IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 October 2024, accepted 30 November 2024, date of publication 4 December 2024,
date of current version 19 December 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3511492*

TOPICAL REVIEW

# Trends and Challenges in Computing-in-Memory for Neural Network Model: A Review From Device Design to Application-Side Optimization

**KE YU[1], (Graduate Student Member, IEEE), SUNMEAN KIM[1,2], (Member, IEEE), AND JUN RIM CHOI[1,2], (Member, IEEE)**

[1]School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Republic of Korea
[2]School of Electronics Engineering, College of IT Engineering, Kyungpook National University, Daegu 41566, Republic of Korea

Corresponding author: Jun Rim Choi (jrchoi@ee.knu.ac.kr)

This work was supported in part by Samsung Electronics Company Ltd.; and in part by the BK21 FOUR Project funded by the Ministry of Education, South Korea, under Grant 4199990113966.

**ABSTRACT** Neural network models have been widely used in various fields as the main way to solve problems in the current artificial intelligence (AI) field. Efficient execution of neural network models requires devices with massively parallel Multiply-accumulate (MAC) and Matrix-vector Multiplication (MVM) computing capability. However, existing computing devices based on von Neumann architecture suffer from bottlenecks, and the separation of memory and computation module makes data on the move wasting a lot of meaningless computation time and energy. Computing-in-memory (CIM) based on performing MAC computation inside the memory is considered a promising direction to solve this problem. However, large-scale application of CIM still faces challenges due to the non-idealities of current CIM devices and the lack of a common and reliable programmable interface on the application side. In this paper, we will comprehensively analyze the current problems faced by CIMs from various perspectives, such as CIM memory arrays, peripheral circuits, and application-side design, and discuss the possible future development opportunities of CIMs.

**INDEX TERMS** Neural network model, von Neumann architecture, computing-in-memory, multiply-accumulate, matrix-vector multiplication, peripheral circuits, memory array.

## I. INTRODUCTION

The significance of various computing devices developed based on the von Neumann architecture in the contemporary human society that highly depends on Artificial Intelligence (AI) technology is evident. The emergence of the von Neumann computing paradigm shown in FIGURE 1(a) has transformed the original computer system, where programs were only controllable by hardware [1]. Von Neumann's design enabled the programmability of computers by storing programs in the form of binary codes along with data in memory. This paradigm separates hardware design from software programming, making the use of the computer more flexible [2]. As a result, this greatly advanced the development of computers led to the emergence and rapid growth of AI based on Machine Learning and Deep Learning [3]. While neural network models are the core component of AI currently used for problem solving [4].

The three essential components of AI, namely algorithm, computility, and data, are closely interconnected [5]. The large-scale implementation of AI in both production and daily life as illustrated in FIGURE 1(b) has led to more widely use of computing devices, resulting in an exponential growth of data [6]. At the same time, as Neural Network-based large models in the field of AI continue to develop, computing algorithms are becoming increasingly complex. The explosive amount of data and the increasing complexity of algorithms put forward higher requirements for computing devices based on computility [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Bo Pu.

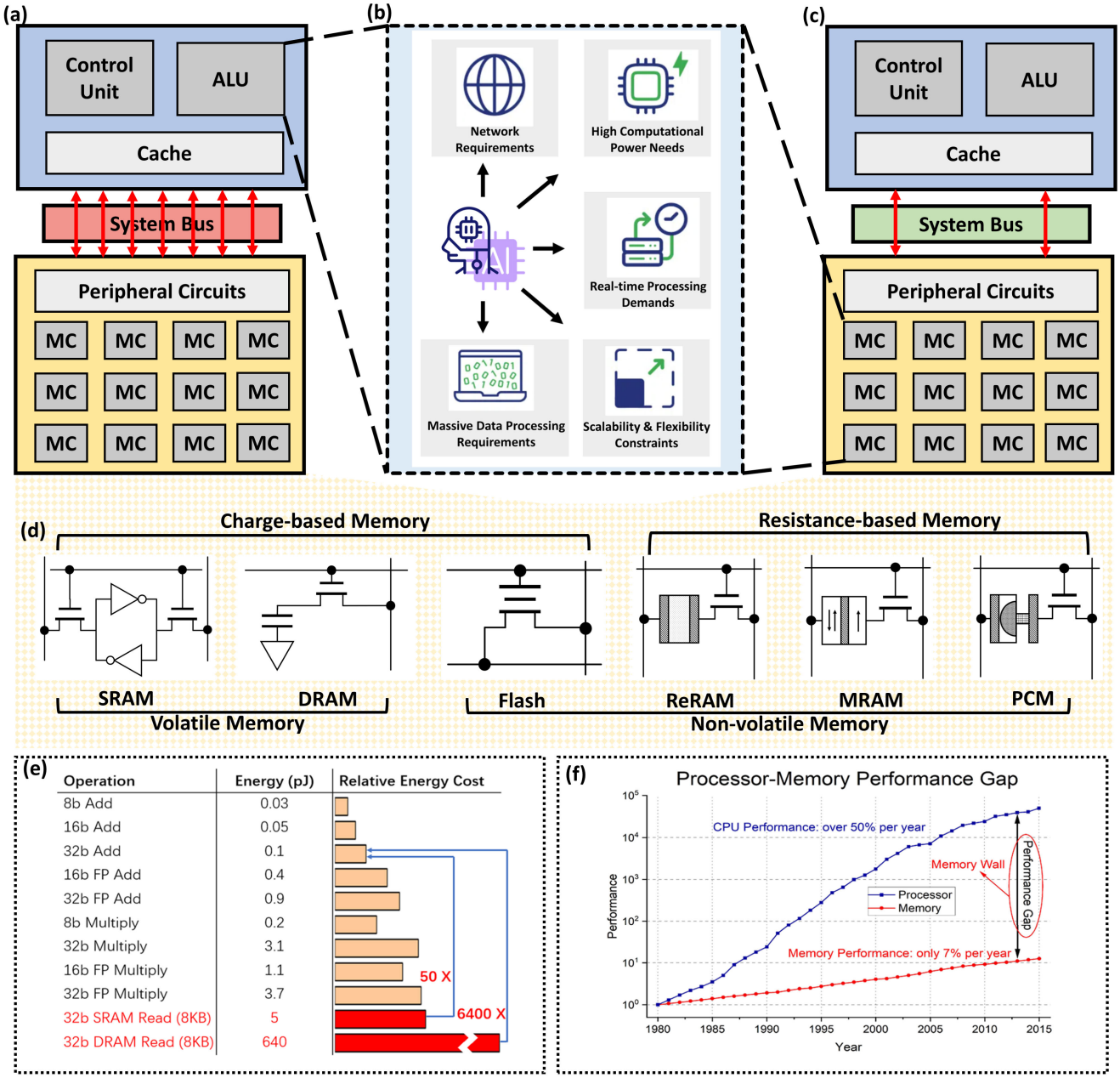


**FIGURE 1.** Background on the von Neumann computational paradigm.(a)A diagram of the von Neumann architecture. The computational load is located in the ALU in the processing module. (b)The main workload of AI based on neural network models. (c)The CIM computing paradigm. The computational load is located in the array based on memory cells (MCs) in the storage module. (d)The six common standard memory cells on the market. (e)The "buckets effect" problem with memory. (f)Comparison of the energy consumption of data read/write with that of the computing process.

However, the limitation of computing devices based on von Neumann architecture is becoming increasingly prominent in light of the growing volume of data. In the separate-storage-computing model, data must be frequently transferred between modules. Researches have indicated that the energy expended on reading and writing data far surpasses that used for computation itself which is shown in FIGURE 1(e) [8]. This is deemed unacceptable, especially considering the projected increase in data volume in the future. On the other hand, the development of memory has significantly lagged behind that of the processor due to the separation of the integrated circuit industry and the gap in manufacturing processes [9]. As FIGURE 1(f), reading and writing data is significantly slower than the computation process, creating a ''Buckets effect'' in the computing system. Although memory manufacturers are constantly working to increase the interface speed of general-purpose memory devices, the number of IOs and operating frequency to boost DRAM memory read/write speeds through TSV-based 2.5D/3D stacking have been increased [10]. And

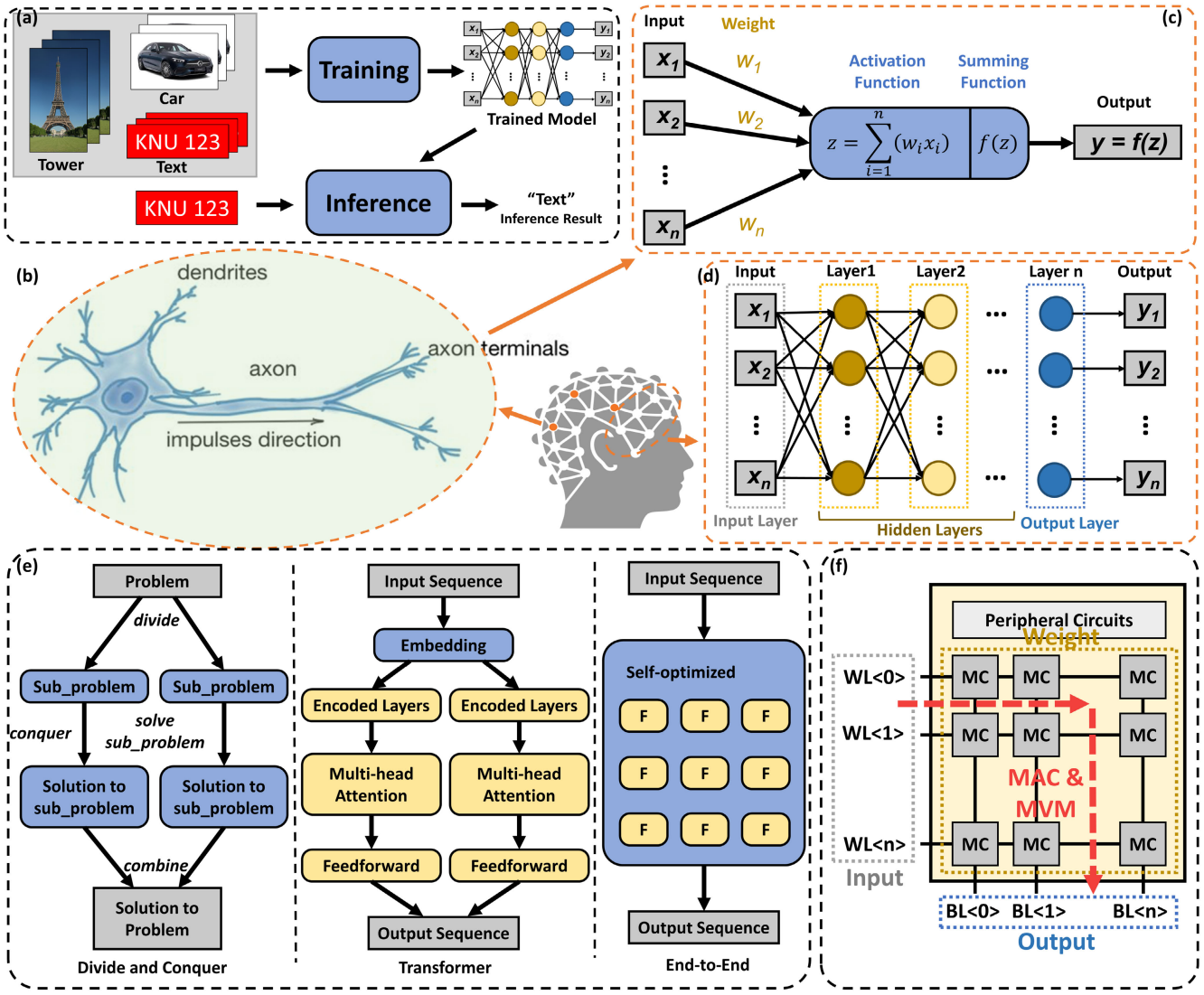


**FIGURE 2.** Introduction to the principles and computations of neural network models. (a)Demonstration of the training and inference process using neural network models. (b)The neuronal structure of the human brain used to perform computations. (c)The neural network signaling approach based on MAC computation is executed for inputs and weights. (d)Composition and signaling approach of layer-based neural network models. The neural network consists of input layers, hidden layers,and output layers. (e)Comparison of the problem solving process for "Divide and Conquer", "Transformer", and "End-to-End". (f)Accelerate MAC and MVM computations with CIM devices.

NAND Flash storage density and write speeds through 280-stacked-layer technology have been increased and optimized the data encoding method [11]. But it is still difficult to completely solve the drawbacks of the traditional computing paradigm.

In order to address the aforementioned two issues, researchers have proposed CNM (Computing-near-Memory) method based on shortening the distance between storage and computing modules and improving the memory read and write speed [12], and CIM (Computing-in-Memory) solution directly performing computing functions in the memory module. Considering that CNMs are still based on von Neumann architectures and cannot fully overcome the limitations of the existing computational paradigm, CIM as shown in FIGURE 1(c) is considered to be a more efficient solution [13]. Depending on the physical properties of conventional general-purpose memories, the possibility of performing arithmetic or logical computations is achieved by modifying their cellular structure and interconnecting their array set combinations, peripheral circuits, and control modules in series [14]. Currently there is extensive research on performing CIM for six common types of memories as shown in TABLE 1 and FIGURE 1(d) composed by SRAM (Static Random Access Memory), DRAM (Dynamic Random Access Memory), Flash, ReRAM (Resistive Random Access Memory), MRAM (Magnetic Random Access Memory), and PCM (Phase-change Memory) [15], and verifying that they can perform Machine Learning (ML) and

Deep Learning (DL) based computations within a certain degree [16].

While the emergence of CIM offers a solution for efficient computation, current progress in optimizing CIM-based circuits and systems still struggles to meet the computational requirements of Large Models, and faces multiple challenges in the design and practical applications. The remainder of this paper is as follows: we analyze the requirements of computing devices for large model systems in current AI applications in Section II, and review the design of CIM memory cell array, and peripheral circuits optimization methods in Sections III and IV, respectively. In Section V, we discuss the related work with the application of CIM computational paradigm to neural network model. In Section VI, we summarize some of the current challenges facing the CIM computational approach and analyze possible future responses and directions. Section VII concludes the entire paper.

## II. REQUIREMENTS OF NEURAL NETWORK MODEL FOR COMPUTATION DEVICES

Training and inference as shown in FIGURE 2(a) are the main operation modes of large AI models based on ML and DL [17]. The training process is performed by feeding a series of labeled data into the algorithmic model in the cloud for computation, and continuously adjusting and optimizing the algorithmic parameters until the algorithmic recognition accuracy reaches a high level. The inference process utilizes the trained model to reason out various conclusions using new data, i.e., with the help of the existing Neural Network model for computation, new input data is used to obtain the correct conclusion at one time. The training and inference process is highly dependent on ANN (Artificial Neural Network) models built to mimic the human brain architecture [18] as FIGURE 2(b) and (d), including DNN (Deep Neural Network) [19], CNN (Convolutional Neural Network) [20] and RNN (Recurrent Neural Network) [21].

Unlike early AI models based on "Divide and Conquer" method, which solved complex problems by splitting them into multiple simple problems, the current more advanced ANN-based models rely more on "Transformer"-based solution ideas [22], and have a tendency to shift towards the "End-to-End" paradigm [23]. FIGURE 2(e) compares the problem solving process of the three. Application side developers prioritize the input and output data of the model over the computational processes involved. This complicates the adaptation to the von Neumann architecture, which is inherently more suited for well-defined computational processes [24].

As FIGURE 2(c) and (d), in artificial neural networks (ANNs), neurons in one layer are sequentially interconnected to a set of neurons in the subsequent layer through weights, which are represented by a matrix [18]. The output of the neurons in the subsequent layer is driven by the activation values of the neurons in the preceding layer and the Synaptic Weight that connect these neurons to those in the next layer. A substantial number of Matrix-vector Multiplication (MVM) computions as

$$f(y) = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} \quad (1)$$

are necessary during operations, whether in Forward/Back propagation, Feature Transformation, or Parameter Updating [25]. Researches have also demonstrated that MVM operations consisting of Multiply-accumulates (MACs) as

$$y = f(z) = \sum_{i=1}^{n} (x_i \times w_i) \quad (2)$$

constitute the predominant component of most neural network models [7].

Desiring to execute MVM operations efficiently imposes significant demands on hardware for large-scale MAC operations. Traditional processors, which rely on complex logic structures, often struggle to perform the necessary processing effectively. Although FPGAs (Field Programmable Gate Array) and GP-GPUs (General-purpose Graphics Processing Units) hardware, which possess enhanced parallel computing capabilities, can partially mitigate this issue, the update rate of hardware remains inadequate to keep pace with the rapid development of large models [26]. Therefore, research focused on the CIM computing paradigm, which aims to address the power consumption and bottleneck issues inherent in von Neumann architectures, has garnered increasing interest. By mapping neural network models onto cross-arrays of storage devices and executing transfer-based computational operations directly within these devices as indicated in FIGURE 2(f), this approach may provide an effective solution to the challenges posed by current computing architectures [16].

## III. CIM ARRAY DESIGN

The CIM array consisting of memory cells that exists on chip in the form of computing macros, serves as the core element of devices that stores data and performs computations. CIM computing is realized by optimizing the computable architecture and the logical way of reading and writing data on the basis of the traditional memory cell and array. In a generic CIM device, the computed weight data of the neural network model is stored in a CIM cell-based array, while the activation data is written to the CIM array in the mode of the input signal. After the CIM array completes the MAC computation, the result of the computation is read out in the form of an output signal. Currently, researchers have extensively investigated the implementation of computing functions on all six common types of memories available in the market in TABLE 1, and they are summarized in TABLE 2.

**TABLE 1. Device characteristics of mainstream (SRAM, DRAM, and Flash) and emerging (ReRAM, MRAM, and PCM) memory technologies on the market [15].**

| | SRAM | DRAM | Flash | | ReRAM | MRAM | PCM |
|---|---|---|---|---|---|---|---|
| | | | NOR | NAND | | | |
| Storage Method | Charge-based | Charge-based | Charge-based | Charge-based | Resistance-based | Resistance-based | Resistance-based |
| Cell Area | $>100F^2$ | $6F^2$ | $10F^2$ | $<4F^2$ (3D) | 4-12$F^2$ | 6-50$F^2$ | 4-30$F^2$ |
| Multibit | 1 | 1 | 2 | 3 | 2 | 1 | 2 |
| Voltage | <1V | <1V | >10V | >10V | <3V | <1.5V | <3V |
| Read Time | ∼1ns | ∼10ns | ∼50ns | ∼10us | ∼10ns | ∼10ns | ∼10ns |
| Write Time | ∼1ns | ∼10ns | 10us-1ms | 100us-1ms | ∼10ns | ∼10ns | ∼50ns |
| Retention | N/A | ∼64ms | >10y | >10y | >10y | >10y | >10y |
| Endurance | $>1E16$ | $>1E16$ | $>1E5$ | $>1E4$ | $>1E6-1E12$ | $>1E15$ | $>1E9$ |
| Write Energy (J/bit) | ∼1fJ | ∼10fJ | ∼100pJ | ∼10fJ | ∼0.1 pJ | ∼0.1pJ | ∼10pJ |

According to the characteristics of data volatility after memory is powered off, it can be categorized into VM-CIM (Volatile Memory CIM) based on SRAM and DRAM, and NVM-CIM (Non-volatile Memory CIM) based on FLASH, ReRAM, MRAM, and PCM.

### A. VM-CIM

SRAM and DRAM, which are based on the charging operation principle and hierarchical arrangement architecture, are the most commonly used memories in computers based on von Neumann paradigm. Moreover, SRAM-CIM and DRAM-CIM have garnered significant attention due to their capability to utilize the current mainstream chip manufacturing technology based on CMOS and they don't have to worry about read/write lifetimes. However, due to the volatile characteristics of the memory, SRAM-CIM and DRAM-CIM is not suitable for applications that require long-term data storage.

#### 1) SRAM-CIM

As FIGURE 2(a), 6T (6 Transistors)-based standard SRAM stores data through two inverters that form an interlocked architecture and are mainly used as cache module in current computers. Due to the advantages of fast read/write and matching the state-of-the-art CMOS process, SRAM-based computing has always been the mainstream of CIM research. The SRAM-CIM implements the computational logic by adding extra transistors to the standard SRAM, and the current typical work of SRAM cell contains 6 to 18 transistors (6-18T). Categorized according to the computation method, there are two ways to implement SRAM-CIM, based on digital computation and based on analog computation.

##### a: DIGITAL COMPUTATION

SRAM-CIM based on digital computation has higher computational accuracy and throughput. As indicated in FIGURE 3(b), multiplication computation is realized by using a set of standard 6T-SRAM modules designed with XNOR-based structures [27] or 4T bit-serial multiplier [28]. Use the structure of the adder tree to add up the values that have been multiplied, thus realizing the MAC computation function. However the additional transistors that need to be deployed for the multiplication and adder structures can significantly increase the area overhead. A separated word line (WL) 6T-SRAM cell for storing weights and performing CIM computations is also proposed in [29]. This design enables to perform only AND computation when the weight is 0 during the read period, saving unnecessary multiplication computation thus reducing the operations. However, SRAM-CIM based on digital computing still has a significant disadvantage in terms of storage density in general, so this approach is more suitable for computing with low data volume and complexity but high speed requirements.

##### b: ANALOG COMPUTATION

To maximize computational density without incurring area overhead, research focused on analog computation has increasingly become the predominant approach in SRAM-CIM over the past two years. By representing the activation input as an analog signal, the computation is done while reading the weights in the memory. Analog computation base on Mixed-signal also offers greater advantages in power consumption compared to digital computation. There are three primary approaches to SRAM-CIM that utilize analog computation within the Voltage-domain, Current-domain, Charge-domain, and Time-domain.

###### I) VOLTAGE-DOMAIN

The Voltage-domain based SRAM-CIM converts the input data into linear analog voltage signals via Digital-to-Analog Converter (DAC). Multiple WLs in the array of SRAM-CIM based on 12T [30] or "6T SRAM + 6T local computing" [31] are activated simultaneously based on the input eigenvalues, and the bit lines (BLs) are charged/discharged by the analog voltage signals to make the weights stored in the SRAM-CIM cells. The multiplication results are combined in BLs. Finally, the voltage results on the BLs are accumulated to obtain the output of the MAC operation and reconverted to a digital signal through Analog to Digital Converter (ADC) model as shown in FIGURE 3(c). However, the voltage-domain based approach imposes stringent requirements on the design of the reference voltage. Additionally, there exists an inherent bottleneck in that the operating voltage of the circuit itself has an upper limit and is difficult to be

**TABLE 2.** Summary of CIM array design for common memories.

| References | Memory | Volatility | Storage Method | Technology | Cell Architecture | Memory Size | Area Overhead | MAC Method | Energy Efficiency |
|---|---|---|---|---|---|---|---|---|---|
| JSSC'2021 [27] | SRAM | Volatile | Charge-based | 65nm | 6T | 16Kb | 0.32mm×0.71mm | Digital | 2.06(TOPS/W) 117.3(TOPS/W) |
| ISSCC'2021 [28] | | | | 22nm | 6T | 64Kb | 0.202mm$^2$ | | 24.7(TOPS/W) 89(TOPS/W) |
| ESSCIRC'2023 [29] | | | | 28nm | 6T | 256Kb | 30(Kb/mm$^2$) | | 30.3(TOPS/W) 121(TOPS/W) |
| JSSC'2020 [30] | | | | 65nm | 12T | 16Kb | 450um×250um | Analog (Voltage-domain) | 403 (TOPS/W) |
| ISSCC'2020 [31] | | | | 28nm | 6T | 64Kb | 5.795um×0.71um | | 23.26(TOPS/W) |
| JSSC'2022 [32] | | | | 65nm | 8T | 16Kb | 3.24um×3.24um | Analog (Current-domain) | 15.8-490 (TOPS/W) |
| ISSCC'2023 [33] | | | | 28nm | 6T | 9Kb | 0.012um$^2$ | | 14.24-33.44 (TOPS/W) |
| JSSC'2024 [34] | | | | 65nm | 6T | 40.5Kb | 559(Kb/mm$^2$) | Analog (Charge-domain) | 18.6-40.2 (TOPS/W) |
| JSSC'2024 [35] | | | | 28nm | 9T-1C | 16Kb | 0.504(Mb/mm$^2$) | | 163(TOPS/W) |
| JSSC'2020 [36] | | | | 65nm | 8T-1C | 2Kb | 270um×300um | | 671.5(TOPS/W) |
| JSSC'2023 [37] | | | | 28nm | 10T-1C | 3,456Kb | 20.9mm$^2$ | | 437(TOPS/W) |
| JSSC'2022 [38] | | | | 16nm | 8T-1C | 4.5Mb | 25mm$^2$ | | (TOPS/W) |
| TCAS-II'2024 [39] | | | | 180nm | 9T | 8Kb | 0.28mm$^2$ | Analog (Time-domain) | 768.7-2124.2 (TOPS/W) |
| TCAS-I'2021 [40] | | | | 40nm | 12T | 8Kb | 250um×350um | | 537(TOPS/W) |
| JSSC'2024 [41] | | | | 28nm | 6T | 1Mb | 8.55um×1.42um | | 22.02–115.6 (TOPS/W) |
| TCAS-I'2020 [42] | DRAM | | | 65nm | 1T-1C | 524.2Kb | N/A | Digital | N/A |
| MICRO'2017 [43] | | | | N/A | 1T-1C | N/A | N/A | | N/A |
| HPCA'2020 [44] | | | | N/A | 1T-1C | N/A | N/A | | N/A |
| JSSC'2024 [45] | | | | 65nm | 1T-1C | 16Kb | 0.57mm$^2$ | Analog (Charge-domain) | 4.76(TOPS/W) |
| ISCAS'2023 [46] | | | | 28nm | 1T-1C | 19Mb | 2.05(Mb/mm$^2$) | | 382-1531.3 (TOPS/W) |
| ISSCC'2021 [50] | | | | 65nm | 3T-0C | 33Kb | 1.5mm×2.2mm | | 44(TOPS/W) |
| ISSCC'2023 [51] | | | | 28nm | 3T-2C | 9600Kb | 0.95(Mb/mm$^2$) | | 7.8-56.0 (TOPS/W) |
| ICEIC'2024 [47] | | | | 28nm | 2T-1C | 16Bit | 400um×200um | Analog (Current-domain) | 233(GOPS/W) |
| VLSIC'2024 [48] | | | | 16nm | 3T-0C | 2Kb | 0.92um$^2$ | | 56.9(TOPS/W) |
| AICAS'2023 [49] | | | | N/A | 2T-0C | 16Kb | N/A | | 11.26(GOPS) |
| TCAS-I'2021 [52] | | | | 65nm | 4T-2C | 16Kb | 1.345um$^2$ | | 17.8-552.5 (TOPS/W) |
| MWSCAS'2023 [53] | Flash-NAND | Non-volatile | | N/A | N/A | N/A | N/A | Analog (Current-domain) | N/A |
| TVLSI'2019 [54] | | | | N/A | N/A | 1Kb | N/A | | N/A |
| TCAS-I'2024 [55] | | | | N/A | N/A | 16Kb | N/A | | N/A |
| TCAS-I'2024 [56] | | | | N/A | N/A | 996 SSL × 10000 BL | N/A | | N/A |
| IEDM'2021 [57] | Flash-NOR | | | 55nm | N/A | 32Bit FP | N/A | Analog (Current-domain) | N/A |
| ISCAS'2019 [58] | | | | 65nm | N/A | 4.3Mb | 0.048mm$^2$ | | $1.97\times10^{-8}$ Energy(J)/Image |
| VLSIC'2020 [59] | | | | N/A | Vertical 2T | N/A | N/A | | N/A |
| IEDM'2022 [60] | | | | N/A | N/A | 2.7Gb | 25mm$^2$ | | N/A |

**TABLE 2.** *(Continued.)* **Summary of CIM array design for common memories.**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ISSCC'2019 [61] | ReRAM | Non-volatile | Resistance-based | 55nm | 1T-1R | 1Mb | 0.2025um$^2$ | Analog (Voltage-domain) | 21.9-53.17 (TOPS/W) |
| ISSCC'2020 [62] | | | | 22nm | 1T-1R | 2Mb | 2mm×3mm | Analog (Current-domain) | 28.93-121.38 (TOPS/W) |
| ISSCC'2021 [63] | | | | 40nm | 1T-1R | 64Kb | 0.437mm$^2$ | | 56.67TOPS/W |
| ISSCC'2022 [66] | | | | 14nm | 1T-1R | 1Mb | 13.4(Mb/mm$^2$) | | N/A |
| ISSCC'2021 [67] | | | | 22nm | 1T-1R | 8MB | 6mm×3mm | Analog (Time-domain) | 21.6-1286.4 (TOPS/W) |
| ISSCC'2022 [68] | MRAM | | | 28nm | 2T-2R | 2Mb | 4.5mm$^2$ | Digital | 22.4(TOPS/W) 41.5(TOPS/W) |
| ISSCC'2023 [65] | PCM | | | 40nm | 1T-1R | 2Mb | 18mm$^2$ | Analog (Current-domain) | 20.5-718 (TOPS/W) |

subdivided many times. This will limit the design of the type of input data. Since read/write operations share the BL with computation, computation may also encounter SRAM interference problems [31].

### II) CURRENT-DOMAIN

The Current-domain based SRAM-CIM shown in FIGURE 3(d) addresses certain limitations of Voltage-domain based SRAM-CIM to a degree. It operates similarly to Voltage-domain based SRAM-CIM, but the multiplication result is expressed by the discharge current of the SRAM-CIM cell. The addition result is then based on Kirchhoff's current law as in

$$I_{output} = I_1 + I_2 + \ldots + I_i, \tag{3}$$

which is determined by the sum of the currents in the parallel circuit. So it will encounter the same issue of interference with the weights stored in SRAM during computation as Voltage-domain based SRAM-CIM. Reference [32] designed an 8T SRAM-CIM cell where two additional access transistors have been designed in addition to the standard SRAM. This modification creates dedicated channels for read operations, effectively mitigating issues related to read/write interference. Related work reduces the nonlinearity and computational inconsistency problems of Mixed-signal, but adding additional BLs significantly increases the area and power overhead. The work presented in [33] still utilizes the 6T SRAM architecture, but a computation-current-tracked current-to-digital convertors (CCT-CDCs) has been designed to minimize the interference caused by Current-domain computational inconsistencies, and maximize the balance between computational accuracy and area overhead. The Current-domain based SRAM-CIM is also susceptible to PVT (Process, Voltage, and Temperature) [34], And it is difficult to adapt to large-scale computation. When the MAC computation is large, it will significantly increase the current on the BLs and change the drain-source voltage, resulting in an increase in nonlinearity [35].

### III) CHARGE-DOMAIN

In order to effectively deal with the PVT effects of Voltage/Current-domain based SRAM-CIM, [34], [35] proposes a design of Charge-domain based SRAM-CIM. The non-ideality of the MAC computation results is reduced by adding a capacitor in the SRAM cell as indicated in FIGURE 3(e) that is more robust to voltage and temperature variations for performing multiplication operations in the SRAM-CIM cell. Charge-domain based SRAM-CIM cells utilize transistors (2T [36], 3T [35], or 4T [37]) based switching to control the multiplication of the activation data with the weights in SRAM and the result of the multiplication is stored as charge in the capacitor. The accumulation computation is accomplished by reading the BL output voltage value using the ADC. References [35] and [36] adopt a charge coupling scheme in their computational approach, while [38] adopt a charge sharing scheme. The charge-domain based scheme, despite having better computational ideality, is challenging in terms of area overhead because of the use of capacitors and the design of logic switches. Reference [34] in their work tried using a shared capacitor for the SRAMs on the BL to significantly reduce the area burden of the charge domain design due to the capacitor structure.

### IV) TIME-DOMAIN

The Time-domain based SRAM-CIM as shown in FIGURE 3(f) implements MAC computation by transforming the computation process into a time-based path delay. In contrast to the space-based computation of Voltage/Current/Charge-domain, Time-domain circumvents intricate ADC designs by outputting signals through time-to-digital converter (TDC) [39], thus consequently reduces both digital output power consumption and area overhead. Reference [40] proposes a scheme to perform weight and activation value MAC calculations based on the CIM cell of a 6T SRAM+6T time-domain (TD) module. The design of the weights based on 2b ($\pm$1) helps to delay a more accurate division. The result of multiplication computation is judged based on the delay time of the voltage signal of a single CIM cell, and the result of MAC is judged based on the total delay time on BL. Reference [39] employs a scheme of designing a dynamic 9T TD CIM for the same row of 6T SRAMs, thereby significantly reducing the area overhead. A software-hardware co-design scheme is proposed in [41],

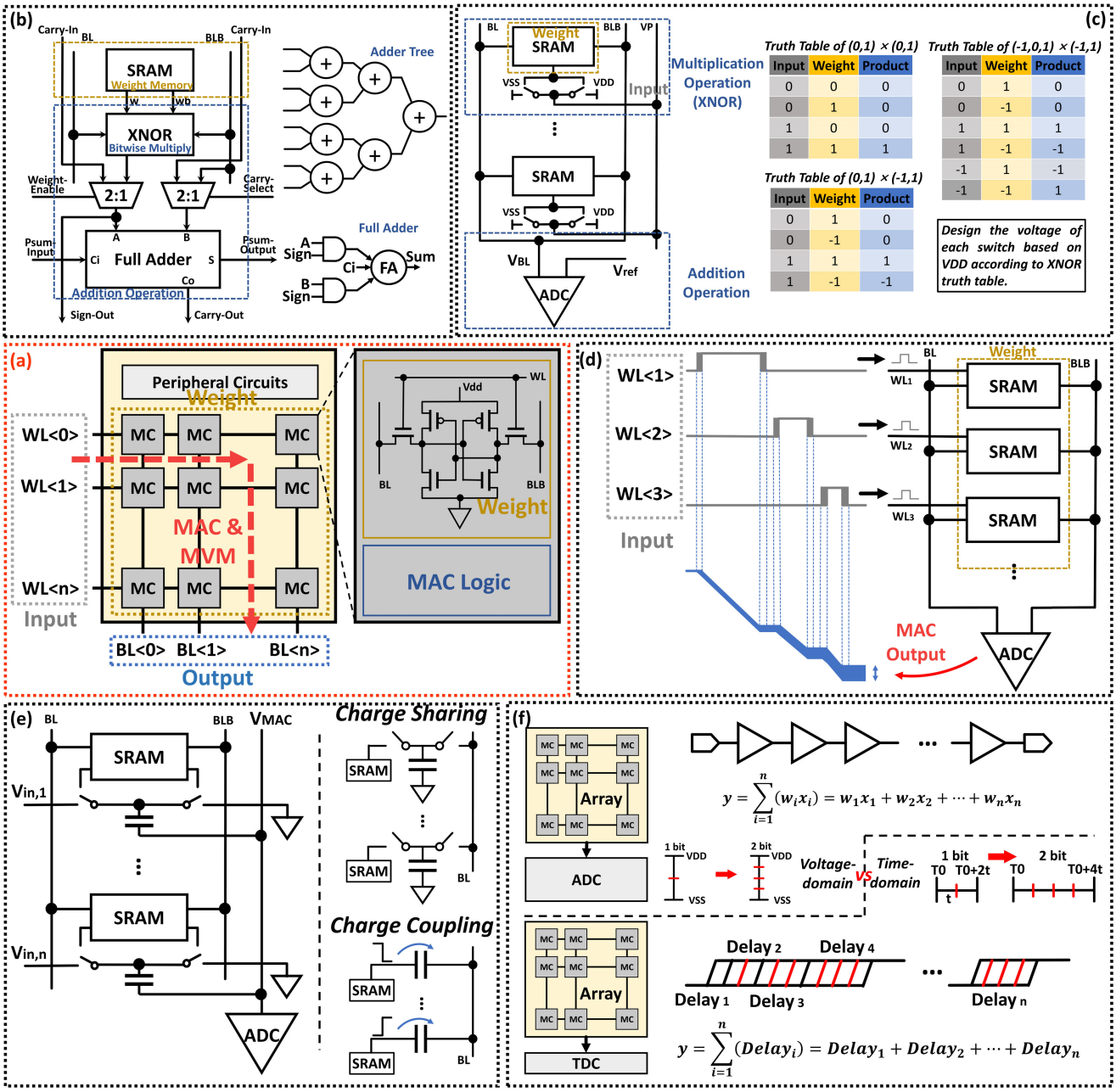


**FIGURE 3.** Review of SRAM-CIM array design approaches. (a)Combination of 6T SRAM cells and MAC computational logic to realize CIM. (b)Design of SRAM-CIM arrays using digital computation [27], [28]. (c)Voltage-domain based SRAM-CIM array implementation [30]. (d)Current-domain based SRAM-CIM array implementation. (e)Charge-domain based SRAM-CIM array implementation. (f)Time-domain based SRAM-CIM array implementation and comparison with more conventional volt-domain based SRAM-CIM implementations.

in which a single CIM structure controls a 6T SRAM in a 16-row x 2-column array, maximizing the computational density and reducing the risk of data leakage when the SRAM is in simple storage mode. At the same time, the multiplication computation with weight 0 is discarded to maximize the hardware computational efficiency. From [39], [40], and [41], it can be found that the Time-domain based design struggles to execute complex multiplicative computations, but is very efficient in dealing with addition-concentrated computations. And the design of the TDC module it relies on also faces a strong test in terms of area and power consumption when the amount of data and computation is large.

### 2) DRAM-CIM

DRAM is the main memory in current mainstream computers and is an important device for processors to temporarily store data off-chip during computation. 1T-1C cell architecture-based DRAM utilizes the amount of charge stored within the capacitance of the cell to represent the value of a binary bit, and this architecture allows for higher

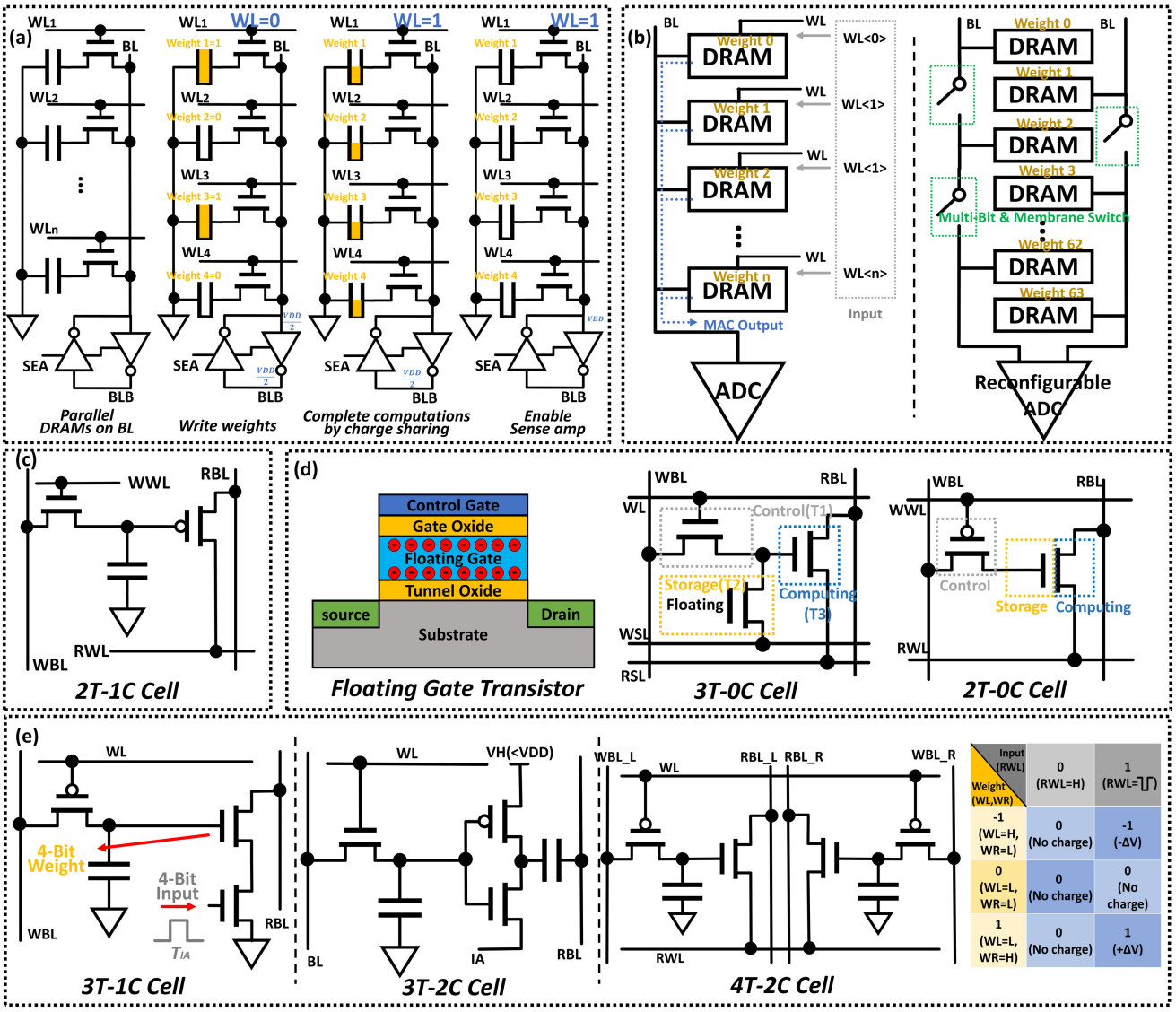


**FIGURE 4.** Review of DRAM-CIM array design approaches. (a)DRAM-CIM design based on Boolean operations for AND computation [42]. (b)The design of DRAM-CIM arrays using the 1T-1C structure [45], [46]. (c)The 2T-1C based DRAM-CIM design approach is used to prevent the weights from being corrupted [47]. (d)Design solutions using floating gate transistors instead of capacitors in DRAM-CIM [48], [49]. (e)3T-1C, 3T-2C, and 4T-2C of DRAM-CIM design solutions are used to implement more complex functions [50], [51], [52].

storage density than SRAM. This is very much in its favor for performing calculations for data-intensive applications, especially since DRAM also has relatively fast read and write speeds, despite its need for periodic charge-based refreshes.

### *a: BOOLEAN OPERATION*

An DRAM-CIM work based on Boolean operations to realize AND computation is proposed in [42] as shown in FIGURE 4(a). This design utilizes the principle of charge sharing when different capacitors are connected in parallel to activate the WL of multiple DRAMs of the same BL. The voltage across the BL is read out using a sense amplifier (SA) to compute the AND result for multiple DRAM cells. NOT operations can also be performed by adding an access transistor to the cell of the SA inverting output [43]. Reference [44] have designed an implementation of OR computation using pseudo precharged states and rewrite operations based on similar principles.

### *b: ANALOG COMPUTATION*

Similar with SRAM, DRAM can also be used to improve the efficiency of MAC operations by using analog-based computation. A 1T-1C DRAM-CIM architecture is proposed in [45]. The activation input is converted into a corresponding analog voltage signal by a DAC. As FIGURE 4(b), after writing this signal to the CIM cell storing the weights, the voltage value on BL is read out to complete the MAC computation by utilizing the principle of charge sharing of the

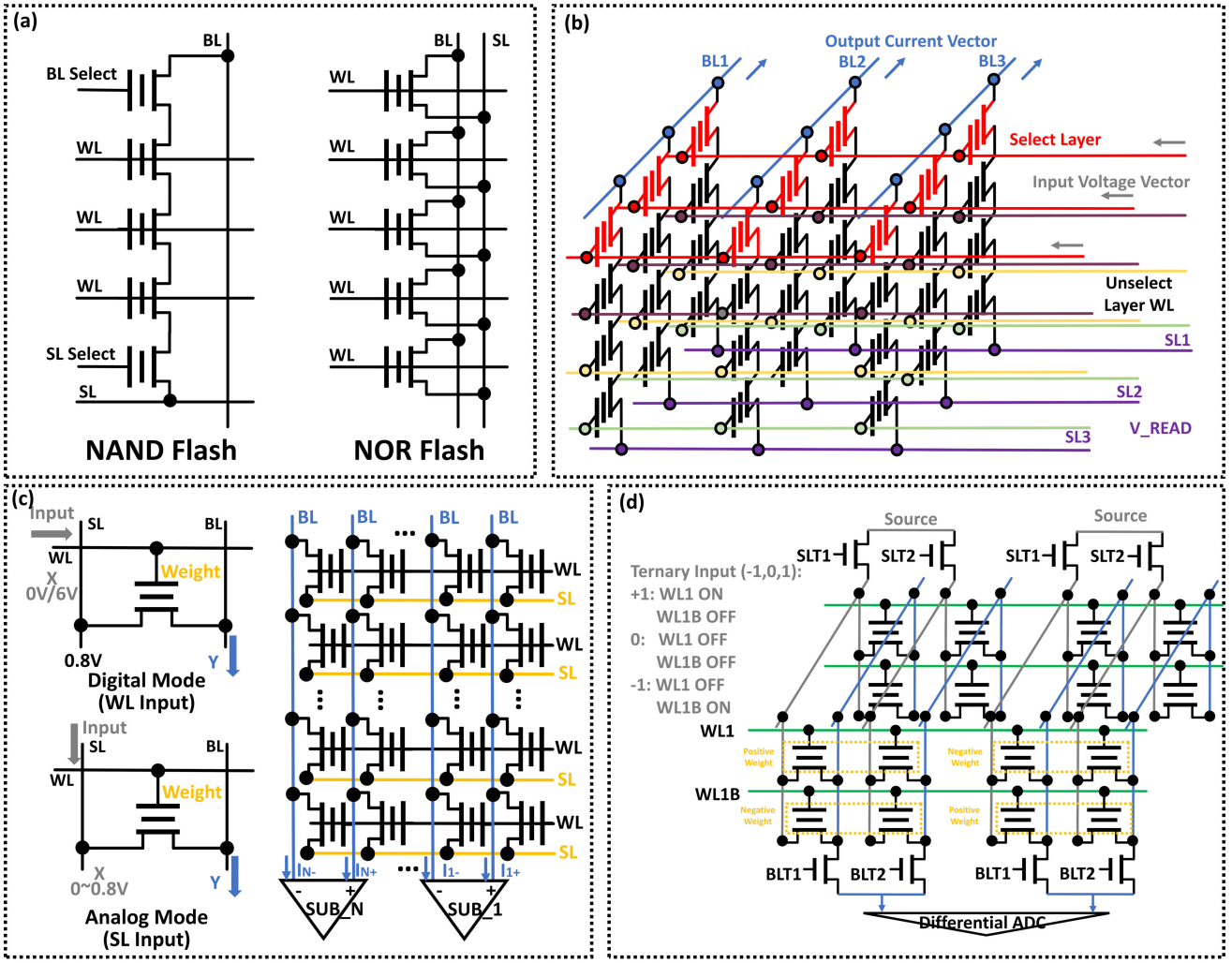


**FIGURE 5.** Review of Flash-CIM array design approaches. (a)The two common types of Flash memory: NAND Flash and NOR Flash. (b)Array design approach for CIM implementation using NAND Flash memory [54]. (c) and (d)Array design approach for CIM implementation using NOR Flash memory [58], [60].

parallel capacitor on BL. The design in [46] presents a similar computational model, but the generation of the activation inputs is done in a way that enables charge sharing of the CIM cell capacitors by designing switches on the BLs. The design can significantly reduce area overhead by avoiding the use of DAC modules.

The 1T-1C based scheme maximizes the DRAM-CIM cell density, but the design will cause the weight data stored in the cell to be corrupted during CIM operation due to charge sharing. Although it is possible to recover the relevant data through SA operations, frequent execution of SA operations results in significant additional energy consumption. A reliable solution is to design additional transistors for the DRAM-CIM cell that are separated from the control of the BLs and WLs. Reference [47] proposed a scheme for the 2T-1C architecture as indicated in FIGURE 4(c) so that the DRAM-CIM cell has four ports connected to two BLs and two WLs respectively, enabling it to perform independent read and write operations. This work uses a Current-domain based computational scheme with a MAC computation principle similar to the Current-domain one in SRAM-CIM.

Since the gate floating of the transistor, it can be regarded as a very small natural capacitor. Based on this principle, [48] proposes a 3T-0C DRAM-CIM cell design in which the capacitor is replaced by a transistor as shown in FIGURE 4(d), which alleviates the difficulties caused by the differences in the processes present in the two devices. An attempt was also made by [49] to implement the capacitor and a BL switch by a single one transistor to further optimize the density and retention time of the 3T-0C scheme by means of a 2T-0C structure.

Solutions based on 3T-1C [50], 3T-2C [51] and 4T-2C as shown in FIGURE 4(e), on the other hand, mainly seek to improve the calculation accuracy or realize more operational functions. In [50], the storage density is increased

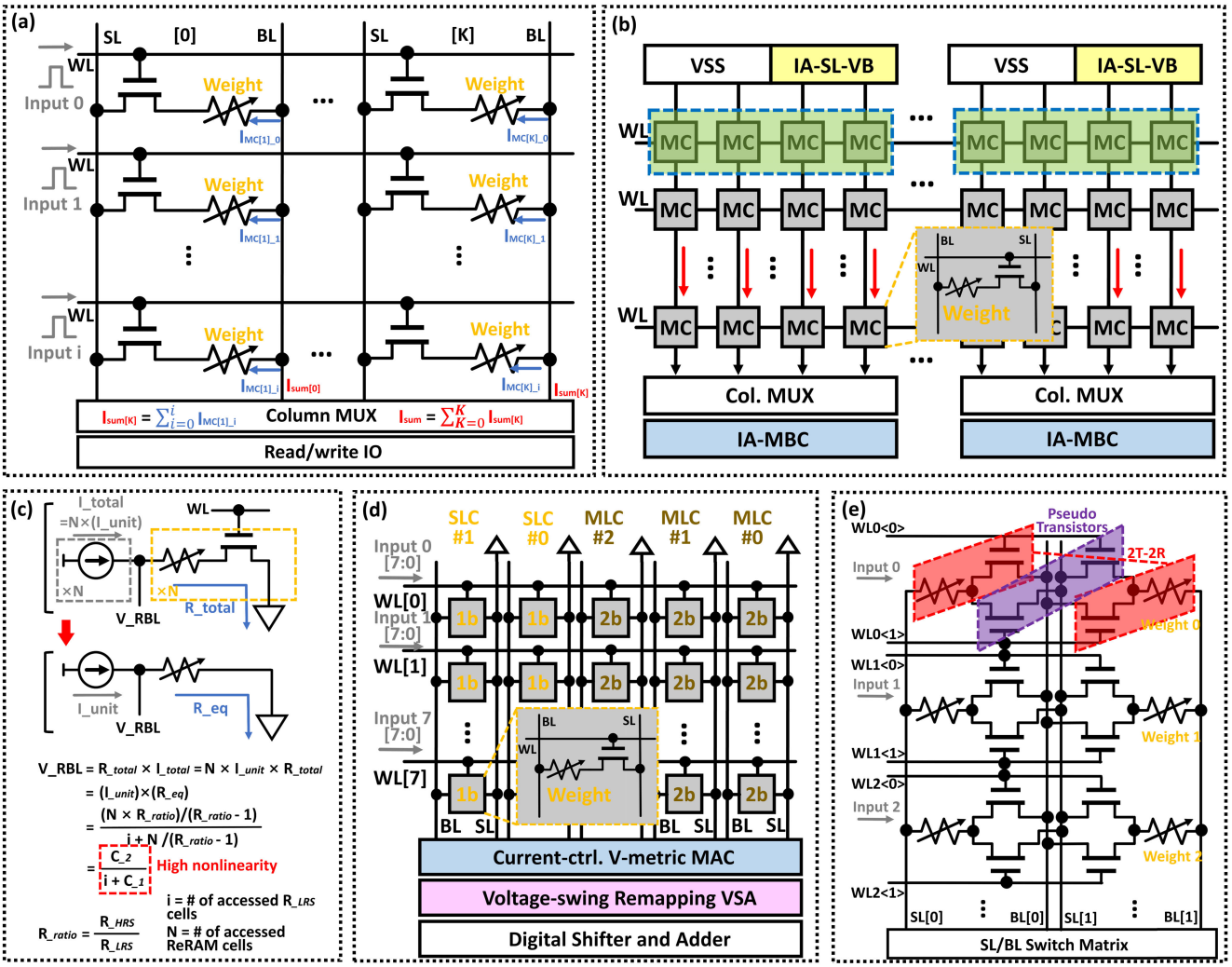


**FIGURE 6.** Review of ReRAM-CIM, MRAM-CIM, and PCM-CIM array design approaches. (a)ReRAM-CIM design based on 1T-1R architecture [61]. (b)A solution to increase read/write efficiency by connecting one single WL to 4-cell BLs [62]. (c)A ReRAM-CIM scheme using voltage-domain based operation to maintain stable sampling margins [63]. (d)Hybrid single-level cell and multi-level cell architecture design based on 1T-1R PCM-CIM [65]. (e)A CIM scheme based on digital Boolean operations using 2T-2R MRAM [68].

by by storing 4-bit weights in a DRAM-CIM cell. The 2Ts connected to the read BL, one of which operates the 4b multi-bit storage weights by varying the analog voltage on the storage node, and the other operates the 4b multi-bit activation inputs by controlling the pulse durations, enable the execution of multi-bit MAC computations in a single cell. The CIM cell based on the 3T-2C design in [51] realizes the dynamic reconfigurable function of the CIM and adds a coupling capacitor that reduces the nonlinear influence of the addition execution by the previous multiplication execution, thus improving the computational accuracy. Reference [52] proposed a novel ternary (±1, 0) 4T-2C cell consisting of two pairs of 2T-1C structures based on Current-domain computation.

### *B. NVM-CIM*

NVM-CIM expands the range of computing equipment use by making it possible for the device not to lose data after a power failure through the physical characteristics of the memory itself. Depending on the difference in the way data is stored and read/written in the memory, NVM-CIM can be categorized into charge-based Flash-CIM, and resistive-based CIM including ReRAM-CIM, MRAM-CIM, and PCM-CIM.

#### 1) FLASH-CIM

Flash memory consists of floating gate transistors, each of which constitutes a memory cell, and storing data is accomplished by controlling the charge on the floating gate. Electrons are injected into the floating gate by applying a high voltage, and a change in the charge on the floating gate affects the conduction state of the transistor to indicate different stored information (0 or 1). Unlike SRAM and SRAM, Flash must first remove the electrons on the floating gate with a reverse voltage when re-storing data, and the presence of a separate process for erasing data significantly

inhibits the actual read/write speed of Flash. Depending on the arrangement of floating gate transistors in series or parallel, Flash exists in NAND and NOR two types as shown in in FIGURE 5(a).

The architecture of multiple floating gate transistors in series in NAND-Flash makes it extremely difficult to perform MAC calculations with only small changes to the array structure as in other memories [53]. As FIGURE 5(b), [54] proposed a 3D NAND layout model based on MVM computation, the MAC operation is determined by applying an activation input voltage to one of the floating gate transistor layers and then reading the total architecture output current. A way to perform Boolean operation-based computation at NAND is also proposed in [55]. However, These two schemes are difficult to realize with current NAND manufacturing technology. Reference [56] proposed a heterogeneous scheme based on 3D chips with multiple dies for the work in [54]. However, the reliability of the related ideas needs to be verified, especially with respect to the temperature of the 3D chips and the non-ideal interference it brings.

Since the memory cells in NOR-Flash are connected in parallel with BL, NOR-based Flash-CIM is relatively easier to realize at present. Its operation of performing MAC computation is similar to that of DRAM-CIM based on current-domain [57]. An average pooling process is used to reduce the ADC overhead on the conversion of the MAC output current in [58] as shown in FIGURE 5(c). In [59], a 2T 3D Chiplets scheme is proposed to enhance the device functionality, and in [60], the computational density is enhanced by connecting two neighboring BL outputs in parallel and using positive and negative activation weight input voltages to realize the trinary computation as indicated in FIGURE 5(d).

#### 2) ReRAM-CIM, MRAM-CIM, AND PCM-CIM

As emerging nonvolatile memories, ReRAM, MRAM, and PCM are resistance memories that represent stored data based on the resistance value of the structure within the cell. Resistance memories based on binary computation represent the storage of two types of data based on the difference between the cell resistance values being in a high-resistance state (HRS) or a low-resistance state (LRS). ReRAM consists of a metal electrode layer (at both ends) and an insulating layer (in the middle) made of metal oxides, and the resistance is switched between HRS and LRS by applying a high voltage or current to the electrodes at both ends. MRAM consists of a Magnetic Tunnel Junction (MTJ) based on two layers of ferromagnetic material (one free layer and one fixed layer) and one insulating tunneling layer, and the direction of the magnetization of the free layer is altered by the applied current while the fixed layer remains unchanged. The two different resistance values of the MTJ can be presented by the change of the magnetization state (parallel or antiparallel). The PCM consists of an electrode layer (at both ends) and a phase change material layer, which utilizes the changes in the crystalline and amorphous states of the PCM to achieve two types of resistance values, HRS and LRS.

Differences in the resistive media and the mechanism of driving resistor changes in the three memories will cause them to have different storage densities and read/write speeds and lifetimes, but the same data read/write method makes the ReRAM-CIM, MRAM-CIM, and PCM-CIM designs basically the same. The 1T-1R resistance-based memories are arranged in a Crossbar arrangement with WLs and BLs connected, which allows them to realize MAC-based MVM computation with higher parallelism and memory density compared to conventional SRAM and DRAM. However, the continuous current during operation changes the resistance value of the memory cell, resulting in a negative effect of voltage nonlinearity that needs to be effectively addressed.

The weights of the neural network are stored as resistors in a Crossbar-arranged 1T-1R structure of the memory as shown in FIGURE 6(a), multiple activation pulse signals are fed to the WLs in [61], and the results of the current-domain-based MAC calculations are read out using an amplifier-based ADC according to the currents passing through the BLs. As illustrated in FIGURE 6(b), [62] improves the read/write efficiency by connecting a single WL to 4-cell BLs, providing a low-latency multi-bit computation scheme. Despite the authors' attempts to optimize in terms of weight mapping and SA output reading, the computational accuracy is still moderately reduced and area overhead is increased under this scheme.

When the storage density increases, resistive memories inevitably face the obstacle of too narrow sampling margins in the current-domain computation mode in terms of the output voltage to the BLs due to the parallel activation of more LRS cells. Reference [63] employs a voltage-domain based operation to maintain a stable sampling margin to mitigate ADC workload as indicated in FIGURE 6(c), and limits the effect of nonlinear levels in the voltage-domain mode by designing an active feedback control module.

In memory, a multi-level cell can provide higher storage density and execution efficiency than a single-level cell, which can be more conducive to data-focused neural network computation [64]. However, because of the non-ideal nature of resistance memory, implementing multi-level cell design on resistance memory is more likely to exacerbate the lack of computational accuracy than single-level cell. Design in [65] balances storage density, computational accuracy, and energy efficiency by performing 8-bit MAC computations with a hybrid single-level cell and multi-level cell architecture based on 1T-1R PCM-CIM as reflected in FIGURE 6(d), where the higher two weight bits are performed by single-level cell to ensure computational accuracy, and the lower six bits are performed by multi-level cell to improve area efficiency. In [66], attempts were made to mitigate the effects of nonlinearities by optimizing the module positions during the multi-array design. The ReRAM-CIM arrays were placed in a deep N-well to allow sufficient P-substrate bias to mitigate high voltage transfer, and the BL and source line (SL) drivers

were placed at the top and bottom of the array to minimize variations in resistive currents in the near and far cells due to voltage drop.

Reference [67] presents a Time-domain based ReRAM-CIM design to avoid the requirement of R for energy-efficient DC currents and its exposure to nonlinearities due to low voltages. The input activation signal is generated by integration-based voltage-to-time converter (IVTC), and after the MAC operation is completed in the array, the delay of the output current hold on is read out by the time-to-digital converter (TDC) and the computation result is derived from the timing calibration table (TCT).

A MRAM-CIM scheme based on XNOR digital Boolean operation is proposed by [68] as shown in FIGURE 6(e), which maintains high storage density by designing a pseudo 2T-2R CIM cell based on a standard 1T-1R cell using folded access transistors. The digital computation-based scheme avoids the interference of resistive memory analog computation and the design of complex peripheral circuits. The full transistor is energized when the weights are written, and the half transistor is energized by the input data on the WL when the CIM is executed, thus realizing high-speed XNOR computation.

## IV. PERIPHERAL CIRCUITS DESIGN

The peripheral circuits of the CIM are used to control the input/output signal and data read/write of the device and to assist the memory cells in the CIM array to complete computing operations. As the bit-based storage capacity of CIM continues to improve, the peripheral circuit overhead of CIM arrays is also increasing proportionally [69]. Even with area optimization, the SRAM-CIM peripheral circuits account for more than 50%, most of which is occupied by the DAC/ADC module as illustrated in FIGURE 7(a) [70]. The peripheral circuits in the resistance-based memory-based CIM, which has higher storage density and parallelism, are even more complex [63]. Meanwhile, due to the slow progress of the process, the enhancement of the storage density of CIM arrays has been gradually saturated, and the optimization of the overhead of the peripheral circuits has become a key step to improve the performance of CIM devices and has received widespread attention. This section will mainly discuss the progress of CIM devices in terms of peripheral circuit optimization.

### A. NO-DAC/ADC

Considering that the DAC/ADC module of the CIM device's peripheral circuitry imposes excessive overhead on the peripheral circuitry, No-DAC/ADC replaces the use of DAC/ADC area overhead by using other structures to read out the results of the computation.

Pure Boolean digital computation is the most direct way to No-DAC/ADC [27], [28], [29], [42], [43], [44], [68]. However, the computational efficiency of pure Boolean digital computation is low, and its essence is also equivalent to transferring part of the peripheral circuitry to the inside of the CIM array, which is more suitable for scenarios where the storage requirement is higher than the computation requirement. In recent years, there are also mixed-mode designs that reduce overhead by putting part of the addition computation outside the array [71].

The use of Time-domain based operation is another common means of No-DAC/ADC [39], [40], [41], [67]. By determining the hold on delay time of the current signal at a certain position to determine the MAC computation results, more complex DAC/ADC designs can be avoided and the impact of voltage/current nonlinearities needs not to be worried about. However, time-domain based operations require the use of a TDC to convert the hold on delay to the output binary computation result [72]. When performing large-scale Bit computations, the area overhead of the TDC is not less than that of a DAC/ADC [40], [41]. Nevertheless but the time-domain based approach has less computational overhead than the pure Boolean digital computation approach [73], and is also of interest in the last few years as it is effective in small-scale, high-precision and speed-demanding applications. Reference [74] is optimized for TDC-based time-domain computation, and a temporal-coding spiking neuron structure is proposed to perform time-domain-based multiplication for SRAM-CIM on BL and uses Adder-tree to achieve digital computation-based accumulation. This approach avoids complex TDC design at the output location and maintains high computational accuracy. A MAC output module based on a charge storage and integral counter is proposed in [75] to implement ADC-Free, and the input module is implemented based on the conversion of a spike pulse sequence. The accumulated charge of each BL is converted into the number of pulses and stored in a counter consisting of a D flip-flop and the MAC computed value is read out based on the number of pulses.

### B. LESS-DAC/ADC

Since it is difficult to perform more efficient analog computation in the high storage capacity CIM without using the DAC/ADC module in the surrounding circuitry and will limit the choice of how the device is computed, the Less-DAC/ADC scheme, which retains the DAC/ADC but optimizes its overhead, has a higher degree of applicability.

References [70] and [76] utilizes a dedicated reference voltage to perform input bit serial multiplication in the charge domain to replace the use of a DAC module during MAC input. Reference [76] reduces the use of a high-precision ADC with greater overhead by splitting the data stream, resulting in a compact ADC output readout module. Simultaneous use of an integrated ADC based on hybrid charge sharing improves ADC conversion time by utilizing a reference voltage. Reference [77] designs near-CIM analog memory and nonlinear activation units (NAUs) that can execute activation functions on the periphery of conventional digital-based CIM arrays to mitigate the power consumption and area overhead of the DAC/ADC

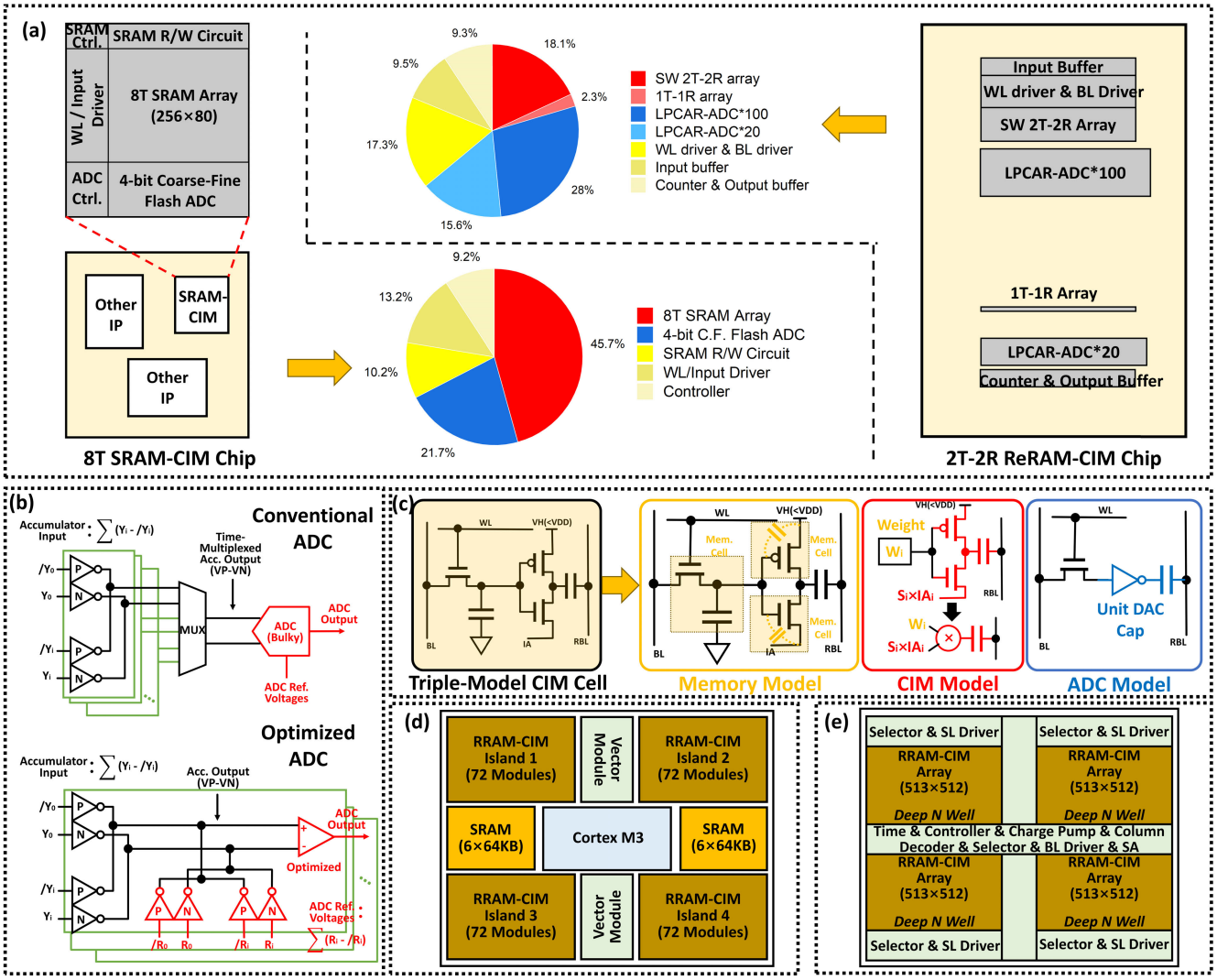


**FIGURE 7.** Review of CIM peripheral circuits optimization approaches. (a)Analysis of CIM peripheral circuits overhead in SRAM-CIM [70] and ReRAM-CIM [64]. (b) An optimized approach to Less-DAC/ADC based on replicated units and shared architectures [78]. (c)A dynamically reconfigurable triple-mode (Memory mode,CIM mode, and ADC model) DRAM-CIM cell design [51]. (d)Optimizing the CIM's peripheral circuits and mitigating the effects of device nonlinearities through a multi-array design [81], [82].

modules. Reference [78] designed a row-by-row ADC based on replicated bit cells and readout amplifiers as indicated in FIGURE 7(b). The shared architecture design allows the CIM array to halve the ADC requirements while significantly reducing the nonlinear variations in analog computation. However, it is necessary to be aware that the shared architecture causes a significant increase in wiring and may affect device heat dissipation.

### C. RECONFIGURABLE DAC/ADC

Since the design of the DAC/ADC is based on the BL of the CIM array, the requirements on the DAC/ADC to perform different bit computations vary significantly. A low-bit DAC/ADC cannot accurately read out the results of higher-bit MAC computations, while the continuous use of a high-bit DAC/ADC results in exponential waste of resources. Therefore compared to a fixed DAC/ADC design a Reconfigurable DAC/ADC based design can better cope with the varying workloads due to the CIM execution.

Reference [79] presents a CIM design containing a reconfigurable ADC capable of 1b to 5b outputs for performing a multilayer perceptron that achieves high classification accuracy while still maintaining excellent power efficiency. A differential merged array ADC is proposed by [80], which shares multiplicative value sampling capacitors with hybrid computing units within the array to save area and utilizes a multi-element sparse aware scheme to achieve dynamic ADC reconfiguration based on the analog signals generated by the in-array weights and activation input read/write process to save power consumption. Reference [51] proposes a dynamically reconfigurable three-mode DRAM-CIM cell as shown in FIGURE 7(c). The designed DRAM-CIM cell can

be flexibly adjusted to the memory, CIM computing, and ADC modes according to the workload to maximize hardware utilization. A hierarchical in-memory ADC is also used to share the capacitors of the DRAM-CIM cell to save area overhead.

### D. MULTI-ARRAY

In addition to direct optimization of DAC/ADC modules, multi-array design of CIM memory arrays is a common way to optimize peripheral circuits as reflected in FIGURE 7(d). Multiple-array designs can solve the problem of over-complexity of single-array circuits, and although additional control units may be required, multiple arrays can significantly improve device parallelism and utilization efficiency [81], [82]. The nonlinear effects of the device can be mitigated by placing peripheral circuits such as control and read/write in the middle of multiple arrays, and the sharing of peripheral circuit modules by multiple CIM arrays can be realized to reduce total overhead [66].

## V. NEURAL NETWORK BASED CIM APPLICATIONS

As shown in FIGURE 8, CIM is expected to be better suited for AI applications based on neural network models because it can efficiently perform parallel MAC computations. However, due to the storage density limitations of current CIM devices, applying the CIM computing paradigm to large-scale neural network models still faces considerable challenges. Therefore, at this stage, researchers have mainly selected some subdivided applications in neural networks and achieve a more efficient computing paradigm through CIM-based mapping and dataflow optimization.

### A. LOW-COMPUTILITY SPECIFIC EDGE AI APPLICATION

As Cloud AI requires high-computility, edge AI applications with low-computility are obviously more suitable for current CIM devices. Related work focuses on expanding the environment in which the CIM computing paradigm can be used and customised for specific tasks to achieve the most extreme performance.

#### 1) VISION AND SPEECH PROCESSING

Visual processing and speech recognition are currently the most common low-computility edge AI applications. Related products have been widely used in daily life, such as wearable devices, intelligent wake-up devices, and driver assistance device etc. A method for character recognition by designing an architecture of Tree-form memory modules is presented in [83]. The work in [84] presents a CIM-based optimization of the data flow during the processing of a pedestrian detection model in occupancy network-based autonomous driving. ML based neural network models can avoid repeated reading and writing of weights by fixing parameters through the CIM computational paradigm. Considering that edge neural network application chips are cost-sensitive, designing a specific SoC chip for each application can significantly raise the cost overhead. Reference [85] designed a scalable multi-chiplet modules combinatorial neural network processing system based on hierarchical pipeline for different edge neural network computational tasks and validated it using GSCD-based keyword recognition, Cifar 10-based image classification and Tiny-YOLO-based target detection neural network models. Edge AI applications typically need to maintain low-power operation during non-computation intensive periods. Reference [86] enables devices to maintain low-power operation most of the time through a visual-wake-word (VWW) detector by means of a capacitor and analog computation-based CIM that is woken up only when an event is detected, and designs an error desensitisation algorithm to optimise the lack of computational accuracy caused by the nonlinearity of charge-based analog computation. Neuromorphic vision sensors can save processing consumption during non-event triggering by performing data readout only when the pixel contrast exceeds the trigger value, however noise in the pixels degrades the image quality and therefore noise reduction processing needs to be maintained at all times. Reference [87] achieves low-power highly parallel pixel-consuming noise reduction processing acceleration by designing a near-sensor CIM. Reference [81] maps the neural network model of an edge recommender system that handles notifications or received events in the form of speech into a ReRAM-CIM device to achieve near-zero leakage power and complete power-down during non-computation period.

#### 2) DATA COMPRESSION AND ENCRYPTION

Data compression techniques can greatly reduce the data storage space and transmission bandwidth requirements, thus improving the efficiency and performance of neural network model training and inference, especially extremely important for edge AI computing devices with low-computility. Reference [88] demonstrates a design that uses the CIM architecture for compression of data from 3D stacked dynamic vision sensors (DVSs). The hardware-software co-design utilises a 4-bit shallow auto-encoder with nearly 10k parameters to achieve highly efficient real-time compression of 256 $\times$ 256 DVS data within 6 mW power consumption. Reference [89] proposed a simple lossless algorithm (SLA) for airborne hyperspectral imagery (HSI) data compression using the CIM architecture, which avoids the increased probability of error due to 'bit flipping' in the local difference (LD) decision module of the SLA when computing the absolute difference between neighbouring pixels in the von Neumann architecture and reduces the power consumption of the processing.

In addition to data compression, data encryption techniques, because of the important role they play in protecting sensitive data, safeguarding intellectual property, and ensuring compliance, are likewise having an increasingly widespread application in neural network-based AI field. The emphasis on data privacy protection and countering cyber-attacks has led to a continuous growth in the amount of data with encryption requirements, driving research into

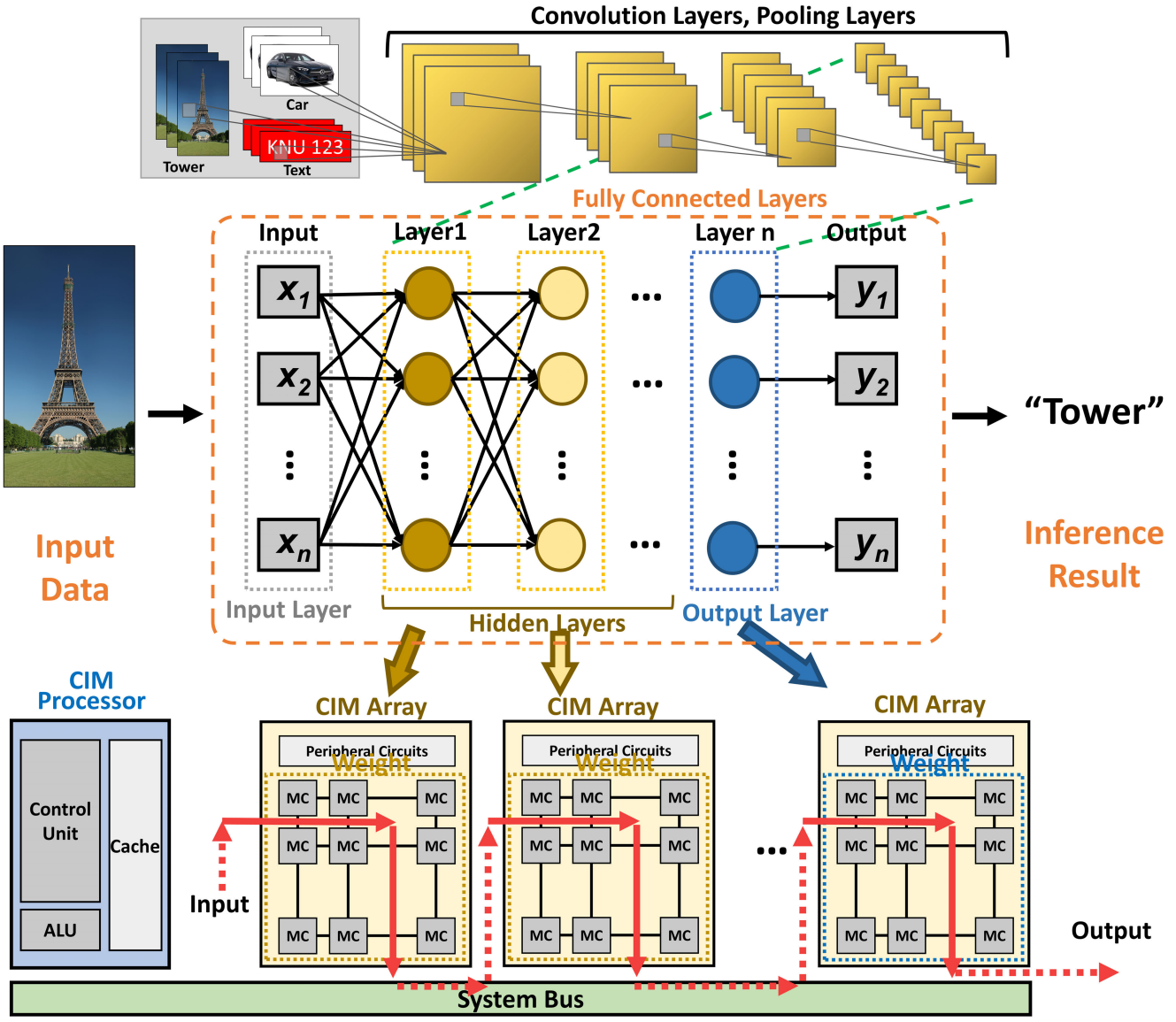


**FIGURE 8.** Demonstration of the application of CIM devices to neural network model computations. Mapping fully connected layers in neural network models to memory arrays in CIM devices for MAC or MVM computation acceleration.

more efficient data encryption and decryption systems. In [90], it is shown that encryption and decryption of 7 × 6 pixel text image data can be achieved without reading the device state using the XOR CIM architecture and Feistel cipher in an SRAM array. Reference [91] proposes a programmable CIM architecture for accelerating multi-mode advanced encryption standard (AES)-based data encryption and decryption. By placing the encryption and decryption modules in the middle of a column-based memory array, simultaneous encryption and decryption can be achieved by using a LUT buffer in parallel with the array's external processor module to maximise parallelism.

### 3) SAT SOLVER

Satisfiability (SAT) solvers determine the satisfiability of Boolean formulas, help optimise neural networks, enhance constraint satisfaction and verify model consistency, and can be used to solve complex combinatorial problems in areas such as automated planning, scheduling, and validation, and thus have a wide range of applications in industrial production. They play a key role in the decision-making process by encoding AI tasks as logical constraints and finding optimal solutions.

Since the initial variables of the SAT need to be stored in the memory module, [92] attempts to accelerate the computation of the popular Boolean SAT solver using the CIM architecture to avoid the energy consumption caused by performing off-chip memory accesses on a large scale for intensive industrial data during the computation. The work in [93] improves the flexibility of the SAT solver and its solvability for complex problems by implementing unsupervised learning to look for optimal solutions using

the CIM-based Recurrent Neural Network (RNN) module, which works in conjunction with the CIM-based SAT. [94], on the other hand, has undertaken a more customised CIM-based SAT solver design, which enables more efficient solvability through the key feature of the parallel uncorrelated variable processing element (VPE), which allows multiple uncorrelated variables to be processed at the same time, resulting in fast column-by-column SAT solving operations.

### B. FLOATING-POINT COMPUTATION

Floating-point computation can accurately represent small values, support complex mathematical and statistical operations, and ensure the accuracy of computations involving large data sets such as neural networks and high-performance computing, so it is indispensable in scientific research, finance, engineering and AI fields. Currently, with the development of science and technology, more and more applications and models involve floating-point calculations, while traditional computers can only store integers, complex floating-point computations increase the difficulty and time of computations, and the speed and ability of floating-point computations have become an important indicator of the computer processors performance.

Considering the need to deploy CIMs in high-computility Clound AI, the implementation of efficient floating-point computation in existing CIM devices is mandatory, and the current research on floating-point computation has become the main direction of the most advanced research work on CIM applications, which tests the computational accuracy of the devices.

Floating-point computation is standardised under IEEE 754 to ensure cross-platform consistency. Floating point values consist of 3 parts: sign, exponent, and mantissa, as in

$$Value = (-1)^{Sign} \times (Mantisssa) \times (2)^{Exponent}. \quad (4)$$

Floating-point computation consists of three processes: exponent matching, mantissa computation, and formatting. Implementing floating point calculations in CIM requires the components of a floating point number to be designed with the appropriate number of bits in the memory array. Since floating-point numbers with different numbers of bits consume different amounts of memory cells, it is necessary to ensure that there are enough bits available for floating-point numbers while avoiding, as much as possible, wasting bits during floating-point computations. In particular, the multiple floating-point number addition process requires complex exponent matching and mantissa shifting. References [95] and [96] designed the maximum 8b-based exponent E-max and pairing all exponents up to E-max. Total bit width of 16b is maintained after two mantissas shift, thus avoiding the complicated process of finding the maximum value of the exponents. Reference [97] designs a floating-point computation model for 4b mantissa and uses linear multiplication in posit format to replace the float format in IEEE 754. For numbers that are not too large, the posit format has a shorter exponent, so higher accuracy can be achieved in a CIM array of the same bits. A scheme for offline pre-pairing of its weights is proposed in [98]. That is, the weights are grouped by kernel, and the weights of each kernel share the same exponent, and different kernels exist with different exponent. In this way, only one exponent needs to be stored for each group of kernels, which will significantly reduce the storage space overhead, and the weights of the same exponent are computed together, which also helps to improve the computational efficiency. The work in [99] optimizes the per-bit MAC computation itself, which is based on CIM floating-point computation. A balance between computational accuracy and area overhead is achieved by performing 8b per-bit multiplication operations inside the memory cell array and adder trees outside the array. And the accumulation process uses a fixed number of bits shifted to avoid more hardware overhead.

### C. MODEL INFERENCE

Although the computational power of current CIM devices struggles to cope with large-scale neural network models, especially in model training, because of the need to constantly update the weights based on the results of the computation, the data communication capabilities of the devices are in high demand. However, it is still very attractive to try to accelerate the inference of neural network models using the CIM computational paradigm because the inference process weights are usually fixed and the data does not need to be read and written repeatedly. Moreover, the crossbar structure of the CIM device computing unit fits well with neural network models.

Convolutional Neural Network (CNN) is one of the most common neural network models, which mainly consists of convolutional, pooling and fully connected layers, and is characterised by the fact that features can be automatically extracted from the original data without the need for manual design. Since the weight values can be shared, CNNs require fewer parameters to be considered. A hybrid analog/digital structure CIM is designed in [100] to balance computational accuracy and energy efficiency for CNN-based ResNet-ImageNet model inference.

Spiking neural networks are based on the communication between neurons via spiking pulse signals and are computed only when a peak occurs (i.e., an input change), making them very advantageous for the processing of sudden events. Reference [101] designed a CIM-based SNN chip to accelerate inference on NMNIST and DVS-Gesture datasets. By exploiting the data inherent in the spikes, the convolutional SNN network scalability can be enhanced to achieve higher accuracy in the inference of the datasets with very few weights. Due to their event-driven nature, SNNs are well suited for deployment on NVM-CIMs such as ReRAM [102], [103] for inference.

## VI. CHALLENGES AND FUTURE TRENDS

Researchers have long been working to solve the various challenges of designing and applying CIM-based systems.

This section summarizes and analyzes the challenges and future development trends of CIM when applied to neural network models.

## A. CHALLENGES

### 1) MEMORY DENSITY ENHANCEMENT

The evolution of ML and DL applications based on neural network models necessitates a marked increase in memory density to accommodate the ever-expanding parameter sets of modern neural networks. Achieving higher memory density is hampered by the physical limitations of current semiconductor technologies, including issues related to thermal management and reliability. Optimization of the density of CIM storage cells under existing processes has gradually reached a saturation point. Innovations in materials science and memory architecture, such as the exploration of 3D-based multi-layer memory cells and advanced packaging techniques, are required to address these limitations and meet the stringent demands of CIM.

### 2) PERIPHERAL CIRCUITS OVERHEAD

The integration of CIM often involves significant peripheral circuit overhead, which can introduce latency and power inefficiencies. These circuits are essential for managing data movement and executing arithmetic operations. However, their complexity can lead to increased area costs on chips, thus limiting the feasibility of scaling. Efficient design methodologies, such as minimizing unnecessary data transfers through improved circuit architectures based on the application scenarios of CIM devices, will be crucial in mitigating these overheads. Advances in circuit design techniques, including asynchronous design and adaptive voltage scaling, will also play a pivotal role.

### 3) LACK OF RELIABLE DIMULATION TOOLS

A substantial barrier to the rapid prototyping and deployment of in-memory computing systems is the lack of simulation tools capable of accurately modeling the intricate behaviors of such architectures. Although some tools exist for CIM approaches, including DRAMsim [104], PIMSim [105], and CIM-SIM [106], simulation software can only validate system-level modeling and can only target specific architectures, making it difficult to simulate large-scale CIM arrays. It is even more difficult to efficiently help developers optimize computational efficiency. On the other hand, there is a lack of simulation tools for CIM chip device design, especially for CIM devices based on analog computation, which makes it difficult to fully validate the reliability of the architecture during the circuit development phase. Current tools may not adequately capture the nuances of non-volatile memory interactions, emerging device characteristics, or the heterogeneous nature of in-memory processing. The development of comprehensive, reliable simulation environments that can account for the diverse operational scenarios of in-memory computing will be imperative for validating designs and ensuring their performance under varied workloads.

### 4) LACK OF GENERAL-PURPOSE COMPILERS

The current memory capacity of CIM devices is low and far from being able to meet the requirements of conventional neural network models. Therefore, the neural network model needs to be segmented according to the storage density of the CIM, and then mapped to the CIM device for operation separately [107]. However, the existing CIM work is mostly an individualized implementation for specific scenarios in terms of compilation methods, and there is a lack of a general-purpose compiler based on CIM. This will increase the difficulty of design for software developers who do not have a good understanding of the underlying hardware architecture, which is not conducive to the application and promotion of CIM devices. In the field of software model development, the concept of Tensor operators has already been established [108]. The algorithm of a neural network is regarded as a series of tensor computations, and it can be better deployed on the GPU by segmenting it based on Tensor operators. A general-purpose compiler based on CIM needs to slice the model into suitable tensor operators, and then generate corresponding instructions to call the underlying hardware to process. Although there have been CIM-based tensor research [109], there is a lack of a general-purpose compiler that can efficiently connect different software algorithms in the upper layer with different CIM hardware in the lower layer, making it difficult to utilize the efficiency of the CIM chip.

### 5) LACK OF TURING COMPLETENESS VERIFICATION

Turing completeness is an important idea in computer science that implies that a programming language can implement any computable algorithm [110]. Current CIM devices lack Turing completeness, severely limiting their computational flexibility and general applicability. CIM devices are specialized for tasks such as matrix-vector multiplication and AI inference, making them incapable of executing arbitrary algorithms that require complex control flow, recursion, or branching. In particular, the approximate nature of many analog computation-based CIM architectures introduces accuracy issues that complicate error-sensitive calculations and reduce the reliability required for deterministic results. Considering a wider range of application scenarios and adaptability to future computing needs, CIM devices also need to continuously improve the capability and computational accuracy of complex computations in accordance with the goal of Turing completeness [111].

## B. FUTURE TRENDS

### 1) DEVICE (HARDWARE)

#### *a: COMBINATION OF MULTIPLE MODEL COMPUTATIONS*

In Section III, we introduced the advantages and disadvantages of various computation modes for CIM implementation.

Digital computation is accurate but inefficient, while simulation is more efficient but susceptible to nonlinear effects [112]. In the case of analog computations, the implementation methods based on the Voltage/Current-domain, Charge-domain and Time-domain also have their own characteristics. Neural network inference, especially edge-based applications, has different computational accuracy and power requirements depending on the task.Therefore, depending on the application scenario of the hardware, Combining multiple Computation modes in CIM architecture can better balance the overhead and requirements, and can achieve the ultimate in hardware performance based on computational efficiency [99], [113].

#### b: COMBINATIONS OF MULTIPLE MEMORIES

Most of the current CIM work is implemented based on SRAM because of mature fabrication process and the stable device operation. Despite the advantages of fast read/write and high computational accuracy of SRAM-CIM, SRAM has low storage density and data cannot be saved after power failure. Therefore, designing CIM devices with a mixture of memories will help to complement each other's strengths [114]. Use NVM-CIM for analog computing and long-term storage, DRAM-CIM for high-speed digital tasks, and SRAM-CIM for low-latency access. This hybrid storage structure allows task partitioning, thereby improving performance, energy efficiency and versatility when processing neural network models [115], [116].

#### c: CHIPLET AND 3D DESIGN

The transition towards chiplet architectures and 3D integration will enable more efficient interconnectivity and reduced distances for data transfer, directly addressing the bandwidth limitations of traditional chip designs [117]. And technology based on 3D and chiplets can also combine CNM and CIM chips to make the advantages of the von Neumann computing paradigm and CIM complementary [118]. By stacking multiple die in a single package, designers can optimize space and power efficiency while enhancing performance metrics like latency and bandwidth, crucial for real-time processing tasks in AI applications.

#### d: RECONFIGURABLE PERIPHERAL CIRCUITS

The negative impact of the overhead of peripheral circuits on the storage density and computing efficiency of CIM cannot be ignored, especially as CIM application scenarios expand and the complexity of data that needs to be stored and calculated continues to increase. Future designs will increasingly adopt reconfigurable peripheral circuits that can dynamically adjust to specific computational needs and workloads [51]. Such adaptability will enable more efficient resource allocation and will allow the hardware to support a wider range of applications, facilitating the transition from fixed-function architectures to more versatile computing environments.

### 2) APPLICATION (SOFTWARE)

#### a: APPLICATION PROGRAMMABLE INTERFACES

Considering the long development cycle of chip hardware and the fact that AI applications are significantly faster than hardware in terms of software, the programmability of the device must be improved in order to accelerate the application of CIM [119]. The development of CIM-based application programmable interfaces (APIs) for commonly used software algorithms will help address this issue. The emergence of dedicated APIs for in-memory computing can significantly simplify the software development process, enabling software developers to create applications that seamlessly utilize the unique capabilities of in-memory architectures. These APIs will simplify the complexity of hardware interactions and enable more direct integration of advanced computing capabilities into existing frameworks.

#### b: HARDWARE AND SOFTWARE CO-DESIGN

The future landscape of CIM will increasingly emphasize the co-design of hardware and software. This approach will enable tighter integration of system components, allowing for optimizations that consider both hardware capabilities and software demands [120]. Hardware and software co-design reduces bottlenecks by ensuring that hardware resources are aligned with software requirements, which is critical for applications in edge computing and mobile devices [121]. By collaborating across disciplines, engineers can innovate solutions that fully exploit the advantages of in-memory architectures, resulting in systems that are not only more efficient but also tailored to specific application requirements.

## VII. CONCLUSION

Neural network models, which make decisions in a manner similar to the human brain, have gradually become the main direction of work in the field of artificial intelligence due to their excellent performance in dealing with complex problems. This paper analyzes the bottlenecks faced by current computing systems based on von Neumann architectures in dealing with AI problems based on neural network models, and points out that the CIM-based computing paradigm is an effective means to solve the problem. This paper reviews excellent works related to CIM in recent years in terms of arrays of CIM device memory cells, array peripheral circuits, and CIM-based application design. It also summarizes the challenges and possible solutions to the current CIM work. As the reliability of materials and manufacturing processes continue to advance, and circuit design solutions including 3D chips continue to improve, related problems are gradually being overcome. And CIM applications in more fields continue to be attempted, which will arouse the attention and interest of more hardware and software developers, and gradually form a unified standard in the industry. In the future, with the pursuit of high computational efficiency, CIM-based computation will continue to play an increasingly important role in various fields, including neural network modeling.

## REFERENCES

[1] P. Darche, "Computation model and architecture: Illustration with the von Neumann approach," in *Microprocessor 1: Prolegomena-Calculation and Storage Functions-Models of Computation and Computer Architecture*. Hoboken, NJ, USA: Wiley, 2020, pp. 63–130, doi: 10.1002/9781119779667.

[2] A. Marowka, "Pitfalls and issues of manycore programming," *Adv. Comput.*, vol. 79, pp. 71–117, Jan. 2010, doi: 10.1016/S0065-2458(10)79002-1.

[3] R. V. Zicari, J. Brodersen, J. Brusseau, B. Dudder, T. Eichhorn, T. Ivanov, G. Kararigas, P. Kringen, M. McCullough, F. Moslein, N. Mushtaq, G. Roig, N. Sturtz, K. Tolle, J. J. Tithi, I. van Halem, and M. Westerlund, "Z-inspection: A process to assess trustworthy AI," *IEEE Trans. Technol. Soc.*, vol. 2, no. 2, pp. 83–97, Jun. 2021, doi: 10.1109/TTS.2021.3066209.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386.

[5] K. Yu, B. Yusupbaev, M. Kim, and J. R. Choi, "Application of crop-sum algorithm to character recognition and pedestrian detection by memory-centric computing," in *Proc. TENCON IEEE Region 10 Conf. (TENCON)*. Chiang Mai, Thailand: TENCON, Oct. 2023, pp. 1–6, doi: 10.1109/TENCON58879.2023.10322394.

[6] W. Liu, G. Zeng, and K. Hu, "Growth scale prediction of big data for information systems based on a deep learning SAEP method," *IEEE Access*, vol. 8, pp. 62883–62894, 2020, doi: 10.1109/ACCESS.2020.2966770.

[7] A. Ankit, I. Chakraborty, A. Agrawal, M. Ali, and K. Roy, "Circuits and architectures for in-memory computing-based machine learning accelerators," *IEEE Micro*, vol. 40, no. 6, pp. 8–22, Nov. 2020, doi: 10.1109/MM.2020.3025863.

[8] M. Ali, S. Roy, U. Saxena, T. Sharma, A. Raghunathan, and K. Roy, "Compute-in-memory technologies and architectures for deep learning workloads," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 11, pp. 1615–1630, Nov. 2022, doi: 10.1109/TVLSI.2022.3203583.

[9] C. Zhang, H. Sun, S. Li, Y. Wang, H. Chen, and H. Liu, "A survey of memory-centric energy efficient computer architecture," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 10, pp. 2657–2670, Oct. 2023, doi: 10.1109/TPDS.2023.3297595.

[10] K. Kim and M.-J. Park, "Present and future, challenges of high bandwith memory (HBM)," in *Proc. IEEE Int. Memory Workshop (IMW)*, Seoul, South Korea, May 2024, pp. 1–4, doi: 10.1109/IMW59701.2024.10536972.

[11] W. Jung et al., "A 280-layer 1Tb 4b/cell 3D-NAND flash memory with a 28.5Gb/mm$^2$ areal density and a 3.2GB/s high-speed IO rate," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 236–237, doi: 10.1109/isscc49657.2024.10454343.

[12] Y. Li, T. Bai, X. Xu, Y. Zhang, B. Wu, H. Cai, B. Pan, and W. Zhao, "A survey of MRAM-centric computing: From near memory to in memory," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 2, pp. 318–330, Apr. 2023, doi: 10.1109/TETC.2022.3214833.

[13] A. Yusuf, T. Adegbija, and D. Gajaria, "Domain-specific STT-MRAM-based in-memory computing: A survey," *IEEE Access*, vol. 12, pp. 28036–28056, 2024, doi: 10.1109/ACCESS.2024.3365632.

[14] N. Lepri, A. Glukhov, L. Cattaneo, M. Farronato, P. Mannocci, and D. Ielmini, "In-memory computing for machine learning and deep learning," *IEEE J. Electron Devices Soc.*, vol. 11, pp. 587–601, 2023, doi: 10.1109/JEDS.2023.3265875.

[15] S. Yu and P.-Y. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid StateCircuits Mag.*, vol. 8, no. 2, pp. 43–56, Spring. 2016, doi: 10.1109/MSSC.2016.2546199.

[16] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnol.*, vol. 15, no. 7, pp. 529–544, Jul. 2020, doi: 10.1038/s41565-020-0655-z.

[17] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020, doi: 10.1109/COMST.2020.2970550.

[18] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, U. Gana, and M. U. Kiru, "Comprehensive review of artificial neural network applications to pattern recognition," *IEEE Access*, vol. 7, pp. 158820–158846, 2019, doi: 10.1109/ACCESS.2019.2945545.

[19] J. Qi, J. Du, S. M. Siniscalchi, and C.-H. Lee, "A theory on deep neural network based Vector-to-Vector regression with an illustration of its expressive power in speech enhancement," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 12, pp. 1932–1943, Dec. 2019, doi: 10.1109/TASLP.2019.2935891.

[20] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[21] K. L. Tan, C. P. Lee, K. S. M. Anbananthen, and K. M. Lim, "RoBERTa-LSTM: A hybrid model for sentiment analysis with transformer and recurrent neural network," *IEEE Access*, vol. 10, pp. 21517–21525, 2022, doi: 10.1109/ACCESS.2022.3152828.

[22] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 87–110, Jan. 2023, doi: 10.1109/TPAMI.2022.3152247.

[23] Q. Dong and G. Luo, "Progress estimation for end-to-end training of deep learning models with online data preprocessing," *IEEE Access*, vol. 12, pp. 18658–18684, 2024, doi: 10.1109/ACCESS.2024.3359996.

[24] K. Asifuzzaman, N. R. Miniskar, A. R. Young, F. Liu, and J. S. Vetter, "A survey on processing-in-memory techniques: Advances and challenges," *Memories -Mater., Devices, Circuits Syst.*, vol. 4, Jul. 2023, Art. no. 100022, doi: 10.1016/j.memori.2022.100022.

[25] Z. Zhang, J. Jiang, Y. Zhu, Q. Wang, Z. Mao, and N. Jing, "A universal RRAM-based DNN accelerator with programmable cross-bars beyond MVM operator," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 7, pp. 2094–2106, Jul. 2022, doi: 10.1109/TCAD.2021.3107252.

[26] B. Feinberg, U. K. R. Vengalam, N. Whitehair, S. Wang, and E. Ipek, "Enabling scientific computing on memristive accelerators," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Los Angeles, CA, USA, Jun. 2018, pp. 367–382, doi: 10.1109/ISCA.2018.00039.

[27] H. Kim, T. Yoo, T. T. Kim, and B. Kim, "Colonnade: A reconfigurable SRAM-based digital bit-serial compute-in-memory macro for processing neural networks," *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, Jul. 2021, doi: 10.1109/JSSC.2021.3061508.

[28] Y.-D. Chih, P.-H. Lee, H. Fujiwara, Y.-C. Shih, C.-F. Lee, R. Naous, Y.-L. Chen, C.-P. Lo, C.-H. Lu, H. Mori, W.-C. Zhao, D. Sun, M. E. Sinangil, Y.-H. Chen, T.-L. Chou, K. Akarvardar, H.-J. Liao, Y. Wang, M.-F. Chang, and T. J. Chang, "An 89TOPS/W and 16.3TOPS/mm$^2$ all-digital SRAM-based full-precision compute-in memory macro in 22 nm for machine-learning edge applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, San Francisco, CA, USA, Feb. 2021, pp. 252–254, doi: 10.1109/ISSCC42613.2021.9365766.

[29] M. Gupta, S. Cosemans, P. Debacker, and W. Dehaene, "A 2Mbit digital in-memory computing matrix-vector multiplier for DNN inference supporting flexible bit precision and matrix size achieving 612 binary TOPS/W," in *Proc. IEEE 49th Eur. Solid State Circuits Conf. (ESSCIRC)*, Lisbon, Portugal, Sep. 2023, pp. 417–420, doi: 10.1109/ESSCIRC59616.2023.10268763.

[30] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for Binary/Ternary deep neural networks," *IEEE J. Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, Jun. 2020, doi: 10.1109/JSSC.2019.2963616.

[31] X. Si et al., "A 28 nm 64Kb 6T SRAM computing-in-memory macro with 8b MAC operation for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2020, pp. 246–248, doi: 10.1109/ISSCC19947.2020.9062995.

[32] C. Yu, T. Yoo, K. T. C. Chai, T. T. Kim, and B. Kim, "A 65 nm 8T SRAM compute-in-memory macro with column ADCs for processing neural networks," *IEEE J. Solid-State Circuits*, vol. 57, no. 11, pp. 3466–3476, Nov. 2022, doi: 10.1109/JSSC.2022.3162602.

[33] B. Wang, C. Xue, Z. Feng, Z. Zhang, H. Liu, L. Ren, X. Li, A. Yin, T. Xiong, Y. Xue, S. He, Y. Kong, Y. Zhou, A. Guo, X. Si, and J. Yang, "A 28 nm horizontal-weight-shift and vertical-feature-shift-based separate-WL 6T-SRAM computation-in-memory unit-macro for edge depthwise neural-networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 134–136, doi: 10.1109/ISSCC42615.2023.10067526.

[34] Z. Chen, Z. Wen, W. Wan, A. R. Pakala, Y. Zou, W.-C. Wei, Z. Li, Y. Chen, and K. Yang, "PICO-RAM: A PVT-insensitive analog compute-in-memory SRAM macro with in situ multi-bit charge computing and 6T thin-cell-compatible layout," *IEEE J. Solid-State Circuits*, early access, Aug. 13, 2024, doi: 10.1109/JSSC.2024.3422826.

[35] B. Zhang, J. Saikia, J. Meng, D. Wang, S. Kwon, S. Myung, H. Kim, S. J. Kim, J.-S. Seo, and M. Seok, "MACC-SRAM: A multistep accumulation capacitor-coupling in-memory computing SRAM macro for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 59, no. 6, pp. 1938–1949, Jun. 2024, doi: 10.1109/JSSC.2023.3332017.

[36] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: An in-memory-computing SRAM macro based on robust capacitive coupling computing mechanism," *IEEE J. Solid-State Circuits*, vol. 55, no. 7, pp. 1888–1897, Jul. 2020, doi: 10.1109/JSSC.2020.2992886.

[37] B. Zhang, S. Yin, M. Kim, J. Saikia, S. Kwon, S. Myung, H. Kim, S. J. Kim, J.-S. Seo, and M. Seok, "PIMCA: A programmable in-memory computing accelerator for energy-efficient DNN inference," *IEEE J. Solid-State Circuits*, vol. 58, no. 5, pp. 1436–1449, May 2023, doi: 10.1109/JSSC.2022.3211290.

[38] H. Jia, M. Ozatay, Y. Tang, H. Valavi, R. Pathak, J. Lee, and N. Verma, "Scalable and programmable neural network inference accelerator based on in-memory computing," *IEEE J. Solid-State Circuits*, vol. 57, no. 1, pp. 198–211, Jan. 2022, doi: 10.1109/JSSC.2021.3119018.

[39] C. Xue, X. Qiao, X. Ren, G. Du, Y. Wang, and Y. He, "A 768.7–2124.2 TOPS/W time-domain computing-in-memory macro with low static leakage and precision-configurable TDC," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 71, no. 5, pp. 2789–2793, May 2024, doi: 10.1109/TCSII.2023.3345878.

[40] J. Song, Y. Wang, M. Guo, X. Ji, K. Cheng, Y. Hu, X. Tang, R. Wang, and R. Huang, "TD-SRAM: Time-domain-based in-memory computing macro for binary neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 8, pp. 3377–3387, Aug. 2021, doi: 10.1109/TCSI.2021.3083275.

[41] P.-C. Wu, J.-W. Su, Y.-L. Chung, L.-Y. Hong, J.-S. Ren, F.-C. Chang, Y. Wu, H.-Y. Chen, C.-H. Lin, H.-M. Hsiao, S.-H. Li, S.-S. Sheu, S.-C. Chang, W.-C. Lo, C.-I. Wu, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "An 8b-precision 6T SRAM computing-in-memory macro using time-domain incremental accumulation for AI edge chips," *IEEE J. Solid-State Circuits*, vol. 59, no. 7, pp. 2297–2309, Jul. 2024, doi: 10.1109/JSSC.2023.3343669.

[42] M. F. Ali, A. Jaiswal, and K. Roy, "In-memory low-cost bit-serial addition using commodity DRAM technology," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 1, pp. 155–165, Jan. 2020, doi: 10.1109/TCSI.2019.2945617.

[43] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry, "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Boston, MA, USA, Oct. 2017, pp. 273–287, doi: 10.1145/3123939.3124544.

[44] X. Xin, Y. Zhang, and J. Yang, "ELP2IM: Efficient and low power bitwise operation processing in DRAM," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, San Diego, CA, USA, Feb. 2020, pp. 303–314, doi: 10.1109/HPCA47549.2020.00033.

[45] S. Xie, C. Ni, A. Sayal, P. Jain, F. Hamzaoglu, and J. P. Kulkarni, "EDRAM-CIM: Reconfigurable charge domain compute-in-memory design with embedded dynamic random access memory array realizing adaptive data converters," *IEEE J. Solid-State Circuits*, vol. 59, no. 6, pp. 1950–1961, Jun. 2024, doi: 10.1109/JSSC.2023.3326094.

[46] S. Kim, S. Kim, S. Um, S. Kim, Z. Li, S. Kim, W. Jo, and H.-J. Yoo, "A reconfigurable 1T1C eDRAM-based spiking neural network computing-in-memory processor for high system-level efficiency," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Monterey, CA, USA, May 2023, pp. 1–5, doi: 10.1109/ISCAS46773.2023.10181420.

[47] T. E. Jang, K. H. Lee, G. Y. Kim, S. Y. Yun, D.-H. Youn, H. Choi, J. Kim, S. Y. Kim, and M. Song, "Compute-in-memory with SAR ADC and 2T1C DRAM for MAC operations," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Taipei, Taiwan, Jan. 2024, pp. 1–3, doi: 10.1109/iceic61013.2024.10457128.

[48] T. C. Kao, M. J. Huang, Y. R. Liu, Y. K. Wang, J. C. Guo, and S. S. Chung, "An ultra-low voltage auger-recombination enhanced hot hole injection scheme in implementing a 3 bits per cell e-DRAM CIM macro for inference accelerator," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, Honolulu, HI, USA, Jun. 2024, pp. 1–2, doi: 10.1109/vlsitechnologyandcir46783.2024.10631412.

[49] Y. Zhao, Z. Shen, J. Xu, K. C. T. Chai, Y. Wu, and C. Wang, "A novel transpose 2T-DRAM based computing-in-memory architecture for on-chip DNN training and inference," in *Proc. IEEE 5th Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Hangzhou, China, Jun. 2023, pp. 1–4, doi: 10.1109/AICAS57966.2023.10168641.

[50] Z. Chen, X. Chen, and J. Gu, "A 65 nm 3T dynamic analog RAM-based computing-in-memory macro and CNN accelerator with retention enhancement, adaptive analog sparsity and 44TOPS/W system energy efficiency," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, San Francisco, CA, USA, Feb. 2021, pp. 240–242, doi: 10.1109/ISSCC42613.2021.9366045.

[51] S. Kim, Z. Li, S. Um, W. Jo, S. Ha, J. Lee, S. Kim, D. Han, and H.-J. Yoo, "DynaPlasia: An eDRAM in-memory-computing-based reconfigurable spatial accelerator with triple-mode cell for dynamic resource switching," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 256–258, doi: 10.1109/ISSCC42615.2023.10067352.

[52] C. Yu, T. Yoo, H. Kim, T. T. Kim, K. C. T. Chuan, and B. Kim, "A logic-compatible eDRAM compute-in-memory with embedded ADCs for processing neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 2, pp. 667–679, Feb. 2021, doi: 10.1109/TCSI.2020.3036209.

[53] A. C. Undavalli, G. Cauwenberghs, A. Natarajan, S. Chakrabartty, and A. Nagulu, "ADC-less 3D-NAND compute-in-memory architecture using margin propagation," in *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Tempe, AZ, USA, Aug. 2023, pp. 89–92, doi: 10.1109/MWSCAS57524.2023.10406082.

[54] P. Wang, F. Xu, B. Wang, B. Gao, H. Wu, H. Qian, and S. Yu, "Three-dimensional NAND flash for vector–matrix multiplication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 4, pp. 988–991, Apr. 2019, doi: 10.1109/TVLSI.2018.2882194.

[55] B. S. Swaroop, A. Saxena, and S. Sahay, "Satisfiability attack-resilient camouflaged multiple multivariable logic-in-memory exploiting 3D NAND flash array," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 71, no. 2, pp. 660–669, Feb. 2024, doi: 10.1109/TCSI.2023.3326332.

[56] P.-K. Hsu, V. Garg, A. Lu, and S. Yu, "A heterogeneous platform for 3D NAND-based in-memory hyperdimensional computing engine for genome sequencing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 71, no. 4, pp. 1628–1637, Apr. 2024, doi: 10.1109/TCSI.2024.3362354.

[57] Y. Feng, B. Chen, J. Liu, Z. Sun, H. Hu, J. Zhang, X. Zhan, and J. Chen, "Design-technology co-optimizations (DTCO) for general-purpose computing in-memory based on 55nm NOR flash technology," in *IEDM Tech. Dig.*, San Francisco, CA, USA, Dec. 2021, pp. 1–4, doi: 10.1109/IEDM19574.2021.9720625.

[58] Y. C. Xiang, P. Huang, Z. Zhou, R. Z. Han, Y. N. Jiang, Q. M. Shu, Z. Q. Su, Y. B. Liu, X. Y. Liu, and J. F. Kang, "Analog deep neural network based on NOR flash computing array for high speed/energy efficiency computation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Sapporo, Japan, May 2019, pp. 1–4, doi: 10.1109/ISCAS.2019.8702401.

[59] H.-T. Lue, T.-H. Hsu, T.-H. Yeh, W.-C. Chen, C. R. Lo, C.-T. Huang, G.-R. Lee, C.-J. Chiu, K.-C. Wang, and C.-Y. Lu, "A vertical 2T NOR (V2T) architecture to enable scaling and low-power solutions for NOR flash technology," in *Proc. IEEE Symp. VLSI Technol.*, Honolulu, HI, USA, Jun. 2020, pp. 1–2, doi: 10.1109/VLSITECHNOLOGY18217.2020.9265037.

[60] M.-L. Wei, H.-T. Lue, S.-Y. Ho, Y.-P. Lin, T.-H. Hsu, C.-C. Hsieh, Y.-C. Li, T.-H. Yeh, S.-H. Chen, Y.-H. Jhu, H.-P. Li, H.-W. Hu, C.-H. Hung, K.-C. Wang, and C.-Y. Lu, "Analog computing in memory (CIM) technique for general matrix multiplication (GEMM) to support deep neural network (DNN) and cosine similarity search computing using 3D AND-type NOR flash devices," in *IEDM Tech. Dig.*, San Francisco, CA, USA, Dec. 2022, pp. 33.3.1–33.3.4, doi: 10.1109/IEDM45625.2022.10019495.

[61] C.-X. Xue et al., "A 1Mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN based AI edge processors," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2019, pp. 388–390, doi: 10.1109/ISSCC.2019.8662395.

[62] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, Y.-R. Chen, T.-H. Hsu, Y.-K. Chen, Y.-C. Lo, T.-H. Wen, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 22 nm 2Mb ReRAM compute-in-memory macro with 121–28TOPS/W for multibit MAC computing for tiny AI edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2020, pp. 244–246, doi: 10.1109/ISSCC19947.2020.9063078.

[63] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A 40 nm 64Kb 56.67TOPS/W read-disturb-tolerant compute-in-memory/digital RRAM macro with active-feedback-based read and in-situ write verification," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, San Francisco, CA, USA, Feb. 2021, pp. 404–406, doi: 10.1109/ISSCC42613.2021.9365926.

[64] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen, J. Tang, Y. Wang, M.-F. Chang, H. Qian, and H. Wu, "A fully integrated analog ReRAM based 78.4TOPS/W compute-in-memory chip with fully parallel MAC computing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2020, pp. 500–502, doi: 10.1109/ISSCC19947.2020.9062953.

[65] W.-S. Khwa, Y.-C. Chiu, C.-J. Jhang, S.-P. Huang, C.-Y. Lee, T.-H. Wen, F.-C. Chang, S.-M. Yu, T.-Y. Lee, and M.-F. Chang, "A 40 nm, 2M-cell, 8b-precision, hybrid SLC-MLC PCM computing-in-memory macro with 20.5–65.0TOPS/W for tiny-al edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, San Francisco, CA, USA, Feb. 2022, pp. 1–3, doi: 10.1109/ISSCC42614.2022.9731670.

[66] J. Yang, X. Xue, X. Xu, Q. Wang, H. Jiang, J. Yu, D. Dong, F. Zhang, H. Lv, and M. Liu, "A 14 nm-FinFET 1Mb embedded 1T1R RRAM with a 0.022$\mu$m$^2$ cell size using self-adaptive delayed termination and multi-cell reference," in *Proc. IEEE Int. Solid- State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2021, pp. 336–338, doi: 10.1109/ISSCC42613.2021.9365945.

[67] J.-M. Hung, Y.-H. Huang, S.-P. Huang, F.-C. Chang, T.-H. Wen, C.-I. Su, W.-S. Khwa, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Y.-D. Chih, T. J. Chang, and M.-F. Chang, "An 8-Mb DC-current-free binary-to-8b precision ReRAM nonvolatile computing-in-memory macro using time-space-readout with 1286.4-21.6TOPS/W for edge-AI devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, San Francisco, CA, USA, Feb. 2022, pp. 1–3, doi: 10.1109/ISSCC42614.2022.9731715.

[68] H. Cai, Z. Bian, Y. Hou, Y. Zhou, J.-L. Cui, Y. Guo, X. Tian, B. Liu, X. Si, Z. Wang, J. Yang, and W. Shan, "A 28 nm 2Mb STT-MRAM computing-in-memory macro with a refined bit-cell and 22.4–41.5TOPS/W for AI inference," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 500–502, doi: 10.1109/ISSCC42615.2023.10067339.

[69] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 14–26, doi: 10.1109/ISCA.2016.12.

[70] K. Lee, J. Kim, and J. Park, "A 28 nm 50.1-TOPS/W P-8T SRAM compute-in-memory macro design with bl charge-sharing-based in-SRAM DAC/ADC operations," *IEEE J. Solid-State Circuits*, vol. 59, no. 6, pp. 1926–1937, Jun. 2024, doi: 10.1109/JSSC.2023.3334566.

[71] X. Sun, W. Cao, B. Crafton, K. Akarvardar, H. Mori, H. Fujiwara, H. Noguchi, Y.-D. Chih, M.-F. Chang, Y. Wang, and T.-Y.-J. Chang, "Efficient processing of MLPerf mobile workloads using digital compute-in-memory macros," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 4, pp. 1191–1205, Apr. 2024, doi: 10.1109/TCAD.2023.3333290.

[72] L. Chang, S. Yang, Z. Chang, H. Fan, J. Zhou, and J. Zhou, "TDPRO: Time-domain-based computing-in memory engine for ultra-low power ECG processor," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 10, pp. 3908–3919, Oct. 2023, doi: 10.1109/TCSI.2023.3294181.

[73] Y. Zhang, J. Wang, C. Lian, Y. Bai, G. Wang, Z. Zhang, Z. Zheng, L. Chen, K. Zhang, G. Sirakoulis, and Y. Zhang, "Time-domain computing in memory using spintronics for energy-efficient convolutional neural network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 3, pp. 1193–1205, Mar. 2021, doi: 10.1109/TCSI.2021.3055830.

[74] Z. Xuan, C. Liu, Y. Zhang, Y. Li, and Y. Kang, "A brain-inspired ADC-free SRAM-based in-memory computing macro with high-precision MAC for AI application," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 4, pp. 1276–1280, Apr. 2023, doi: 10.1109/TCSII.2022.3224049.

[75] H. Zhang, J. Liu, W. Kang, Y. Fan, S. Fu, J. Bai, B. Pan, Y. Liu, and W. Zhao, "A 40 nm 33.6Tops/W 8T-SRAM computing-in-memory macro with DAC-less spike-pulse-truncation input and ADC-less charge-reservoir-integrate-counter output," in *Proc. IEEE Int. Conf. Integr. Circuits, Technol. Appl. (ICTA)*, Zhuhai, China, Nov. 2021, pp. 123–124, doi: 10.1109/ICTA53157.2021.9661898.

[76] R. Sehgal, T. Thareja, S. Xie, C. Ni, and J. P. Kulkarni, "A bit-serial, compute-in-SRAM design featuring hybrid-integrating ADCs and input dependent binary scaled precharge eliminating DACs for energy-efficient DNN inference," *IEEE J. Solid-State Circuits*, vol. 58, no. 7, pp. 2109–2124, Jul. 2023, doi: 10.1109/JSSC.2023.3235210.

[77] P. Chen, M. Wu, W. Zhao, Y. Ma, T. Jia, and L. Ye, "A 44.3 TOPS/W SRAM compute-in-memory with near-CIM analog memory and activation for DAC/ADC-less operations," *IEEE Solid-State Circuits Lett.*, vol. 7, pp. 299–302, 2024, doi: 10.1109/LSSC.2024.3418099.

[78] J. Mu, H. Kim, and B. Kim, "SRAM-based in-memory computing macro featuring voltage-mode accumulator and row-by-row ADC for processing neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 6, pp. 2412–2422, Jun. 2022, doi: 10.1109/TCSI.2022.3152653.

[79] Q. Zang, W. L. Goh, L. Lu, C. Yu, J. Mu, T. T.-H. Kim, B. Kim, D. Li, and A. T. Do, "282-to-607 TOPS/W, 7T-SRAM based CiM with reconfigurable column SAR ADC for neural network processing," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Monterey, CA, USA, May 2023, pp. 1–5, doi: 10.1109/ISCAS46773.2023.10181435.

[80] L. Wang, W. Li, Z. Zhou, H. Gao, Z. Li, W. Ye, H. Hu, J. Liu, J. Yue, J. Yang, Q. Luo, C. Dou, Q. Liu, and M. Liu, "A flash-SRAM-ADC-fused plastic computing-in-memory macro for learning in neural networks in a standard 14nm FinFET process," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 582–584, doi: 10.1109/ISSCC49657.2024.10454372.

[81] M. Chang, S. D. Spetalnick, B. Crafton, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A 40 nm 60.64TOPS/W ECC-capable compute-in-memory/digital 2.25MB/768KB RRAM/SRAM system with embedded cortex M3 microprocessor for edge recommendation systems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, San Francisco, CA, USA, Feb. 2022, pp. 1–3, doi: 10.1109/ISSCC42614.2022.9731679.

[82] Y. He, S. Fan, X. Li, L. Lei, W. Jia, C. Tang, Y. Li, Z. Huang, Z. Du, J. Yue, X. Li, H. Yang, H. Jia, and Y. Liu, "A 28nm 2.4Mb/mm$^2$ 6.9–16.3TOPS/mm$^2$ eDRAM-LUT-based digital-computing-in-memory macro with in-memory encoding and refreshing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 578–580, doi: 10.1109/ISSCC49657.2024.10454323.

[83] K. Yu, M. Kim, and J. R. Choi, "Memory-tree based design of optical character recognition in FPGA," *Electronics*, vol. 12, no. 3, p. 754, Feb. 2023, doi: 10.3390/electronics12030754.

[84] K. Yu, J. Baek, B. Yusupbaev, and J. R. Choi, "Architecture design for pedestrian detection based on memory grid occupancy and data reuse," in *Proc. Int. Tech. Conf. Circuits/Systems, Comput., Commun. (ITC-CSCC)*, Okinawa, Japan, Jul. 2024, pp. 1–4, doi: 10.1109/itc-cscc62988.2024.10628251.

[85] H. Zhu, B. Jiao, J. Zhang, X. Jia, Y. Wang, T. Guan, S. Wang, D. Niu, H. Zheng, C. Chen, M. Wang, L. Zhang, X. Zeng, Q. Liu, Y. Xie, and M. Liu, "COMB-MCM: Computing-on-memory-boundary NN processor with bipolar bitwise sparsity optimization for scalable multi-chiplet-module edge machine learning," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 65, San Francisco, CA, USA, Feb. 2022, pp. 1–3, doi: 10.1109/ISSCC42614.2022.9731657.

[86] S.-E. Hsieh, C.-H. Wei, C.-X. Xue, H.-W. Lin, W.-H. Tu, E.-J. Chang, K.-T. Yang, P.-H. Chen, W.-N. Liao, L. L. Low, C.-D. Lee, A.-C. Lu, J. Liang, C.-C. Cheng, and T.-H. Kang, "A 70.85–86.27TOPS/W PVT-insensitive 8b word-wise ACIM with post-processing relaxation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 136–138, doi: 10.1109/ISSCC42615.2023.10067335.

[87] X. Zhang and A. Basu, "A 915–1220 TOPS/W, 976–1301 GOPS hybrid in-memory computing based always-on image processing for neuromorphic vision sensors," *IEEE J. Solid-State Circuits*, vol. 58, no. 3, pp. 589–599, Mar. 2023, doi: 10.1109/JSSC.2022.3218573.

[88] G. R. Nair, P. S. Nalla, G. Krishnan, Anupreetham, J. Oh, A. Hassan, I. Yeo, K. Kasichainula, M. Seok, J.-S. Seo, and Y. Cao, "3-D in-sensor computing for real-time DVS data compression: 65 nm hardware-algorithm co-design," *IEEE Solid-State Circuits Lett.*, vol. 7, pp. 119–122, 2024, doi: 10.1109/LSSC.2024.3375110.

[89] V. Joshi and J. S. Rani, "In-memory-computing (IMC) technique in local difference decision block of an on-board satellite hyperspectral data compression algorithm," in *Proc. IEEE 66th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Tempe, AZ, USA, Aug. 2023, pp. 536–540, doi: 10.1109/MWSCAS57524.2023.10406095.

[90] S. Kavitha and B. S. Reniwal, "In-memory encryption using XOR-based feistel cipher in SRAM array," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Singapore, Singapore, May 2024, pp. 1–5, doi: 10.1109/ISCAS58744.2024.10558309.

[91] D. Reis, H. Geng, M. Niemier, and X. S. Hu, "IMCRYPTO: An in-memory computing fabric for AES encryption and decryption," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 5, pp. 553–565, May 2022, doi: 10.1109/TVLSI.2022.3157270.

[92] S. Xie, M. Yang, S. A. Lanham, Y. Wang, M. Wang, S. Oruganti, and J. P. Kulkarni, "Snap-SAT: A one-shot energy-performance-aware all-digital compute-in-memory solver for large-scale hard Boolean satisfiability problems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 420–422, doi: 10.1109/ISSCC42615.2023.10067380.

[93] D. Kim, N. M. Rahman, and S. Mukhopadhyay, "A 32.5 mW mixed-signal processing-in-memory-based k-SAT solver in 65 nm CMOS with 74.0% solvability for 30-variable 126-clause 3-SAT problems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 28–30, doi: 10.1109/ISSCC42615.2023.10067570.

[94] C. Shim, J. Bae, and B. Kim, "30.3 VIP-sat: A Boolean satisfiability solver featuring 5×12 variable in-memory processing elements with 98% solvability for 50-variables 218-clauses 3-SAT problems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 486–488, doi: 10.1109/ISSCC49657.2024.10454397.

[95] P.-C. Wu, J.-W. Su, L.-Y. Hong, J.-S. Ren, C.-H. Chien, H.-Y. Chen, C.-E. Ke, H.-M. Hsiao, S.-H. Li, S.-S. Sheu, W.-C. Lo, S.-C. Chang, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 22 nm 832 Kb hybrid-domain floating-point SRAM in-memory-compute macro with 16.2–70.2TFLOPS/W for high-accuracy AI-edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2023, pp. 126–128, doi: 10.1109/ISSCC42615.2023.10067527.

[96] W.-S. Khwa, P.-C. Wu, J.-J. Wu, J.-W. Su, H.-Y. Chen, Z.-E. Ke, T.-C. Chiu, J.-M. Hsu, C.-Y. Cheng, Y.-C. Chen, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 16 nm 96Kb integer/floating-point dual-mode-gain-cell-computing-in-memory macro achieving 73.3–163.3TOPS/W and 33.2–91.2TFLOPS/W for AI-edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 568–570, doi: 10.1109/ISSCC49657.2024.10454447.

[97] Y. Wang, X. Yang, Y. Qin, Z. Zhao, R. Guo, Z. Yue, H. Han, S. Wei, Y. Hu, and S. Yin, "A 28 nm 83.23TFLOPS/W POSIT-based compute-in-memory macro for high-accuracy AI applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 566–568, doi: 10.1109/ISSCC49657.2024.10454567.

[98] T.-H. Wen, H.-H. Hsu, W.-S. Khwa, W.-H. Huang, Z.-E. Ke, Y.-H. Chin, H.-J. Wen, Y.-C. Chang, W.-T. Hsu, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, S.-H. Teng, C.-C. Chou, Y.-D. Chih, T.-Y.-J. Chang, and M.-F. Chang, "A 22 nm 16 Mb floating-point ReRAM compute-in-memory macro with 31.2TFLOPS/W for AI edge devices," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 580–582, doi: 10.1109/ISSCC49657.2024.10454468.

[99] Y. Yuan, Y. Yang, X. Wang, X. Li, C. Ma, Q. Chen, M. Tang, X. Wei, Z. Hou, J. Zhu, H. Wu, Q. Ren, G. Xing, P.-I. Mak, and F. Zhang, "A 28 nm 72.12TFLOPS/W hybrid-domain outer-product based floating-point SRAM computing-in-memory macro with logarithm bit-width residual ADC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 576–578, doi: 10.1109/ISSCC49657.2024.10454313.

[100] A. Guo et al., "A 22 nm 64kb lightning-like hybrid computing-in-memory macro with a compressed adder tree and analog-storage quantizers for transformer and CNNs," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 570–572, doi: 10.1109/ISSCC49657.2024.10454278.

[101] Y. Liu, Y. Ma, N. Shang, T. Zhao, P. Chen, M. Wu, J. Ru, T. Jia, L. Ye, Z. Wang, and R. Huang, "A 22 nm 0.26nW/synapse spike-driven spiking neural network processing unit using time-step-first dataflow and sparsity-adaptive in-memory computing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 484–486, doi: 10.1109/ISSCC49657.2024.10454472.

[102] L. Han, P. Huang, Y. Wang, Z. Zhou, Y. Zhang, X. Liu, and J. Kang, "Efficient discrete temporal coding spike-driven in-memory computing macro for deep neural network based on nonvolatile memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 11, pp. 4487–4498, Nov. 2022, doi: 10.1109/TCSI.2022.3194918.

[103] W. Yi, K. Mo, W. Wang, Y. Zhou, Y. Zeng, Z. Yuan, B. Cheng, and B. Pan, "RDCIM: RISC-V supported full-digital computing-in-memory processor with high energy efficiency and low area overhead," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 71, no. 4, pp. 1719–1732, Apr. 2024, doi: 10.1109/TCSI.2024.3350664.

[104] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "DRAMsim3: A cycle-accurate, thermal-capable DRAM simulator," *IEEE Comput. Archit. Lett.*, vol. 19, no. 2, pp. 106–109, Jul. 2020, doi: 10.1109/LCA.2020.2973991.

[105] S. Xu, X. Chen, Y. Wang, Y. Han, X. Qian, and X. Li, "PIMSim: A flexible and detailed processing-in-memory simulator," *IEEE Comput. Archit. Lett.*, vol. 18, no. 1, pp. 6–9, Jan. 2019, doi: 10.1109/LCA.2018.2885752.

[106] A. BanaGozar, K. Vadivel, S. Stuijk, H. Corporaal, S. Wong, M. A. Lebdeh, J. Yu, and S. Hamdioui, "CIM-SIM: Computation in memory SIMulator," in *Proc. 22nd Int. Workshop Softw. Compil. Embedded Syst.*, New York, NY, USA, May 2019, pp. 1–4, doi: 10.1145/3323439.3323989.

[107] R. Guo, Z. Yue, X. Si, H. Li, T. Hu, L. Tang, Y. Wang, H. Sun, L. Liu, M.-F. Chang, Q. Li, S. Wei, and S. Yin, "TT@CIM: A tensor-train in-memory-computing processor using bit-level-sparsity optimization and variable precision quantization," *IEEE J. Solid-State Circuits*, vol. 58, no. 3, pp. 852–866, Mar. 2023, doi: 10.1109/JSSC.2022.3198413.

[108] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017, doi: 10.1109/TSP.2017.2690524.

[109] Y. Wang, Z. Wu, W. Wu, L. Liu, Y. Hu, S. Wei, F. Tu, and S. Yin, "TensorCIM: Digital computing-in-memory tensor processor with multichip-module-based architecture for beyond-NN acceleration," *IEEE J. Solid-State Circuits*, early access, Jun. 13, 2024, doi: 10.1109/JSSC.2024.3406569.

[110] G. Michaelson, "Programming paradigms, Turing completeness and computational thinking," *Art Sci. Eng. Program.*, vol. 4, no. 3, p. 4, Feb. 2020, doi: 10.22152/programming-journal.org/2020/4/4.

[111] S. Chung and H. Siegelmann, "Turing completeness of bounded-precision recurrent neural networks," in *Proc. Neural Inf. Process. Syst.*, vol. 34, Dec. 2021, pp. 28431–28441.

[112] M. R. H. Rashed, S. K. Jha, and R. Ewetz, "Hybrid analog-digital in-memory computing," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Munich, Germany, Nov. 2021, pp. 1–9, doi: 10.1109/ICCAD51958.2021.9643526.

[113] R. Guo, L. Wang, X. Chen, H. Sun, Z. Yue, Y. Qin, H. Han, Y. Wang, F. Tu, S. Wei, Y. Hu, and S. Yin, "A 28 nm 74.34TFLOPS/W BF16 heterogenous CIM-based accelerator exploiting denoising-similarity for diffusion models," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2024, pp. 362–364, doi: 10.1109/isscc49657.2024.10454308.

[114] S. Kim and H.-J. Yoo, "An overview of computing-in-memory circuits with DRAM and NVM," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 71, no. 3, pp. 1626–1631, Mar. 2024, doi: 10.1109/TCSII.2023.3333851.

[115] H. Zhang, J. Liu, J. Bai, S. Li, L. Luo, S. Wei, J. Wu, and W. Kang, "HD-CIM: Hybrid-device computing-in-memory structure based on MRAM and SRAM to reduce weight loading energy of neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 11, pp. 4465–4474, Nov. 2022, doi: 10.1109/TCSI.2022.3199440.

[116] S. Koshino, C. Matsui, and K. Takeuchi, "Computation-in-memory with hybrid integration of non-volatile memory & SRAM for reservoir computing," in *Proc. IEEE Silicon Nanoelectron. Workshop (SNW)*, Honolulu, HI, USA, Jun. 2022, pp. 1–2, doi: 10.1109/SNW56633.2022.9889042.

[117] W. Li, L. Wang, Z. Li, W. Ye, Z. Zhou, H. Zhou, H. Gao, J. Yue, H. Hu, F. Liu, Q. Luo, and C. Dou, "A 2T P-channel logic flash cell for reconfigurable interconnection in chiplet-based computing-in-memory accelerators," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Singapore, Singapore, May 2024, pp. 1–4, doi: 10.1109/iscas58744.2024.10558443.

[118] Z. Dai, F. Xiang, X. Fu, Y. He, W. Sun, Y. Liu, G. Yang, F. Zhang, J. Yue, and L. Li, "A multichiplet computing-in-memory architecture exploration framework based on various CIM devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 12, pp. 4613–4625, Dec. 2024, doi: 10.1109/TCAD.2024.3416256.

[119] S. Yin, B. Zhang, M. Kim, J. Saikia, S. Kwon, S. Myung, H. Kim, S. J. Kim, M. Seok, and J.-S. Seo, "PIMCA: A 3.4-mb programmable in-memory computing accelerator in 28 nm for on-chip DNN inference," in *Proc. Symp. VLSI Circuits*, Kyoto, Japan, Jun. 2021, pp. 1–2, doi: 10.23919/VLSICircuits52068.2021.9492403.

[120] Y. Huang, L. Zheng, P. Yao, J. Zhao, X. Liao, H. Jin, and J. Xue, "A heterogeneous PIM hardware-software co-design for energy-efficient graph processing," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, New Orleans, LA, USA, May 2020, pp. 684–695, doi: 10.1109/IPDPS47924.2020.00076.

[121] M. P. E. Apolinario, A. K. Kosta, U. Saxena, and K. Roy, "Hardware/software co-design with ADC-less in-memory computing hardware for spiking neural networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 12, no. 1, pp. 35–47, Jan. 2024, doi: 10.1109/TETC.2023.3316121.

**KE YU** (Graduate Student Member, IEEE) received the B.S. and M.S. degrees from the Department of Electronic and Electrical Engineering, Kyungpook National University (KNU), Daegu, South Korea, in 2021 and 2023, respectively, where he is currently pursuing the Ph.D. degree with the Intelligent Semiconductor Laboratory.

Since 2020, he has been involved in several research projects on circuits and systems design with Samsung Electronics and the National Research Foundation of Korea (NRF). His research interests include the design and analysis of VLSI, FPGA accelerator, computing-in-memory (CIM) based architecture, and system development.

**SUNMEAN KIM** (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea, in 2016, 2018, and 2021, respectively.

From 2021 to 2023, he was a Postdoctoral Researcher with the Institute of Artificial Intelligence, Pohang University of Science and Technology (POSTECH), Pohang, South Korea. Since 2023, he has been an Assistant Professor with the School of Electronics Engineering, Kyungpook National University (KNU), Daegu, South Korea. His current research interests include integrated circuits and systems design for emerging technologies.

**JUN RIM CHOI** (Member, IEEE) was born in Seoul, South Korea, in January 1964. He received the B.S. degree from Yonsei University, Seoul, in 1986, the M.S. degree from Cornell University, Ithaca, NY, USA, in 1988, and the Ph.D. degree from the University of Minnesota, Twin Cities, in 1991, all in Electrical and Electronic Engineering Department.

From 1991 to 1993, he was a member of Technical Staff with the GoldStar Central Research Laboratory, Seoul, where he worked on the design and fabrication of integrated sensors and actuators. In 1994, he joined the LG Electronics Research Center, Seoul, where he has been a Senior Research Engineer on the design of ASIC library and advanced digital circuits. Since 1997, he has been a Professor with Kyungpook National University, Daegu, South Korea. He was the Director of Kyungpook National University's IC Design Education Center (IDEC) Campus, from 2000 to 2016. His research interests include system on chip (SoC), wireless power transfer (WPT), smart sensor, and ASIC library development.

Dr. Choi has served as the Geneal Chair of the IEEE International SoC Conference (ISOCC), in 2014.