. Upon activation, the final state of cell *Y*, *i.e.*, the output *Z*, is determined according to the IMP function. Specifically, if *Y* is initially in the HCS ('1'), the applied voltage polarity will not trigger the switching. If *Y* is initially in the LCS ('0'), its switching is conditional on *X*: if *X* is in the LCS, the BL potential will be close to 0 due to the isolation by the two LCS devices. Hence, the voltage drop across device *Y* is sufficient to switch it to HCS; if *X* is in the HCS, however, the applied voltage V_p will contribute significantly to raising the BL potential, thus preventing the switching of *Y*.

In Fig. 3c, a generic model of RRAM stateful logic is presented from the viewpoint of ANN, emerging the concept of stateful neural network (SFNN)[32]. This model uses two RRAM cells as inputs and one as output. Three WLs are applied with analog voltages (V_X, V_Y, and V_Z) that are calculated to determine the logic gate. The grounded BL resistor can be viewed as a parallel device applied with zero voltage. According to KCL, this circuit turns out to be a single-layer perceptron network, where the inputs are the conductance states (conductance G_i) of *X* and *Y*, the output is the final conductance state of *Z* (initialized as LCS), the weights are determined by the applied voltages, namely $w_i = V_Z - V_i - V_{set}$, with i representing devices *X*, *Y*, and BL resistor. The non-linear activation function in this model is provided by the set transition of device *Z* that mimics the hard-limit function, namely $Z = \begin{cases} 1, & \sum_i w_i G_i \geq 0 \\ 0, & \sum_i w_i G_i < 0 \end{cases}$. Based on SFNN, any linearly separable logic function can be performed with the circuit. Fig. 3c shows two

examples of NOR and NAND logic gates. Linearly inseparable functions such as XOR and one-bit full adder can be solved using a two-layer SFNN by cascading two operations of the circuit. The SFNN concept can be extended to the case where the output device is initialized as HCS and the BL is floating, resulting in another important stateful logic proposal, namely the MAGIC that performs the universal NOR logic[40]. The stateful logic concept is applicable to various NVM devices, and it has been extended to PCM[42] and MRAM[43,44].

DRAM bitwise logic of XYZ-CIM. DRAM bitwise logic is another important XYZ-CIM proposal, relying on the latch-type SA simultaneous rewrite during the computation. It can be realized with the commercial DRAM product with little or even no modifications[45,46]. Fig. 3d shows a column of three DRAM cells, with a latch-type SA at the end of the BL. The SA is a bi-stable circuit consisting of two CMOS inverters that form a positive feedback loop. The terminal connected to the BL acts as both the input and output node, through which the BL voltage is sensed and modified. When the BL voltage is higher (or lower) than $V_{DD}/2$, the SA quickly responds and stabilizes the output at V_{DD} (or 0). For logic operations, multiple rows are activated simultaneously. As a result, the circuit naturally performs the Majority logic function of the three-input cells, and the logic result is eventually re-stored in all three cells.

The BL is pre-charged to $V_{DD}/2$ in the first step to implement the bitwise logic. Then, three WLs are simultaneously activated with V_{DD}, while the SA is yet to be activated. The charges stored in the DRAM cells *X*, *Y*, and *Q* are shared among all DRAM capacitors (capacitance C_C) and the parasitic BL capacitor (capacitance C_B), resulting in the BL potential $V_{BL} = \frac{kC_C V_{DD} + C_B \cdot \frac{1}{2} V_{DD}}{3C_C + C_B}$, where $k = 0, 1, 2, 3$ is the number of cells at state '1'. Depending on the *k* value, V_{BL} might be higher or lower than $V_{DD}/2$. Specifically, if k=0 or 1, there is $V_{BL} < V_{DD}/2$, then upon the enablement of SA, the BL voltage is sensed and driven to be 0. If k=2 or 3, there is $V_{BL} > V_{DD}/2$, then the BL voltage will be driven to be V_{DD} by SA. Finally, following the Majority function, the stabilized BL voltage rewrites all three cells to a '0' or '1' state accordingly. By fixing the input *Q* as '1' or '0', the Majority circuit is reduced to the two-input AND or OR logic gate.

The Majority is a linearly separable logic function, and the DRAM circuit can be viewed as a single-layer perceptron, similar to the SFNN. In this model, the inputs are the stored voltage levels, the network weights are given by the capacitances, and the non-linear activation neuron is enabled by the latch-type SA whose threshold is $V_{DD}/2$. Based on the Majority gate, more complicated functions, *e.g.*, full adder, can be conveniently realized[47]. To enable a complete set of logic gates, the NOT gate can be designed by taking advantage of the complementary bit in the SA, which is written to a dual-contact cell through another select transistor[45]. In contrast to SFNN where all inputs are reserved, DRAM bitwise logic is destructive to logic inputs. To solve this issue, three rows in the array can be specially designed for logic operations. Additionally, before and after the logic operations, the RowClone operation[62], which copies data in a source row to a destination row by using the same charge sharing principle, should be performed to transfer the inputs and outputs within the array.

XZ-CIM. XZ-CIM also relies on the conditional switching of NVM devices, such as RRAM[48] and MRAM[49]. While stateful logic is conditional on the conductance states of two input memory cells, the two inputs for conditioning in XZ-CIM are represented by conductance state and voltage, respectively. Such an encoding method offers more convenience for constructing logic gates and enables one-step operation of the linearly inseparable functions such as XOR, but raises the cost of converting the heterogeneous attributes of input and output for cascading. A typical XZ-CIM logic gate based on two RRAM cells is shown in Fig. 3e. One input operand X is provided by an RRAM device conductance state, whereas the other input Y is encoded as the applied voltages[48]. The output is re-stored in the second cell that is initialized at the LCS (‘0’). The BL load resistor’s conductance is set between the LCS and the HCS, for appropriate voltage dividing. The output memory cell is applied with a constant voltage V_p subject $V_{set}/2 < V_p < V_{set}$, and then the voltages applied to WL1 and BL dictate the kind of logic gate. In the case of XOR, the WL1 and BL voltages are $(Y\text{-}1)V_p$ and $(\text{-}Y)V_p$, respectively. By changing the encoding scheme of the applied voltages, all 16 two-input Boolean logic gates can be realized with this circuit, which could be used to simplify the logic synthesis of complicated functions and thus reduce the latency of CIM.

Z-CIM. The NVM-based logic can be extended to Z-CIM, where both input operands *X* and *Y* are provided by applying voltages. The output *Z* is stored *in situ* as the conductance state of the memory cell (Fig. 3f). RRAM has been the most actively investigated object for Z-CIM[50-52], in addition to MRAM[53] and PCM[54]. It is essentially based on the conventional write operation of NVM, but with logic extensions to other input combinations traditionally considered ineffective. The RRAM switching depends on the polarity of the voltage drop and the initial conductance state. When the memory cell is initially in LCS (Z_0='0'), only the input combination of *X*='1' (V_w) and *Y*='0' switches the device to HCS, *i.e.*, *Z*='1', and in other input cases, the device remains at LCS (*Z*='0'). When the memory cell is initially in HCS (Z_0='1'), only the combination of *X*='0' and *Y*='1' (V_w) turns the device off, storing *Z*='0', and in other cases, it remains at *Z*='1'. The two situations correspond to the non-implication (NIMP) and the complementary implication (CIMP) functions. By fixing one input as '1' or '0', or by interchanging the operands applied to WL and BL, and cascading such operations, all the 14 linearly-separable logic gates can be implemented with the memory cell. The linearly inseparable XOR/XNOR are exceptional. To make them viable, the complementary RRAM concept based on stacking two resistive switches with opposite polarities should be used, by exploiting its asymmetric readout process[50]. Alternatively, the 1T1R cell can perform the XOR logic of Z-CIM more efficiently, thanks to the more terminals of the structure that facilitate convenient input operands encoding[63].

XY-CIM. XY-CIM has been proposed for logic operations as well, based on NVMs or SRAM. In the case of NVM[55,56], the two input operands are the binary conductance states (LCS or HCS) of memory cells. Basically, any NVM device featuring two distinct resistive states can be used for XY-CIM, including the intrinsic three-terminal devices such as FeFET[64]. As shown in Fig. 4a, upon the simultaneous activation of two WLs, the memory cell states are read out to BL, where the currents (I_{LCS} or I_{HCS}) are accumulated and sensed by a current-mode SA. The SA can be viewed as a binary neuron circuit to produce the logic output, with a reference current as the activation threshold. The four combinations of two inputs result in three distributions of BL currents centred at $2I_{LCS}$, $I_{LCS}+I_{HCS}$, and $2I_{HCS}$. Consequently, setting a threshold between $2I_{LCS}$ and $I_{LCS}+I_{HCS}$ (or between $I_{LCS}+I_{HCS}$ and $2I_{HCS}$) for the SA gives the linearly separable OR

(or AND) logic function. Given the neuronal activation is implemented with a CMOS circuit, it is convenient to have the inverses of the two logic gates, namely NOR and NAND. The combination of OR and NAND will result in the linearly inseparable XOR logic, obtained by applying successively two reference currents.

In the case of SRAM, the two input operands of XY-CIM are provided by the voltage levels stored in SRAM cells. The logic operation relies on sensing the voltage change of the pre-charged BL[57] (Fig. 4b). The core of an SRAM cell is a bi-stable circuit, whose two internal nodes store a binary voltage level and its complement. In the standard 6T SRAM structure, two select transistors control BL and the complementary BLB to access the two nodes. For logic operations, both BL and BLB are firstly pre-charged to V_{DD}, as in the SRAM readout process. Upon the simultaneous activation of two WLs, BL and BLB may discharge, depending on the states of the two SRAM cells. Specifically, only if both inputs *X* and *Y* are '1', the BL remains at V_{DD}. If there is a cell at state '0', BL discharges to a lower voltage. In the case of both cells at state '0', the BL voltage reduction is intensified. The SA output is recognized as the AND logic result by setting a reference voltage for the SA to distinguish V_{DD} from other reduced BL voltages. As the BLB accesses the complements of input bits *X* and *Y*, the SA adopting the same reference voltage delivers the NOR logic. Again, the combination of AND and NOR contributes to the XOR logic[65]. The bitwise logic operations with conventional 6T SRAM suffers from the disturbance of memory cells, due to the coupled write and read routes through the same port. When multiple WLs are simultaneously turned on, the BL/BLB might be discharged, which may, in turn, flip-flop the memory cell states. To overcome this issue, a main strategy is to decouple the write and read routes by adding access transistors or modifying their configurations, forming the 4+2T/8T/10T SRAM structures[65-67]. Furthermore, under-driven or asynchronous activation of WLs may also help solve the disturbance issue[67,68].

X-CIM. The underlying physical principle of X-CIM is fundamentally identical to XY-CIM, except for the different definitions of the input operands. Usually, X-CIM targets the implementation of parallel MAC operations, where one input operand is provided by a column of memory cells that represent a weight vector $\boldsymbol{x}$, and the other input is a vector $\boldsymbol{y}$ of voltages

externally applied to WLs or other lines. Consequently, the dot product of the two vectors is generated on the BL, in the form of an accumulation of currents or discharges, which is then sensed by the BL peripheral circuit. The intense interest in ANN accelerators has driven all memory technologies listed in Fig. 1c to be used for X-CIM[16-23], which is under active research[69-75], with efforts towards analog/digital hybrid, floating-point precision schemes[76,77]. Note that X-CIM may also imply logic operations, based on NVMs such as MRAM[78] and FeFET[79]. The efforts in this respect, however, have been overshadowed by the enormous volume of work on parallel MAC operations for AI accelerators.

Figs. 4c-4f list several X-CIM schemes with representative memory technologies. Fig. 4c presents the most straightforward case of a two-terminal NVM device, which is typical for RRAM and PCM. Each element of the weight vector $\boldsymbol{x}$ is encoded as a device conductance, and each $\boldsymbol{y}$ element is a sufficiently low voltage applied to the device. Based on Ohm's law and KCL, the current sensed at the BL represents the dot product $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}$. In the case of 1T1R structure that is widely adopted for NVMs, including RRAM, PCM, MRAM and FTJ, there is one more set of source lines (SLs) in the array, which offers another terminal for the dot product operation (Fig. 4d). The vector $\boldsymbol{y}$ may be applied through WLs or SLs, with the other set of lines being concurrently activated but encoding no information[16,80].

In NOR Flash-based X-CIM (Fig. 4e), each $\boldsymbol{x}$ weight element is represented by the amount of charge stored in the floating gate, which determines the channel conduction characteristics. Again, the input vector $\boldsymbol{y}$ may be applied through WLs or SLs. In the former case, the floating-gate transistors are used as gate-coupled programmable current mirrors, in combination with a column of input devices. The weight values are defined by the equivalent gate voltages of the transistors in the sub-threshold regime[21,81]. In the latter case, the floating-gate transistor is basically used as a programmable resistor in the Ohmic regime, whose conductance is related to the threshold voltage of transistor[82]. The latter principle should be applicable to FeFET-based X-CIM as well[20]. NAND Flash memory violates completely the random-access architecture, with memory cells being serially connected in a column, thus vitiating this X-CIM principle. With memory cell modifications, however, it has been proposed that the summation of voltages

or resistances can also lay the basis for dot product operation, with Flash memory or MRAM[83,84].

On the VM side, SRAM has drawn much attention for parallel MAC acceleration with X-CIM, thanks to its unique advantages and industrial maturity, including fast read/write speed, low power, unlimited endurance, and compatibility with the state-of-the-art logic process, albeit at the expense of large cell footprint. Fig. 4f shows a typical SRAM-based X-CIM scheme, where the weight vector $\boldsymbol{x}$ is stored as the voltage levels in the latch circuits, and the input vector $\boldsymbol{y}$ consists of WL voltages as usual[23]. Then, discharging the pre-charged BL to a certain level represents the dot product of the two vectors. Notably, in the latch circuits, the complement binary vector of $\boldsymbol{x}$ is also included, which would be an asset for signed computations. SRAM structure is of rich flexibilities for reliable, efficient X-CIM optimizations, but the 4T latch circuit always remains the core for weight storage. One strategy is to decouple the readout route to protect the SRAM cell from disturbance, by adopting the 8T/10T/12T structure[85-88], which, however, even aggravates the cell footprint issue. In several schemes based on the standard 6T SRAM array[89], specially designed local computing units and global bit-lines have been included, thus, to balance the trade-off between circuit functionality and area overhead.

Depending on the memory technology, the weight vector $\boldsymbol{x}$ stored in memory cells could be single-bit or multi-bit, as illustrated in Fig. 4g. While VMs are generally single-bit devices, many NVMs show multi-bit and even analog states, which is a key enabler for enhancing the X-CIM throughput that is highly desired for machine learning accelerations. Flash, FeFET, PCM, RRAM, and FTJ are excellent analog conductance devices, thanks to their fundamental physics that allows for continuous tuning of state variables such as charge storage, ferroelectric polarity ratio, crystalline volume, and conducting filament diameter[12]. Accordingly, these NVM devices show large memory windows, ranging from 10 to 10^6 that allow to accommodate multi-bit information stored in one single cell[90,91]. Due to its small conductance switching ratio, MRAM is considered a single-bit memory, although there are undergoing efforts to develop multi-bit devices[92]. To maximize computing efficiency, input vector $\boldsymbol{y}$ is usually encoded as multi-bit values such as WL voltage pulses with analog magnitude or width[23,93], although serial binary pulses might be adopted to save the data conversion cost[94,95]. As a result, the CIM

operation is carried out in the current domain, time domain, or charge domain. Given that both $\boldsymbol{x}$ and $\boldsymbol{y}$ might be binary or analog values or bipolar values enabled by the differential operation, the multiplication of two elements might be in the form of AND logic, bipolar XNOR, or purely analog result (Fig. 4h). Due to the multi-bit inputs, and the simultaneously activated multiple WLs (typically >>2), sensing of the dot product requires a conversion circuit that quantizes many discrete output levels (Fig. 4i), which is usually achieved with a multi-level SA or analog-to-digital converter (ADC). SA and ADC are usually much larger and more power-intensive than memory cells. Therefore, along with the *in situ* computing and the inherent parallelism of X-CIM, otherwise SA or ADC has become another efficiency bottleneck, requesting design optimizations to maintain the performance improvement offered by CIM[96].

NVM-based X-CIM may also stay in the analog domain through readout with transimpedance amplifiers (TIAs), for fully-analog cascading of computations[97]. Specifically, by storing a weight matrix $\boldsymbol{X}$ as analog conductance in a memory array, and upon the application of the input voltage vector $\boldsymbol{y}$, the outputs of TIAs give the MVM result, namely $\boldsymbol{z} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$ (Fig. 4j). Along with MVM, other basic matrix operations can be accelerated with crosspoint NVM arrays. Fig. 4h shows the matrix inversion circuit that solves a system of linear equations $\boldsymbol{Xz} = \boldsymbol{y}$, which is exactly the inverse problem of MVM. A set of negative feedback operational amplifiers (OPAs) connect the crosspoint WLs and BLs one-to-one, forming a closed-loop circuit[98]. It can provide the solution $\boldsymbol{z} = \boldsymbol{X}^{-1}\boldsymbol{y}$ in one computational step, which is represented by the BL voltages. This concept has been extended to solve matrix eigenvectors and generalized inverses[98,99].

O-CIM. Recently, SRAM-based CIM has progressively shifted to the fully digital domain, by incorporating conventional logic gates in the vicinity of memory cells. As there is no fusion of memory cells during the computation process, it is reasonably termed O-CIM. In this approach, one input operand is provided externally through a specially designed line, and the other input is read out from a SRAM cell and fed to the neighbouring logic gate, which carries out the multiplication of the two operands. To sum up the multiplication results, a hierarchy of adder trees must be deployed nearby, producing partial sums in the digital domain[60]. In the case of DRAM, rather than embedding individual logic gates around memory cells, conventional

computational blocks are built close to the arrays, thus utilizing the array-level parallelism for MAC acceleration[61]. O-CIM is usually designed with mature volatile memories, to seek industrial compatibility with contemporary commercial products[100]. By incorporating standard digital computing units, O-CIM is also more reliable than other types of CIM based on analog computation.

Discussion

CIM is a disruptive technology over the traditional von Neumann computer in two aspects, namely fusing memory and computing, and providing spatial parallelism for computing acceleration. Also, it may enable highly efficient computations by utilizing unconventional while powerful logic gates, *e.g.*, the Majority function[101], or by mapping directly the arithmetic operations to hardware circuits. Compared to the conventional memory mode, CIM is enabled by simultaneously activating multiple WLs to initiate the interaction of memory cells through physical laws in the array. To do so, many CIM technologies have been developed, majorly based on the crosspoint array architecture that accommodates almost all memory technologies. Emerging NVMs have been the initiators and important candidates of CIM[39], allowing for the unrestricted exploration of beyond-memory applications. Conversely, mature memory technologies such as SRAM and DRAM, favor fewer modifications. DRAM has been highly optimized for storage density and leakage reduction, thus disfavoring process modifications, albeit the CIM causes merely an overhead of less than 1% in the array[45]. SRAM is more advantageous in terms of its flexibility in the modern logic fabrication process that allows for customized memory array designs, although the standard 6T SRAM is fairly appreciated[89].

By elaborating on the fundamental principles of CIM technologies, the spectrum proposed in this Review provides an overview of different types of CIM in a parallel manner. In Table 1, we summarize the main features, advantages and disadvantages, as well as challenges regarding device reliability and computing efficiency. In the CIM paradigm, logic gates are developed to provide a functionally complete logic set for universal computations, while parallel MAC is developed for accelerating specific applications such as neural networks. Among the different CIM schemes for logic gates, there are two factors that distinguish each other, namely depending on whether the computation (*i.e.*, nonlinear activation) is performed by a passive RS

cell, and whether the physical attributes of input and output operands are identical. In the former situation, while the implementations of NVM stateful logic, XZ-CIM, and Z-CIM that use a RS cell as a neuron are highly compact, the DRAM bitwise logic and XY-CIM scheme requiring an additional active SA for computation sacrifices some of the area efficiency. In the latter situation, XYZ-CIM is considered a real CIM paradigm, as all input/output operands are represented *in situ* by memory cells in the array[102]. It is free of a conversion process, thus enabling easy cascading and benefitting the overall latency of sequential processing. By contrast, XZ-CIM, Z-CIM, and XY-CIM of NVMs require additional operations to read out the output as voltage (or to write the output as conductance) for cascading next logic gate, causing a delay that hinders the throughput improvement. In the case of SRAM, a special unit might be designed to write the logic output in a cell for succeeding access[67].

X-CIM and O-XIM are both used for parallel MAC, but in totally different manners, that is analog vs digital[103]. The advantages and disadvantages of each scheme are obvious. X-CIM is able to deliver high energy/area efficiency, thanks to the efficient way of direct mapping of the computing primitive to the memory array. However, also due to the analog nature of the computing process, it suffers from the accuracy degradation caused by non-idealities of device (*e.g.*, resistive cell, capacitor, or transistor), array, and circuit. It is also possible to increase the resolution of input, memory capacity of NVM, and the parallelism of CIM cells within one operation, to significantly improve the computing throughput. However, such a benefit comes with a remarkable overhead of DAC for input and ADC for output. In X-CIM of NVMs, neural networks have been a well-posed application, where the MVM can be naturally cascaded thanks to the stationary weight matrices between every two layers of neurons. For general-purpose applications, due to the isolation of one vector in the memory cells, attribute conversion processes may be required for operation cascading as in the logic gate cases. O-CIM works in the digital domain with conventional CMOS logic gates, hence it affords much more robust computations. Also, because of the elimination of ADC and DAC, the data conversion burden is overcome. Contrary to analog X-CIM where the mapping to hardware is fixed, digital O-CIM has a higher flexibility to be accommodated for a wider range of problems. Nevertheless, it has traded off the area and energy efficiencies, as each processing element consists of a

multiplier and an adder for MAC operations[104], which actually resembles other digital accelerators such as systolic array-based designs.

Among the CIM types for logic operations, NVM stateful, XZ-CIM, and Z-CIM rely on the dynamical RS of memory cell, thus desiring high endurance of device to support the frequent logic gate operations[105]. As such, PCM, whose endurance has been reported to be $>10^{10}$, appears to be more suitable (though not enough yet) for these kinds of CIM. RRAM, whose state-of-the-art endurance is quite limited, $>10^6$, must be significantly improved to be qualified[106,107]. MRAM performs even better, generally showing an endurance of 10^{12}. However, it is intrinsically limited by the small memory window that presents a barrier for reliable analog computing, especially for multi-input logic gates[108,109]. Since XY-CIM and X-CIM are basically a parallel readout process of two or more cells, any memory device featuring two or more distinct states could be employed. Consequently, the temporal retention property and state variations of NVM devices should be good enough to guarantee reliable readout, summation, and discretization by SA or ADC for logic operation or parallel MAC, respectively. Although NVM devices generally show sufficient retention (>10 years at 85℃) for traditional binary memory application[106], the linear superposition of two or more cell states in logic operation should impose more strict restrictions on the retention performance. Particularly, in X-CIM for parallel MAC, it is highly desired to have a multi-bit cell, and usually multiple cells are simultaneously activated to achieve an enhanced throughput. In this context, all NVM devices except for MRAM show large memory windows, enabling the multi-bit storage. Because of the critical limitation by state variations, however, the state-of-the-art NVMs can only deliver reliable 2-bit capacity, as summarized in a recent excellent Review paper[110]. Therefore, there remain a large space for device optimizations to achieve higher capacity of a memory cell, where the mature multi-level NAND Flash could be a good reference[111]. Note that the analog conductance of NVMs has been excessively utilized for computations in many cases, to achieve high equivalent throughputs and energy efficiencies for typical applications such as AI accelerators. However, one emphasis should be reiterated that the conventional memory mode should always be preserved, where the multiple states should be distinguishable with sufficient readout margins. To develop reliable resistive NVM-based CIM technologies, there have been

a large number of efforts at the algorithm and system levels, which, however, are usually limited to specific applications[112-114]. Strategies such as bit slicing, divide-and-conquer, and compensation have been used to extend computing precision in large-scale problems. While such solutions are conveniently applicable to the forward matrix multiplications, the problems become intractable for the matrix inversions of X-CIM, leaving a space to be explored towards the resistive NVM-based general matrix computations.

The inadequate endurance property of NVM devices represents the major issue of stateful logic, XZ-CIM, and Z-CIM, preventing them from moving forward to practical applications. For stateful logic and XZ-CIM, since both rely on analog multiplication, summation, and nonlinear activation, it is critical to have low cycle-to-cycle and device-to-device variations of set/reset voltages, and LCS and HCS, to limit the bit error rate of logic operation. Z-CIM is free of analog multiplication and summation, hence it only requires low variations of set/reset voltages. By contrast, XY-CIM and X-CIM favor low LCS and HCS variations for reliable CIM operations. Particularly, in parallel MAC with multi-bit NVM devices, the conductance distributions of input memory cells and then the dot product results become even more complicated, necessitating designs of delicate readout circuits (SA/ADC). Additionally, X-CIM may be involved in neural network training, it is therefore important to have a good linearity of conductance updates towards the minimal or maximal bounds of the conductance range. To overcome the challenges of NVM device endurance, spatial and temporal uniformity, and update linearity, which are fundamentally controlled by the device physics, it should be promising to carry out optimizations in terms of device materials and structures to solve these issues at the source[115,116]. All these CIM schemes rely on analog computation, which would be easily disturbed by the process-voltage-temperature (PVT) variations. There have been rare investigations into the PVT issue of CIM with emerging NVMs, manifesting a pressing challenge towards practical applications[117]. O-CIM works with mature digital circuit designs, such issues have usually been well investigated and thus represent a less concern.

To maximize the energy efficiency of CIM, several issues should be addressed for these CIM schemes. In the RS-based CIM schemes, every event causes a significant amount of power consumption, hence optimizations of set/reset voltages and currents are highly demanded. In

XY-CIM and X-CIM that work with stationary memory cell states, the lower absolute conductance of NVM device should help reduce the energy dissipation. That said, such optimizations are not easy, as low conductance is usually accompanied with nonlinear current-voltage characteristics, which would introduce additional computing errors. Therefore, there should be a dilemmatic tradeoff to account for both issues. In the synthesis of complicated logic operations with a functionally complete logic set, the choice of methods is quite diverse. It could be composed of many 2-input gates or few multi-input gates, resulting in a huge difference regarding hardware and latency costs, which accompany with different sensitivities to analog non-idealities. As a result, there should exist a tradeoff consideration for the reliable and efficient logic synthesis. Regarding X-CIM, it has been struggling to deal with the overhead of ADC and DAC, which is virtually the major challenge for energy efficiency improvement. In O-CIM, while multiplier is convenient to implement by using a single logic gate, *e.g.*, NOR gate, the adder tree has been the accepted bottleneck. Therefore, more efforts are greatly appreciated in this aspect to improve the energy and area efficiencies.

Lastly, this spectrum unifies various CIM technologies as a continuum with clear identifications of fundamental principles, it is expected to inspire novel research directions and novel, efficient, optimized CIM schemes to further enrich the design space. Since it provides a platform for model abstraction of all possible CIM schemes, it should also be beneficial to the benchmarking of CIM in terms of computational complexity, latency, throughput, and energy efficiency[118,119]. While usually only one type of CIM is considered for a special purpose, integrating different types of CIM based on the same memory technology would be a promising alternative to combine the advantages while avoiding the disadvantages of each. Additionally, while the parallel MAC of X-CIM has been demonstrated with a high potential for computing accelerations, it is inherently incapable of a general-purpose computing system. In this context, the combination of logic gates and parallel MAC operations in the CIM paradigm would be worth more future investments.

REFERENCES

1. Horowitz, M. Computing's energy problem (and what we can do about it). *IEEE ISSCC Dig. Tech. Papers*, 10-14 (IEEE, 2014).
2. Tsividis, Y. Not your father's analog computer. *IEEE Spectr.* **55**, 38-43 (2017).
3. Sebastian, A. et al. Memory devices and applications for in-memory computing. *Nat. Nanotechnol.* **15**, 529-544 (2020).
4. Ankit, A. et al. Circuits and architectures for in-memory computing-based machine learning accelerators. *IEEE Micro* **40**, 8-22 (2020).
5. Mannocci, P. et al. In-memory computing with emerging memory devices: Status and outlook. *APL Mach. Learn.* **1**, 010902 (2023).
6. Mutlu, O. et al. A modern primer on processing in memory. In *Emerging Computing: From Devices to Systems: Looking Beyond Moore and Von Neumann*, 171-243 (Springer, 2023).
7. Kang, W. et al. Spintronic logic-in-memory paradigms and implementations. In *Applications of Emerging Memory Technology*, 215-229 (Springer, 2020).
8. Wan, W. et al. A compute-in-memory chip based on resistive random-access memory. *Nature* **608**, 504-512 (2022)
9. Verma, N. et al. In-memory computing: Advances and prospects. *IEEE J. Solid-State Circuits* **11**, 43-55 (2019).
10. Jhang, C. J. et al. Challenges and trends of SRAM-based computing-in-memory for AI edge devices. *IEEE Trans. Circuits Syst. I Reg. Papers* **68**, 1773-1786 (2021).
11. Oliveira, G. F. et al. Accelerating Neural Network Inference with Processing-in-DRAM: From the Edge to the Cloud. *IEEE Micro* **42**, 25-38 (2022).
12. Wang, Z. et al. Resistive switching materials for information processing. *Nat. Rev. Mater.* **5**, 173-195 (2020).
13. Xiao, T. P. et al. Analog architectures for neural network acceleration based on non-volatile memory. *Appl. Phys. Rev.* **7**, 031301 (2020).
14. Khan, A. I., Keshavarzi, A. & Datta, S. The future of ferroelectric field-effect transistor technology. *Nat. Electron.* **3**, 588-597 (2020).
15. Hu, M. et al. Hardware realization of BSB recall function using memristor crossbar arrays.

In *Proc 49th Annu. Design Automation Conf. (DAC)*, 498-503 (2012).

16. Chen, W.-H. et al. CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors. *Nat. Electron.* **2**, 420-428 (2019).
17. Burr, G. W. et al. Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Trans. Electron Devices* **62**, 3498-3507 (2015).
18. Patil, A. D. et al. An MRAM-based deep in-memory architecture for deep neural networks. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, (IEEE, 2019).
19. Berdan, R. et al. Low-power linear computation using nonlinear ferroelectric tunnel junction memristors. *Nat. Electron.* **3**, 259-266 (2020).
20. Jerry, M. et al. Ferroelectric FET analog synapse for acceleration of deep neural network training. In *Technical Digest of the International Electron Devices Meeting (IEDM)*, 6.2.1-6.1.4 (IEEE, 2017).
21. Guo, X. et al. Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology. In *Technical Digest of the International Electron Devices Meeting (IEDM)*, 6.5.1-6.5.4 (IEEE, 2017).
22. Xie, S. et al. eDRAM-CIM: Compute-in-memory design with reconfigurable embedded-dynamic-memory array realizing adaptive data converters and charge-domain computing. *IEEE ISSCC Dig. Tech. Papers*, 248-249 (2021).
23. Zhang, J., Wang, Z. & Verma, N. In-memory computation of a machine-learning classifier in a standard 6T SRAM array. *IEEE J. Solid-State Circuits* **52**, 915-924 (2017).
24. Su, Y. et al. CIM-Spin: A Scalable CMOS Annealing Processor With Digital In-Memory Spin Operators and Register Spins for Combinatorial Optimization Problems. *IEEE J. Solid State Circuits* **57**, 2263-2273 (2022).
25. Xie, S. et al. Ising-CIM: A Reconfigurable and Scalable Compute Within Memory Analog Ising Accelerator for Solving Combinatorial Optimization Problems. *IEEE J. Solid State Circuits* **57**, 3453- 3465 (2022).
26. Pagiamtzis, K. & Sheikholeslami, A. Content-addressable memory (CAM) circuits and architectures: a tutorial and survey. *IEEE J. Solid State Circuits* **41**, 712–727 (2006).
27. Ni, K. et al. Ferroelectric ternary content-addressable memory for one-shot learning. *Nat.*

Electron. **2**, 521–529 (2019).

28. Li, J. et al. 1 Mb 0.41 μm2 2T-2R cell nonvolatile TCAM with two-bit encoding and clocked self-referenced sensing. *IEEE J. Solid-State Circuits* **49**, 896–907 (2014).
29. Li, C. et al. Analog content-addressable memories with memristors. *Nat. Commun.* **11**, 1–8 (2020).
30. Lin, C. C. et al. A 256b-wordlength ReRAM-based TCAM with 1ns search-time and 14× improvement in word length-energy efficiency-density product using 2.5T1R cell. *IEEE ISSCC Dig. Tech. Papers*, 136–137 (IEEE, 2016).
31. Vahdat, S. et al. Interstice: Inverter-based memristive neural networks discretization for function approximation applications. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **28**, 1578-1588 (2020).
32. Sun, Z. et al. Logic computing with stateful neural networks of resistive switches. *Adv. Mater.* **30**, 1802554 (2018).
33. Imani, M. et al. Ultra-efficient processing in-memory for data intensive applications. In *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 1–6 (IEEE, 2017).
34. Guckert, L. & Swartzlander, E. E. Optimized memristor-based multipliers. *IEEE Trans. Circuits Syst. I, Reg. Papers* **64**, 373–385 (2017).
35. Radakovits, D. et al. A memristive multiplier using semi-serial IMPLY-based adder. *IEEE Trans. Circuits Syst. I, Reg. Papers* **67**, 1495–1506 (2020).
36. Haj-Ali, A. et al. IMAGING: In-memory ALGorithms for image processing. *IEEE Trans. Circuits Syst. I, Reg. Papers* **65**, 4258–4271 (2018).
37. Imani, M. et al. FloatPIM: In-memory acceleration of deep neural network training with high precision. In *Proc. Annu. Int. Symp. Comput. Archit. (ISCA)*, 802–815 (IEEE, 2019).
38. Leitersdorf, O. et al. AritPIM: High-Throughput In-Memory Arithmetic. *IEEE Trans. Emerg. Topics Comput.* (2023). Early access, DOI: 10.1109/TETC.2023.3268137.
39. Borghetti, J. et al. 'Memristive' switches enable 'stateful' logic operations via material implication. *Nature* **464**, 873-876 (2010).
40. Kvatinsky, S. et al. MAGIC - memristor-aided logic. *IEEE Trans*. *Circuits Syst. II Express Briefs* **61**, 895-899 (2014).
41. Kim, Y. S. et al. Ternary Logic with Stateful Neural Networks Using a Bi-layered TaOx-

Based Memristor Exhibiting Ternary States. *Adv. Sci.* **9**, 2104 (2022).

42. Hoffer, B. et al. Stateful Logic Using Phase Change Memory. *IEEE J. Explor. Solid-State Comput. Devices Circuits* **8**, 77-83 (2022).
43. Zabihi, M. et al. In-memory processing on the spintronic CRAM: from hardware design to application mapping. *IEEE Trans. Comput.* **68**, 1159-1173 (2019).
44. Hoffer, B. & Kvatinsky, S. Performing Stateful Logic Using Spin-Orbit Torque (SOT) MRAM. In *IEEE 22nd Int. Conf. Nanotechnol. (NANO)*, 571-574 (2022).
45. S Seshadri, V. et al. Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology. In *Proc. International Symposium on Microarchitecture (MICRO)*, 273-287 (IEEE, 2017).
46. Gao, F., Tziantzioulis, G. & Wentzlaff, D. ComputeDRAM: In-memory compute using off-the-shelf DRAMs. In *Proc. International Symposium on Microarchitecture (MICRO)*, 100-113 (2019).
47. Ali, M. F., Jaiswal, A. & Roy, K. In-memory low-cost bit-serial addition using commodity DRAM technology. *IEEE Trans. Circuits Syst. I Reg. Papers* **67**, 155-165 (2020).
48. Song, Y. et al. Reconfigurable and Efficient Implementation of 16 Boolean Logics and Full-Adder Functions. *Adv. Sci.* **2022**, 2200036 (2022)
49. Kim, D. et al. BiMDiM: Area efficient Bi-directional MRAM Digital in-Memory Computing. In Proc. *International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 74-77 (IEEE, 2022).
50. Linn, E. et al. Beyond von neumann-logic operations in passive crossbar arrays alongside memory operations. *Nanotechnol.* **23**, 305205 (2012).
51. Gao, S. et al. Implementation of Complete Boolean Logic Functions in Single Complementary Resistive Switch. *Sci. Rep.* **5**, 15467 (2015)
52. Gaillardon, P. E. et al. The Programmable Logic-in-Memory (PLiM) Computer. In *Proc. Design Autom. Test Eur. Conf. Exhib. (DATE)*, 427-432 (2016)
53. Zhang, H. et al. Stateful reconfigurable logic via a single-voltage-gated spin Hall-effect driven magnetic tunnel junction in a spintronic memory. *IEEE Trans. Electron Devices* **64**, 4295-4301 (2017).
54. Cassinerio, M., Ciocchini, N. & Ielmini, D. Logic computation in phase change materials

by threshold and memory switching. *Adv. Mater.* **25**, 5975-5980 (2013).

55. Chen, W.-H. et al. A 16 Mb dual-mode ReRAM macro with sub-14 ns computing-in-memory and memory functions enabled by self-write termination scheme. In *Technical Digest of the International Electron Devices Meeting (IEDM)*, 28.2.1-28.2.4 (IEEE, 2017).
56. Jain, S. et al. Computing in memory with spin-transfer torque magnetic RAM. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **26**, 470-483 (2018).
57. Jeloka, S. et al. A 28 nm configurable memory (TCAM/BCAM/SRAM) using push-rule 6T bit cell enabling logic-in-memory. *IEEE J. Solid-State Circuits* **51**, 1009-1021 (2016).
58. Eckert, C. et al. Neural cache: bit-serial in-cache acceleration of deep neural networks. In *Proc. 45th Ann. Int. Symp. Computer Architecture (ISCA)*, 383-396 (IEEE, 2018).
59. Kim, H. et al. A 1-16b precision reconfigurable digital in-memory computing macro featuring column-MAC architecture and bit-serial computation. In *Proc. IEEE 45th Eur. Solid State Circuits Conf. (ESSCIRC)*, 345-348, (IEEE, 2019).
60. Chih, Y.-D. et al. An 89 TOPS/W and 16.3 TOPS/mm² all-digital SRAM-based full-precision compute-in memory macro in 22 nm for machine-learning edge applications. *IEEE ISSCC Dig. Tech. Papers*, 252-253 (IEEE, 2021).
61. Kwon, Y.-C. et al. A 20nm 6GB Function-In-Memory DRAM Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism for Machine Learning Applications. *IEEE ISSCC Dig. Tech. Papers*, 350-351 (IEEE, 2021).
62. Seshadri, V. et al. RowClone: fast and energy-efficient in-DRAM bulk data copy and initialization. In *Proc. International Symposium on Microarchitecture (MICRO)*, 185-197 (ACM, 2013).
63. Wang, Z. et al. Functionally complete Boolean logic in 1T1R resistive random-access memory. *IEEE Electron Device Lett.* **38**, 179-182 (2017).
64. Breyer, E. T. et al. Perspective on ferroelectric, hafnium oxide based transistors for digital beyond von-Neumann computing. *Appl. Phys. Lett.* **118**, 050501 (2021).
65. Dong, Q. et al. A 4 + 2T SRAM for searching and in-memory computing with 0.3-V VDDmin. *IEEE J. Solid-State Circuits* **53**, 1006-1015 (2018).
66. Zhang, Y. et al. Recryptor: a reconfigurable cryptographic cortex-M0 processor with in-memory and near-memory computing for IoT security. *IEEE J. Solid-State Circuits* **53**,

995-1005 (2018).

67. Agrawal, A. et al. X-SRAM: enabling in-memory Boolean computations in CMOS static random-access memories. *IEEE Trans. Circuits Syst. I* **65**, 4219-4232 (2018).
68. Lee, K. et al. Bit parallel 6T SRAM in-memory computing with reconfigurable bit-precision. In *Proc. 57th ACM/IEEE Design Automat. Conf. (DAC)*, (2020).
69. Hung, J. M. et al. A four-megabit compute-in-memory macro with eight-bit precision based on CMOS and resistive random-access memory for AI edge devices. *Nat. Electron.* **4**, 921-930 (2021).
70. Huo, Q. et al. A computing-in-memory macro based on three-dimensional resistive random-access memory. *Nat. Electron.* **5**, 469–477 (2022).
71. Wang, W. et al. A memristive deep belief neural network based on silicon synapses. *Nat. Electron.* **5**, 870-880 (2022).
72. Cui, J. et al. CMOS-compatible electrochemical synaptic transistor arrays for deep learning accelerators. *Nat. Electron.* **6**, 292-300 (2023).
73. Huang, X. et al. An ultrafast bipolar flash memory for self-activated in-memory computing. *Nat. Nanotechnol.* **18**, 486-492 (2023).
74. Hsieh, S.-E. et al. A 70.85-86.27TOPS/W PVT-Insensitive 8b Word-Wise ACIM with Post-Processing Relaxation. *IEEE ISSCC Dig. Tech. Papers*, 136-137 (IEEE, 2023).
75. Chen, P. et al. A 22nm Delta-Sigma Computing-In-Memory (ΔΣCIM) SRAM Macro with Near-Zero-Mean Outputs and LSB-First ADCs Achieving 21.38TOPS/W for 8b-MAC Edge AI Processing. *IEEE ISSCC Dig. Tech. Papers*, 140-141 (IEEE, 2023).
76. Wu, P.-C. et al. A 22nm 832Kb hybrid-domain floating-point SRAM in-memory-compute macro with 16.2-70.2 TFLOPS/W for high-accuracy AI-edge devices. *IEEE ISSCC Dig. Tech. Papers*, 126-127 (IEEE, 2023).
77. Yue, J. et al. A 28nm 16.9-300TOPS/W Computing-in-Memory Processor Supporting Floating-Point NN Inference/Training with Intensive-CIM Sparse-Digital Architecture. *IEEE ISSCC Dig. Tech. Papers*, 252-253 (IEEE, 2023).
78. Ikeda, S. et al. Magnetic tunnel junctions for spintronic memory and beyond. *IEEE Trans. Elect. Dev.* **54**, 991-1002 (2006).
79. Yin, X. et al. Ferroelectric FETs-based nonvolatile logic-in-memory circuits. *IEEE Trans.*

Very Large Scale Integration (VLSI) Syst. **27**, 159-172 (2019).

80. Li, C. et al. Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **1**, 52-59 (2018).
81. Danial, L. et al. Two-terminal floating-gate transistors with a low-power memristive operation mode for analogue neuromorphic computing. *Nat. Electron.* **2**, 596-605 (2019).
82. Agarwal, S. et al. Using floating-gate memory to train ideal accuracy neural networks. *IEEE J. Explor. Solid-State Comput. Devices Circuits* **5**, 52-57 (2019).
83. Lin, Y.-Y. et al. A novel voltage-accumulation vector-matrix multiplication architecture using resistor-shunted floating gate flash memory device for low-power and high-density neural network applications. In *2018 IEEE International Electron Devices Meeting (IEDM)*, 2.4.1-2.4.4 (IEEE, 2018).
84. Jung, S. et al. A crossbar array of magnetoresistive memory devices for in-memory computing. *Nature* **601**, 211-216 (2022).
85. Jaiswal, A. et al. 8T SRAM cell as a multibit dot-product engine for beyond von Neumann computing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **27**, 2556-2567 (2019).
86. Ali, M., Agrawal, A. & Roy, K. RAMANN: In-SRAM Differentiable Memory Computations for Memory-Augmented Neural Networks. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 61-66 (2020).
87. Biswas, A. & Chandrakasan, A. P. CONV-SRAM: an energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks. *IEEE J. Solid-State Circuits* **54**, 217-230 (2019).
88. Jiang, Z. et al. XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks. In *Proc. Symposium on VLSI Technology*, 173-174 (IEEE, 2018).
89. Si, X. et al. 15.5 A 28nm 64Kb 6T SRAM Computing-in-Memory Macro with 8b MAC Operation for AI Edge Chips. *IEEE ISSCC Dig. Tech. Papers*, 246-247 (IEEE, 2020).
90. Chung, W., Si, M. & Ye, P. D. First demonstration of Ge ferroelectric nanowire FET as synaptic device for online learning in neural network with high number of conductance state and Gmax/Gmin. In *Technical Digest of the International Electron Devices Meeting (IEDM)*, 15.2.1-15.2.4 (IEEE, 2018).

91. Giannopoulos, I. et al. 8-bit precision in-memory multiplication with projected phase-change memory. In *Technical Digest of the International Electron Devices Meeting (IEDM)*, 27.7.1-27.7.4 (IEEE, 2018).
92. Lequeux, S. et al. A magnetic synapse: multilevel spin-torque memristor with perpendicular anisotropy. *Sci. Rep.* **6**, 1-7 (2016).
93. Gonugondla, S. K., Kang, M. & Shanbhag, N. A 42pJ/decision 3.12TOPS/W robust in-memory machine learning classifier with on-chip training. *IEEE ISSCC Dig. Tech. Papers*, 490-491 (IEEE, 2018).
94. Dong, Q. et al. 15.3 A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications. *IEEE ISSCC Dig. Tech. Papers*, 242-243 (IEEE, 2020).
95. Yao, P. et al. Fully hardware-implemented memristor convolutional neural network. *Nature* **577**, 641-646 (2020).
96. Yu, S. et al. Computing-in-memory chips for deep learning: recent trends and prospects. *IEEE Circ. Syst. Mag.* **21**, 31-56 (2021).
97. Hu, M. et al. Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix-vector multiplication. In *Proc. 53rd Annual Design Automation Conference (DAC)*, 1-6 (ACM, 2016).
98. Sun, Z. et al. Solving matrix equations in one step with cross-point resistive arrays. *Proc. Natl. Acad. Sci. USA* **116**, 4123-4128 (2019).
99. Sun, Z. et al. One-step regression and classification with cross-point resistive memory arrays. *Sci. Adv.* **6**, eaay2378 (2020).
100. Devaux, F. The true processing in memory accelerator. *Proc. Hot Chips 31 Symp.*, 1-24 (IEEE, 2019).
101. Reuben, J. & Pechmann, S. Accelerated addition in resistive RAM array using parallel-friendly majority gates. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **29**, 1108-1121 (2021).
102. Haj-Ali, A. et al. Not in name alone: A memristive memory processing unit for real in-memory processing. *IEEE Micro* **38**, 13-21 (2018).
103. Seo, J.-S. et al. Digital versus analog artificial intelligence accelerators: Advances trends

and emerging designs. *IEEE Solid-State Circuits Mag.* 14, 65-79 (2022).

104. Kim, D. et al. An overview of processing-in-memory circuits for artificial intelligence and machine learning. *IEEE J. Emerg. Sel. Topics Circuits Syst.* **12**, 338-353 (2022).
105. Kim, Y. S. et al. Stateful in-memory logic system and its practical implementation in a TaOx-based bipolar-type memristive crossbar array. *Adv. Intell. Syst.* **2**, 1900156 (2020).
106. Yu, S. *Semiconductor Memory Devices and Circuits*. (CRC Press, 2022).
107. Banerjee, W. Challenges and applications of emerging nonvolatile memory devices. *Electron.* **9**, 1029 (2020).
108. Wolf, S. A. et al. The promise of nanomagnetics and spintronics for future logic and universal memory. *Proc. IEEE* **98**, 2155–2168 (2010).
109. Endoh, T. et al. An overview of nonvolatile emerging memories—spintronics for working memories. *IEEE J. Emerg. Sel. Top. Circ. Syst.* **6**, 109–119 (2016).
110. Kim, K.-H. et al. Wurtzite and fluorite ferroelectric materials for electronic memory. *Nat. Nanotechnol.* **18**, 422-441 (2023).
111. Goda, A. Recent progress on 3D NAND flash technologies. *Electron.* **10**, 3156 (2021).
112. Chen, P.-Y. et al. Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip. In *Proc. Design Autom. Test Eur. Conf. Exhib. (DATE)*, 854-859 (IEEE, 2015).
113. Feinberg, B., Wang, S. & Ipek, E. Making memristive neural network accelerators reliable. In *Proc. The International Symposium on High Performance Computer Architecture (HPCA)*, 52-65 (IEEE, 2018).
114. Jain, S. & Raghunathan, A. CxDNN: Hardware–software compensation methods for deep neural networks on resistive crossbar systems. *ACM Trans. Embedded Comput. Syst.* **18**, 1-23 (2019).
115. Choi, S. et al. SiGe epitaxial memory for neuromorphic computing with reproducible high performance based on engineered dislocations. *Nat. Mater.* **17**, 335-340 (2018).
116. Nukala, P. et al. Reversible oxygen migration and phase transitions in hafnia-based ferroelectric devices. *Science* **372**, 630-635 (2021).
117. Boybat, I. et al. Temperature sensitivity of analog in-memory computing using phase-change memory. In *Technical Digest of the International Electron Devices Meeting*

(IEDM), 28.3.1-28.3.4 (IEEE, 2021).

118. Shanbhag, N. R. & Roy, S. K. Comprehending in-memory computing trends via proper benchmarking. *In 2022 IEEE Custom Integrated Circuits Conference (CICC)*, 1–7 (2022).
119. Houshmand, P. et al. Benchmarking and modeling of analog and digital SRAM in-memory computing architectures. arXiv: 2305.18335 (2023).

Acknowledgements

This work has received funding from the National Key R&D Program of China (2020YFB2206001), the National Natural Science Foundation of China (62004002, 92064004, 61927901), and the 111 Project (B18001).

Author Contributions

All authors contributed to the preparation of the manuscript.

Competing interests

A.M. is a founder and director of Intrinsic Semiconductor Technologies Ltd (www.intrinsicst.com), a spin-out company commercializing silicon oxide RRAM.

Additional information

Correspondence should be addressed to Zhong Sun.

FIGURE CAPTIONS

Fig. 1. Computing primitives and CIM basics. a, The ouroboros of computing primitives in von Neumann architecture and CIM architecture. In von Neumann computers, the route starts from the basic logic gates, delivering arithmetic operations to support algorithms such as ANNs. All these computations are executed in the processor, which communicates with the whole memory hierarchy to run complete programs. In today's CIM proposals, the computation is generally based on physical MAC operations in the memory unit, by using physical laws for multiplication and summation. The physical MAC can be easily parallelized in memory arrays to carry out vector and matrix arithmetic, which in turn lay the foundation of ANNs. The ANN concept can be used to perform logic gates, where the non-linear activation function can also be realized in hardware. **b**, CIM architecture, including banks of MATs, I/O buffer and controller. In each bank, there are peripheral circuits for global communications and local controls. The memory array is based on the random-access architecture composed of crosspoint WLs and BLs. **c**, Memory technologies, including VMs and NVMs, all of which can be accommodated in the crosspoint architecture to perform CIM. For some memory species like SRAM, the array designs contain complementary BLs. Additionally, with memory cell modifications, more complicated array designs might be adopted, such as those with dual WLs.

Fig. 2. A full spectrum of CIM technologies, along with memory candidates, computing primitives, and dominant applications in each type of CIM. The spectrum is established based on abstracting both parallel MAC (dot product) and logic gate as $Z = X \bullet Y$, where X and Y could be a scalar or a vector, Z is a scalar. All CIM technologies are categorized into six types, ranging from all-in-memory to none-in-memory, each of which has been implemented with several NVM and/or VM species, for performing logic gates or parallel MAC, which, in turn, target general-purpose or specific applications.

Fig. 3. XYZ-CIM, XZ-CIM, and Z-CIM. a, Schematic of a resistive memory device, and its RS behavior. V_R is the voltage drop across this two-terminal device. V_{set} and V_{reset} mark the threshold voltages of set and reset transitions, respectively. HCS and LCS represent the binary '1' and '0', respectively. **b**, Stateful IMP logic of XYZ-CIM. V_p and V_w is an externally applied voltage subject to $V_p < V_{set} < V_w$. **c**, SFNN of XYZ-CIM. V_X, V_Y, and V_Z are externally applied analog voltages

that dictate the weight values of this ANN model, thus defining the Boolean logic function. For NOR gate, there is $V_X=V_Y=0.5V_{set}$, $V_Z=1.1V_{set}$. For NAND gate, there is $V_X=V_Y=0.7V_{set}$, $V_Z=1.35V_{set}$. **d**, DRAM bitwise logic. Upon the simultaneous activation of three WLs, the final BL voltage turns out to be the Majority function of the initial states of the three DRAM cells, which in turn rewrite all the three cells. Due to its non-linear transfer characteristics, the latch-type SA acts as a perceptron neuron to deliver the linearly separable Majority logic. **e**, Logic operation of XZ-CIM. V_p represents logic '1' when implementing the input operand *Y*. Depending on *Y* value, a combination of voltages is applied to WL1 and BL. This circuit is for implementing XOR logic, and the table shows the definitions of the applied voltages for implementing different logic gates. **f**, Logic operation of Z-CIM. V_w is a voltage whose magnitude is larger than both V_{set} and $|V_{reset}|$. Depending on the initial conductance state of the memory cell, the application of WL and BL voltages may change the device state. The results of four input combinations give a logic function. If the initial state is '0' (LCS), the logic function is NIMP. If '1' (HCS), the logic function is CIMP.

Fig. 4. XY-CIM and X-CIM. a, Resistive NVM-based logic operation of XY-CIM. V_r is a small voltage for reading out the conductance state of the resistive memory cell. A preset reference current I_{ref} of the SA differentiates the BL currents that contributed by different combinations of the states of two input memory cells, resulting in a Boolean logic function. **b**, SRAM-based logic operation of XY-CIM. Depending on the states of the two SRAM cells, the pre-charged BL discharges to a certain level, which can be differentiated by the SA with a preset V_{ref}. The results on BL and BLB constitute the AND and NOR logics, respectively. **c**, Passive 1R NVM-based, **d**, 1T1R NVM-based, **e**, NOR Flash memory-based (also FeFET-based), and **f**, SRAM-based dot product operations of X-CIM. The current or the voltage change on the BL is the analog dot product result *Z* of the vector $\boldsymbol{x}$ stored in the memory cells and the externally applied vector $\boldsymbol{y}$. All share the same sensing method to output the digitalized *Z*, by using SA or ADC. In **c** and **d**, the vector $\boldsymbol{y}$ may be applied through WLs (blue symbols) or SLs (red symbols), which usually correspond to binary or analog input values, respectively. **g**, Multi-bit capability of various memory devices. **h**, Illustration of multiplying one $\boldsymbol{x}$ element with one $\boldsymbol{y}$ element, both of which could be single-bit (binary and bipolar) or multi-bit (analog). **i**, Illustration of converting the analog result $\boldsymbol{x}^T\boldsymbol{y}$ to the digitalized output *Z*. The dotted line represents the fully analog result without discretization, *e.g.*, by using TIA for sensing the result

on the BL. **j**, MVM circuit, by generalizing the dot product operation from vector $\boldsymbol{x}$ to matrix $\boldsymbol{X}$. **k**, Matrix inversion circuit, where vector $\boldsymbol{y}$ is represented by currents externally injected to WLs, and vector $\boldsymbol{z}$ consists of output voltages of OPAs.

Table I. Summary of features, advantages and disadvantages, and challenges of various CIM schemes. Here the logic gate is considered to a 2-input one, the parallel MAC operation is considered to be *N*-dimensional. For parallel of X-CIM, it usually targets the acceleration of neural network (NN).

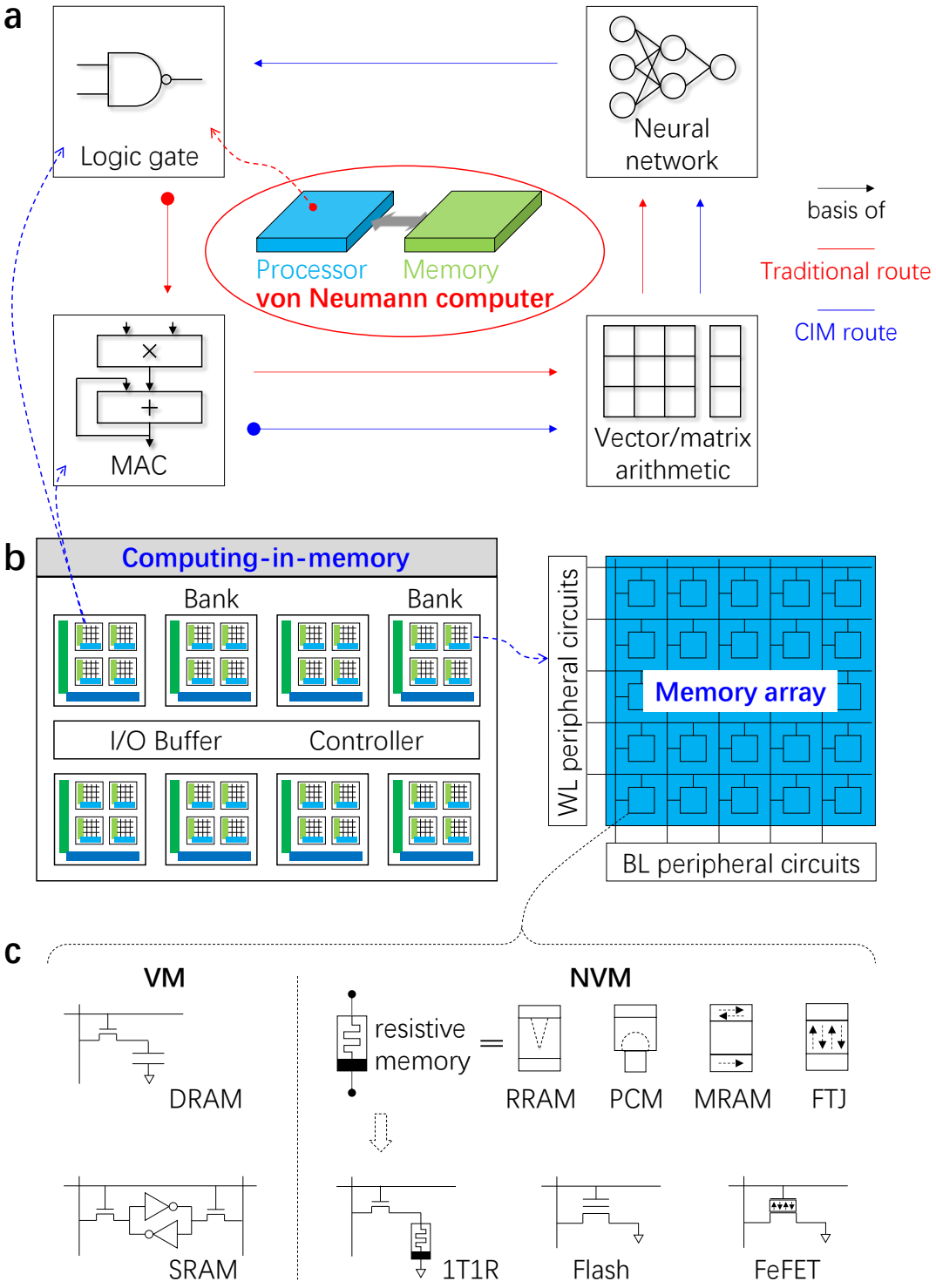
a
Logic gate
Neural network
Processor
Memory
von Neumann computer
basis of
Traditional route
CIM route
MAC
Vector/matrix arithmetic
b
Computing-in-memory
Bank
Bank
I/O Buffer
Controller
WL peripheral circuits
Memory array
BL peripheral circuits
c
VM
NVM
DRAM
resistive memory
RRAM
PCM
MRAM
FTJ
SRAM
1T1R
Flash
FeFET

Figure 1.

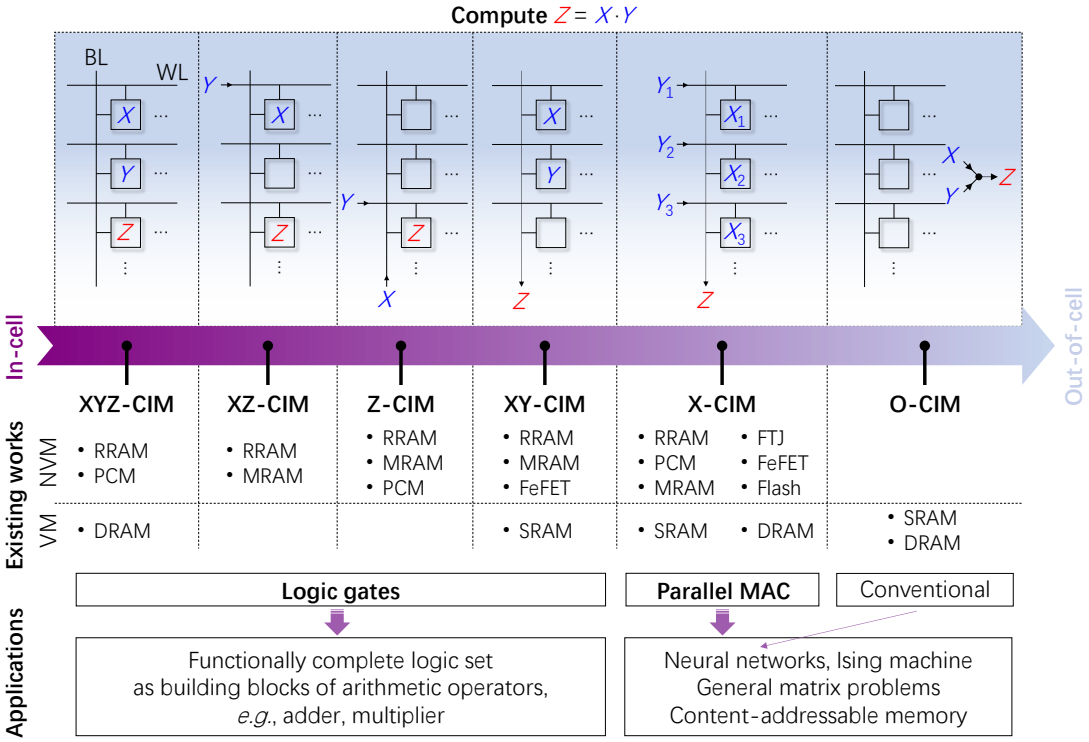
Compute $Z = X \cdot Y$
BL
WL
In-cell
Out-of-cell
XYZ-CIM
XZ-CIM
Z-CIM
XY-CIM
X-CIM
O-CIM
Existing works
NVM
VM
• RRAM
• PCM
• RRAM
• MRAM
• RRAM
• MRAM
• PCM
• RRAM
• MRAM
• FeFET
• RRAM
• PCM
• MRAM
• FTJ
• FeFET
• Flash
• DRAM
• SRAM
• SRAM
• DRAM
• SRAM
• DRAM
Applications
Logic gates
Parallel MAC
Conventional
Functionally complete logic set
as building blocks of arithmetic operators,
e.g., adder, multiplier
Neural networks, Ising machine
General matrix problems
Content-addressable memory

Figure 2.

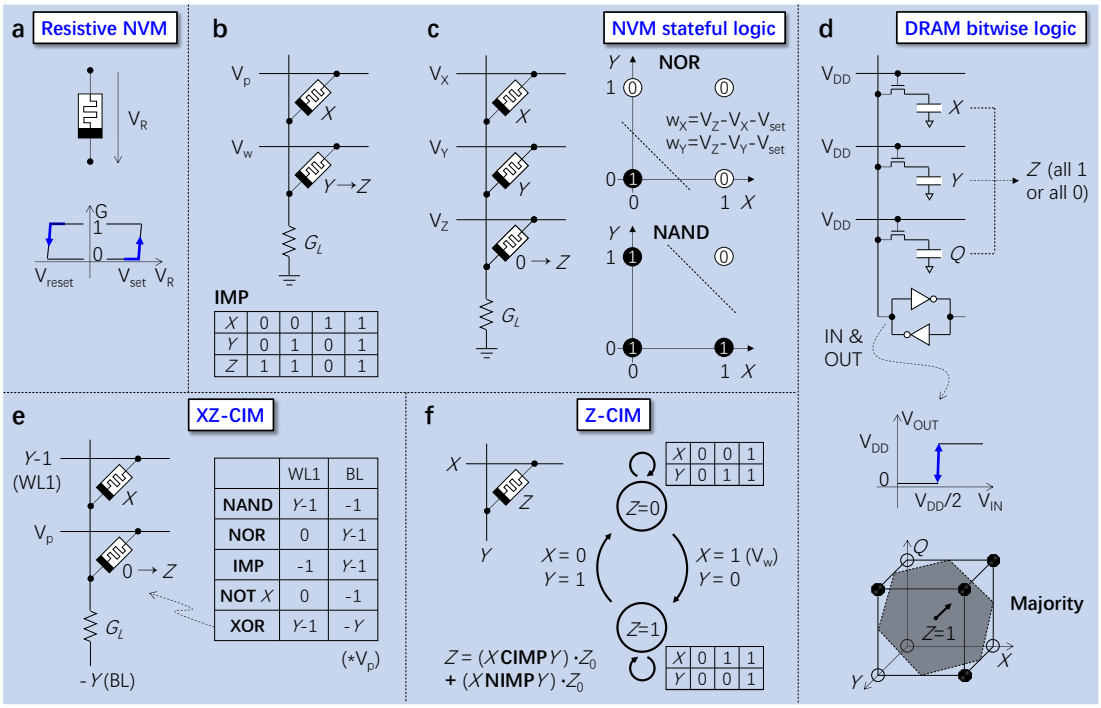
a
Resistive NVM
b
c
NVM stateful logic
d
DRAM bitwise logic
e
XZ-CIM
f
Z-CIM
IMP
NOR
NAND
Z (all 1 or all 0)
IN & OUT
Majority
WL1
BL
NAND
NOR
IMP
NOT X
XOR
(*Vp)
Z=0
Z=1
X = 0 Y = 1
X = 1 (Vw) Y = 0

Figure 3.

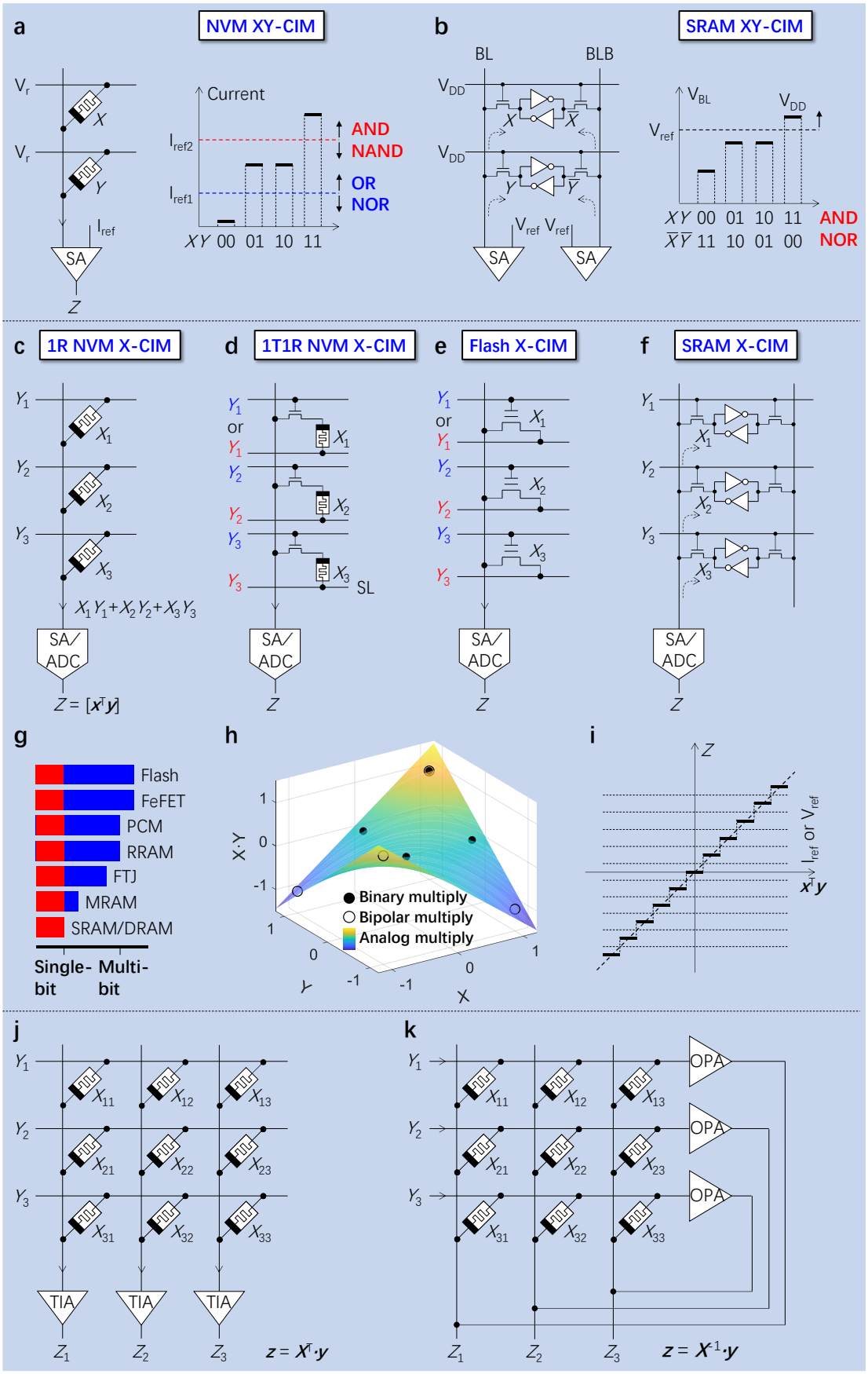
a
NVM XY-CIM
Current
AND
NAND
OR
NOR
SA
b
SRAM XY-CIM
BL
BLB
AND
NOR
c
1R NVM X-CIM
SA/ ADC
d
1T1R NVM X-CIM
SL
e
Flash X-CIM
f
SRAM X-CIM
g
Flash
FeFET
PCM
RRAM
FTJ
MRAM
SRAM/DRAM
Single-bit
Multi-bit
h
Binary multiply
Bipolar multiply
Analog multiply
i
j
TIA
k
OPA

Figure 4.

<table>
<tr><th></th><th>XYZ-CIM</th><th>XZ-CIM</th><th>Z-CIM</th><th>XY-CIM</th><th>X-CIM</th><th>O-CIM</th></tr>
<tr><td>Number of cells per operation</td><td>2 or 3</td><td>2</td><td>1</td><td>2</td><td>N (single-bit or multi-bit)</td><td>N</td></tr>
<tr><td>Additional devices</td><td>1 or 0 resistor (for NVM)
1 SA (for DRAM)</td><td>1 resistor</td><td>None</td><td>1 SA</td><td>1 ADC or SA</td><td>N multipliers +
N adders</td></tr>
<tr><td rowspan="2">Advantages</td><td colspan="4">Boolean logic – universal framework</td><td>Direct mapping of arithmetic operations – high efficiency</td><td></td></tr>
<tr><td>All-in-memory,
Free of input-output conversion</td><td>Easier to construct logic gates</td><td>Simple implementation</td><td>Free of RS,
Better system compatibility</td><td>High energy/area efficiency,
Free of input-output conversion (for NN)</td><td>Robust computation,
Higher flexibility</td></tr>
<tr><td rowspan="3">Disadvantages</td><td colspan="4">Synthesis of logic gates – inefficient</td><td>Limited to MAC acceleration</td><td></td></tr>
<tr><td colspan="3">Limited endurance of NVM</td><td>SA overhead</td><td>ADC overhead,
Rigid dataflow,
Analog non-idealities</td><td>Low energy/area efficiency</td></tr>
<tr><td></td><td colspan="2">Need to read out output as voltage for cascading</td><td>Need to write output as conductance for cascading</td><td></td><td>Conventional read & write</td></tr>
<tr><td rowspan="3">Reliability challenges</td><td colspan="3">High endurance (for NVM)</td><td colspan="2">Good retention (for NVM)</td><td rowspan="3">Mature digital technology</td></tr>
<tr><td colspan="2">Low variations of working voltage and conductance</td><td>Low working voltage variations</td><td>Low conductance variations</td><td>Low conductance variations,
Linear conductance updates</td></tr>
<tr><td colspan="5">PVT variations</td></tr>
<tr><td rowspan="2">Energy efficiency challenges</td><td colspan="3">Optimizations of working voltages and currents</td><td colspan="2">Optimization of conductance range</td><td>Cell design</td></tr>
<tr><td colspan="4">Efficient and reliable logic synthesis</td><td>DAC & ADC</td><td>Adder tree</td></tr>
</table>

Table I.