Processing-in-Memory Technology for Machine Learning: From Basic to ASIC

Brady Taylor *Graduate Student Member, IEEE*, Qilin Zheng *Graduate Student Member, IEEE*, Ziru Li, *Graduate Student Member, IEEE*, Shiyu Li, *Graduate Student Member, IEEE*, and Yiran Chen, *Fellow, IEEE*

***Abstract*—Due to the need for computing models that can process large quantities of data efficiently and with high throughput in many state-of-the-art machine learning algorithms, the processing-in-memory (PIM) paradigm is emerging as a potential replacement for standard digital architectures on these workloads. In this tutorial, we review the progress of PIM technology in recent years, at both the circuit and architecture level. We further present an analysis of when and how PIM technology surpasses the performance of conventional architectures. Finally, we outline our vision for the future of PIM technology.**

***Index Terms*—Processing-in-memory, machine learning accelerator, analogcomputation, dataflow optimization.**

I. Introduction

PROCESSING today's state-of-the-art machine learning (ML) algorithms with high throughput in low-power and resource constrained environments requires innovation at both the circuit level and the architecture level. At the architecture level, the largest optimization space is the substantial amount of data movement (including both feature maps and model weights) between memory and processing elements (PEs). At the circuit level, the largest optimization space is PE computational density. Processing-in-memory (PIM) technology capitalizes on both of these optimization spaces by executing operations on data directly in high-density memory to reduce data transfer costs and employ the large spatial parallelism for higher throughput. In recent years, PIM technology has become an increasingly attractive technology for providing an energy-efficient platform for processing ML algorithms such as deep neural networks (DNN), convolutional neural networks (CNN), support vector machines (SVM), and so on. The design of PIM-based systems is versatile and covers many memory technologies, computational mechanisms, and peripheral circuitry. In this tutorial brief, we will present key techniques for integrating each of these key facets into high performance systems. In Section II, we will explore the operation of analog PIM technology for both emerging and conventional memories. Section III will discuss techniques for digital PIM technology, while Section IV reviews PIM architecture design. Finally, Section V will quantitatively analyze PIM performance and Section VI will present potential research directions.

Manuscript received January 31, 2022; revised April 5, 2022; accepted April 12, 2022. Date of publication April 19, 2022; date of current version May 27, 2022. This work was partially supported by the Natural Science Foundation under Grant 1955246, Grant 1910299, and Grant 1725456; and in part by the Army Research Office under Grant W911NF-19-2-0107. This brief was recommended by Associate Editor Y. Ha. *(Corresponding author: Brady Taylor.)*

The authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27707 USA (e-mail: brady.g.taylor@duke.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2022.3168404.

Digital Object Identifier 10.1109/TCSII.2022.3168404

II. Analog PIM Technology

Analog PIM is characterized by the use of a memory technology to execute multiply-accumulate (MAC) operations by using the values in memory to directly modulate analog input signals into weighted analog output signals [1]. MAC operations are fundamentally multi-row operations that condense the accessing of ML models into single read operations for exceptionally improved energy efficiency and computational density. However, these kinds of multi-row operations have implications on factors such as data precision, sensing margin, and circuit complexities. Furthermore, the nature of this operation varies according to memory technology, weight encoding, input encoding, and output decoding scheme. Input data can take the form of analog voltages, modulated pulse-widths or voltage spikes, while output signals can be converted into digital values or spiking waveforms. The most commonly used memory technologies in analog PIM include RRAM, MRAM, and other emerging resistive memories, but an increasing number of analog SRAM-enabled PIM works are being developed. Weight encoding is often based on memory type and desired precision (i.e., SRAM cells can store single bits while RRAM cells can store one or more bits). Figure 1 features a breakdown of these analog PIM system components. In the following section, we separate analog PIM systems into RRAM/MRAM and SRAM systems, describing their key differences and benefits.

A. RRAM/MRAM PIM Operation

Resistive and magnetoresistive random-access memories (RRAM/MRAM) store weights as conductances in a crossbar array. Voltage signals are driven to each row of the array, and the conductance of each cell produces a proportional current that is accumulated along the column. The resulting current represents a dot-product of the column's weights and voltage signals [2]. Due to the crossbar structure, the voltages across each column of the array allow many dot-products to be executed in parallel. Executing massively parallel dot-products

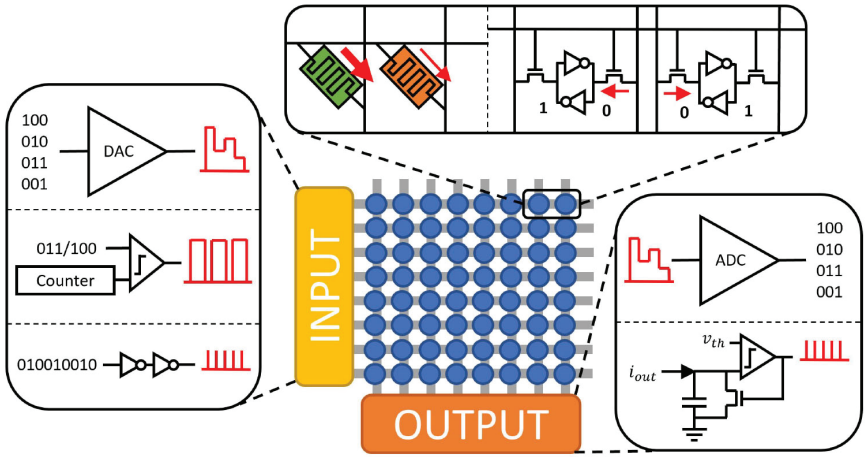


Fig. 1. Breakdown of analog PIM circuit and system components, including various memory technologies, input encoding, and output decoding schemes.

with a simple read operation contributes to the incredible throughput and energy efficiency of resistive PIM.

Encoding weights as conductances can take several forms: multi-level, multi-cell, or a mixture of the two. Although resistive memories can adopt different resistive states for multi-level weights, the precision of a single device is limited by the margin of error for achieving a desired state [3]. Furthermore, incorporating a transistor into each cell can add a control "knob" that assists in high-precision tuning of cells [2], [4]. However, even for high-precision resistive memories, the desired weight precision is sometimes higher. As the number of states achieved by a device and the number of devices in a column increase, the sensing margin decreases, requiring larger and more powerful output circuits. Therefore, many systems use multiple cells spread over several columns to represent a single weight. Columns with larger sensing margins can achieve higher precision when results from each column are shifted and added to produce a final result. For best sensing margins, it is common for each column to have devices representing one or two bits only [5], [6]. Furthermore, as conductances are positive only, negative weights are commonly represented by the difference of two arrays [7].

Another key distinction between systems is the nature of the input signal. A few widely used cases include digital-to-analog (DAC), pulse-width, binary, and spiking-based inputs. DAC-based inputs are the most straight-forward [8], [9], but higher precision DACs require high area and power consumption and have decreased sensing margins. One solution is to drive a single DC voltage and vary the pulse-width, such that the MAC result is measured in accumulated charge and proportional to conductance multiplied by pulse duration [10]. High-precision inputs can also be split into sequential voltage signals, where outputs are buffered for each set of input bits. One example is binary inputs, including bit-wise serial MACs [5] or binary neural networks [11], [12], [13]. Binary inputs are especially area and energy efficient, requiring simple buffers instead of DACs and have larger sensing margins. Spiking neural networks (SNN) are very similar to binary inputs, but information is conveyed through relative timing or frequency of the signal rather than a digital interpretation [14], [15].

How the output current is decoded has a large impact on the precision and efficiency of a resistive PIM system. The most straight-forward method involves analog-to-digital converters (ADC), but these consume large amounts of energy and area [5], [16], [17]. SNNs provide an alternative by converting the current into a spiking voltage signal with the use of an integrate-and-fire circuit (IFC). In an IFC, the current is integrated by a capacitor, which uses a comparator in a feedback loop to emit a spike and discharge the capacitor when a threshold voltage is reached [15], [18], [19], [20].

Despite the benefits of analog PIM, certain nonidealities can affect performance. The voltage drop across a resistive array, or IR-drop, can diminish voltage across cells farthest from the drivers and introduce errors into the dot-products. Solutions for IR-drop include decreasing required array sizes or compensating for IR-drop in training [21]. Sneak path, or inadvertent current paths through an array, can also skew dot-product results and contribute to the perturbation of cells away from their target conductances. Inclusion of selectors in each cell is often used to minimize this inadvertant current [22], [23]. Linear scaling has also been used to counteract sneak path and IR-drop effects on dot-products [2]. Finally, resistive memories suffer from lower endurance compared to conventional technologies, resulting in orders-of-magnitude fewer write cycles before performance drops. To reduce currents and extend device lifetime, weights are trained to have low weights and mapped to underused devices when possible [24].

B. SRAM PIM Operation

Due to issues mentioned in the previous section, more conventional memories, such as static random-access memory (SRAM), are being examined for analog PIM. Most SRAM-based systems execute MAC operations of single-bit inputs multiplied by single-bit weights. While SRAM requires much more area than RRAM or MRAM systems, the benefits of SRAM include improved endurance and reliability.

Analog SRAM-based PIMs, which perform either current-domain computation or charge-domain computation, differ slightly in execution from resistive PIMs. In typical current-domain PIMs, analog voltages are applied to the word-lines (WL) of the SRAM array for nonzero inputs to activate cells. A stored "1" will discharge bit-line bar (BLB), while a stored "0" discharges the bit-line (BL), creating voltage

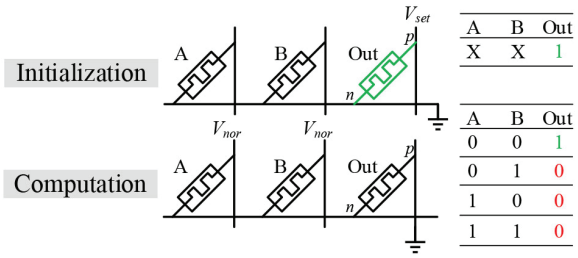
differences [25], [26], [27]. The output circuitry for SRAM-based PIM is designed to maximize sensing margins and collate the results for each column into a final result. In one work, most-significant bit (MSB) cells have larger access transistors to generate larger voltage differences than least-significant bit (LSB) cells to reduce post-processing logic [26]. In some works, switched-capacitor circuits are used to interpret the column outputs, with the MSB column attached to a larger capacitor, decreasing in factors of two down to the LSB column [26], [27]. To increase sensing margins, it has been proposed that columns are split into even and odd cells to reduce parasitics and decrease cells per column [26]. If MAC results are expected to be small, BL header resistors can also be changed to increase sensing margin at the cost of linearity for large MAC results [27].

There also exist SRAM-based PIMs that exploit charge-domain computation [28], [29], [30], which leverage charge sharing of local capacitors within multiple SRAM cells to perform MAC operations. Charge-domain SRAM-based PIMs outperform current-domain counterparts in terms of computation accuracy, but they require additional in-cell capacitors that increase the cell area. Moreover, the implementation of MAC operations with multi-bit weights is also challenging. Another recent solution to the limited sensing margin and computation accuracy of current-domain PIMs is time-domain computation [31], where an inter-cell inverter chain generates time delays controlled by input voltages and weights stored in multiple cells to perform MAC operations. The time delay is then converted into a digital quantity by low-power timing-to-digital converters (TDCs) instead of power-hungry ADCs and DACs, leading to higher energy efficiency.

III. Digital PIM Technology

Analog PIM designs manifest significant improvement in terms of power efficiency and computational efficiency. However, analog circuits are subject to noise and process-voltage-temperature (PVT) variations, and thus result in computational errors. Digital PIM designs provide an alternative solution to leverage the efficient PIM paradigm while maintaining high circuit reliability. In the literature, there are two main forms of digital PIM implementations, i.e., near-memory-computing (NMC) and in-memory-logic. Here, in-memory-logic designs demonstrate more circuit level innovation, and NMC solutions focus on architecture level innovations.

Generally, in-memory-logic designs can be categorized into "read-based logic" and "write-based logic". In read-based logic, simple logic gates are integrated within the memory array. Representative SRAM-based in-memory-logic designs [32], [33] read the stored data from the bit-lines and perform logic computations with bit-serial inputs within the array. Subsequent shift-and-add operations and post-processing are completed outside the array. Compared to analog SRAM-based PIM designs, SRAM-based in-memory-logic designs eliminate DACs/ADCs, and thus save considerable power and peripheral area. Given the same SRAM array size, the SRAM-based in-memory-logic designs cannot compete with analog counterparts in terms of throughput due to its bit-serial computation paradigm. Moreoever, since the direction of SRAM reading is fixed, the patterns of data mapping and processing are also restricted. This degrades the hardware utilization rate and requires carefully reorganizing the data alignment when switching to different layers in DNNs.



A	B	Out
X	X	1

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Fig. 2. An example of RRAM-based in-memory NOR logic.

There also exist nonvolatile memory (NVM) based in-memory-logic designs [34], [35], [36], using binary NVM cells to perform simple logic such as NOR or XOR operations. These designs use mainly write-based logic, which differs from read-based logic in that it involves writing memory cells based on the data stored in other cells. Basic write-based logic operations involve one or two cycles of voltage input and cell programming. For example, Figure 2 depicts the execution of a simple NOR operation implemented by RRAM-based in-memory-logic. In the initialization cycle, the output cell is programmed to the low resistance state (LRS), representing "1" by applying a set voltage V_{set} to its p terminal. In the computation cycle, two input cells, A and B, take in an execution voltage V_{nor}. The voltage on the n terminal of the output cell is high when at least one of the input cells are in LRS, programming the output cell to the high resistance state (HRS), representing "0". Complex logic can be implemented through a combination of these simple operations. Such write-based in-memory-logic designs leverage the dense memory crossbar structure to achieve extremely low area and power consumption. There is no need for additional logic circuits integrated in the memory array compared to read-based in-memory-logic. The drawback of write-based in-memory-logic comes from the large number of cycles required to implement the logic, which prolongs the computation latency [37]. Furthermore, the massive number of write operations leads to degraded endurance of NVM cells.

IV. PIM-Based Architecture Design

Processing-in-memory circuits construct powerful primitives for machine learning applications. However, proper architecture design is required to organize these primitives and provide an interface for programmers. The objective of PIM-based architecture design is to minimize the data movement cost during computation while maintaining normal memory functions when no computation tasks are assigned. Analog PIM-based architectures are an attractive solution thanks to their high computational density and power efficiency. Most analog PIM sub-arrays provide vector-matrix multiplication (VMM) as a primitive. An analog PIM-based accelerator is designed hierarchically, with a network of tiles, each of which can integrate several PIM arrays. The target DNN workload

is partitioned layer-wise and mapped to each tile spatially. An additional buffer, implemented by eDRAM or SRAM, is added into each tile to buffer the intermediate results [5]. Another design methodology is to use the memory arrays to buffer the intermediate results, as proposed by PRIME [8]. The weights are mapped into the PIM-arrays and the inputs are fetched from the input buffer to the PIM-arrays. The output results are stored back to the intermediate data buffer. Optimized data flow is explored within the sub-arrays for different tasks. For example, a layer-wise pipeline is used in [5] to reduce the buffer size overhead. Feature map reuse schemes are explored in [38] to reduce memory accesses for convolutional neural networks. PipeLayer [9] demonstrates a pipelined design to support the training of deep neural network models. ZARA [16] further utilizes the sparsity of generative adversarial networks to reduce the computational cost. A specialized pipeline is proposed in [36] to reduce intermediate data writes in a transformer model.

Other than accelerator design, SRAM-based PIM can also be integrated into modern processors as a complement to the main processing engine for memory-intensive operations. References [39] and [40] demonstrate digital SRAM CIM engines integrated into the processor's cache for logic operations, while [41] further supports DNN inference with bit-serial arithmetic. Since the data stored as a binary value in each cell, only logic operations or bit-serial arithmetic are allowed, which greatly limits the capability of this type of PIM design.

NMC solutions are mostly explored at the architecture level since the custom logic layer provides great flexibility. The key advantage of NMC-based solutions compared to conventional accelerators is that the high internal bandwidth can be effectively exploited. A processing engine is inserted between each channel and the original interface so that each memory channel can operate simultaneously. NMC-based commodity products [42] are also demonstrated providing programmability and high performance.

V. Quantitative Analysis

In this section, we will present a comparison of results for the energy efficiency and throughput among PIM-based accelerators and non-PIM accelerators. We choose a systolic array as the baseline non-PIM accelerator, with spatial dimensions of $[N, N]$, and take a simple convolutional layer with input dimensions of $[I, H, W]$, weight dimensions of $[I, O, K, K]$, and output dimensions of $[O, H, W]$ as a benchmark. We assume that the size of the input, output, and weight memory (PIM memory) is enough to buffer this convolutional layer.

Memory accesses directly affect energy efficiency of a design since memory access energy dominates the whole of energy consumption. The number of weight, input and output memory accesses can be expressed as follows:

$$N_{weight} = O \times I \times K \times K \tag{1}$$

$$N_{input} = \frac{O \times I \times H \times W \times K \times K}{N} \tag{2}$$

$$N_{output} = 2 \times \frac{O \times I \times H \times W \times K \times K}{N} \tag{3}$$

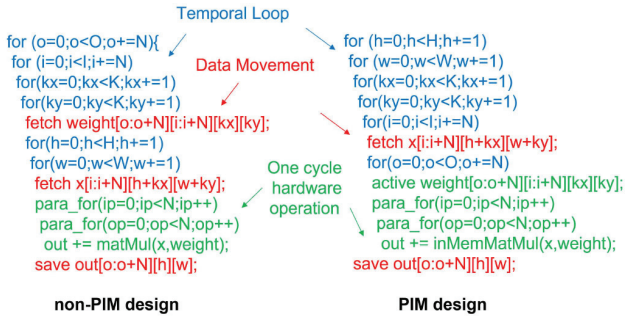


Fig. 3. Pseudo code for a convolutional layer on PIM and non-PIM designs.

Pseudo code for execution of a CNN layer on PIM and non-PIM-based accelerators is shown in Figure 3. Here, we assume the PIM macro has the same spatial parallelism as the baseline non-PIM accelerator. In this case, the number of memory accesses can be expressed as follows.

$$N_{weight} = 0 \tag{4}$$

$$N_{input} = I \times H \times W \times K \times K \tag{5}$$

$$N_{output} = O \times H \times W \tag{6}$$

The elimination of weight transfer in the PIM-based accelerator is clearly observed, as the weights are stored in memory and left undisturbed throughout the computation. The reduction in input memory accesses is attributed to the large memory capacity of PIM-based accelerators compared to register files in PEs of the non-PIM accelerator. Buffering all weights for one layer in memory allows us to reuse the input over all output channels and therefore reorder the temporal O loop below input fetching operations. In terms of output memory accesses, the PIM-based design achieves minimum data movement. Due to the constraint of weight reuse, the temporal H and W loops are below the weight fetching operation in non-PIM based designs. However, these two loops can be moved to the top in a PIM-based design as there are no weight fetching operations.

In terms of throughput, we argue that PIM-based accelerators could not provide significant improvement over non-PIM accelerators. In the throughput optimized non-PIM design, N is usually a relatively large number (256 in TPU [43]), and it is compatible with the layer configurations. Thus, PIM-based designs could not deliver higher parallelism than non-PIM designs. In addition, the clock speed of PIM-based accelerators (50MHz [25] to 300MHz [26]) is usually smaller than non-PIM accelerators (100MHz [44] to 800MHz [43]). Thus, unfortunately, a state-of-the-art PIM macro could not achieve significantly higher throughput than a non-PIM design, even with area-normalized throughput. For example, [45] could provide a 88.6 $GOPS/mm^2$ area-normalized throughput at 28 nm CMOS, while a non-PIM design in [46] could achieve 90.72 $GOPS/mm^2$ at 40 nm CMOS. However, PIM designs can provide significant throughput improvement over non-PIM design if computational accuracy requirements are relaxed. Reference [47] achieves over 12 $TOPS/mm^2$ at 22 nm CMOS by designing a ternary-weight PIM core, which comes at the expense of reduced computational accuracy. Customized ML

models should be used when using this type of PIM macro to build up a system-on-chip (SoC) [48].

VI. Conclusion and Future Research Directions

Due to PIM's inherent dependence on novel and emerging memories as well as machine learning algorithms, continued collaboration between circuit designers, computer scientists, material scientists, and neuroscientists is crucial for the continued success of PIM. This tutorial is unique in its efforts to examine the topic of PIM circuits from various viewpoints: analog vs. digital domains, nonvolatile vs. SRAM memories, and circuit level vs. overall architectural implications. From our analysis from these various perspectives, we believe several obstacles require immediate attention.

Firstly, the training of DNN and ML models, usually executed by servers before deployment to PIM systems, should be supported with much less resources by PIM macros, and in real-time. This could eliminate the need for edge devices to maintain communication with servers for updated models, as well as reduce the increasing dependency on warehouse computing. This further requires software-hardware co-design to minimize the lossiness of deployed ML models and ensure that training techniques optimally utilize the benefits and high parallelism of PIM systems.

Secondly, dual-mode PIM macros and compilers for mapping workloads to them should enable general-purpose workloads and integration with more conventional von Neumann architectures. This will improve the flexibility and accessibility of PIM systems, enabling their use in a broader range of applications for improved overall energy efficiency. Striking a balance between flexibility and performance will be implicit to this brief direction. Making PIM more usable is arguably an important step in garnering support of expanding PIM research.

Finally, continued improvement in nonvolatile memory reliability and 3D integration of technologies at the device and computer-aided design layers of the stack is indispensable for efficient utilization of many of the techniques mentioned in this brief. Reliability and 3D integration are both imperative for increasing memory density and, consequently, improving energy efficiency and performance. Furthermore, development of computer-aided design tools for incorporation of nonvolatile memories can further expand the accessibility of PIM systems.

In summary, we examined key processing-in-memory architectures and circuit techniques that contribute to the high throughput and energy-efficient nature of this emerging paradigm. We further analyzed and compared PIM performance to non-PIM architectures and explained future potential research directions for the field. The emergence of PIM was facilitated by advances in memory technology and the need for a new computing model that can process large quantities of data. We believe, for these reasons and the shrinking resource constraints of intelligent systems, that PIM will be ubiquitous in future systems.

References

[1] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nat. Nanotechnol.*, vol. 8, no. 1, pp. 13–24, 2013.

[2] M. Hu *et al.*, "Memristor-based analog computation and neural network classification with a dot product engine," *Adv. Mater.*, vol. 30, no. 9, 2018, Art. no. 1705914.

[3] B. Li, Y. Shan, M. Hu, Y. Wang, Y. Chen, and H. Yang, "Memristor-based approximated computation," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 2013, pp. 242–247.

[4] E. J. Merced-Grafals, N. Dávila, N. Ge, R. S. Williams, and J. P. Strachan, "Repeatable, accurate, and high speed multi-level programming of memristor 1t1R arrays for power efficient analog computing applications," *Nanotechnology*, vol. 27, no. 36, 2016, Art. no. 365202.

[5] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. 43rd Int. Symp. Comput. Architect.*, 2016, pp. 14–26.

[6] M. A. Zidan *et al.*, "A general memristor-based partial differential equation solver," *Nat. Electron.*, vol. 1, no. 7, pp. 411–420, 2018.

[7] M. Hu, H. Li, Q. Wu, G. S. Rose, and Y. Chen, "Memristor crossbar based hardware realization of BSB recall function," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2012, pp. 1–7.

[8] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Architect. (ISCA)*, 2016, pp. 27–39.

[9] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2017, pp. 541–552.

[10] H. Jiang *et al.*, "Pulse-width modulation based dot-product engine for neuromorphic computing system using memristor crossbar array," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–4.

[11] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on RRAM," in *Proc. 22nd Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2017, pp. 782–787.

[12] X. Sun, S. Yin, X. Peng, R. Liu, J.-S. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *Proc. Design Automat. Test Europe Conf. Exhibit. (DATE)*, 2018, pp. 1423–1428.

[13] L. Song, Y. Wu, X. Qian, H. Li, and Y. Chen, "ReBNN: In-situ acceleration of binarized neural networks in ReRAM using complementary resistive cell," *CCF Trans. High Perform. Comput.*, vol. 1, no. 3, pp. 196–208, 2019.

[14] Z. Li, B. Yan, and H. H. Li, "ReSiPE: ReRAM-based single-spiking processing-in-memory engine," in *Proc. 57th ACM/IEEE Design Automat. Conf. (DAC)*, 2020, pp. 1–6.

[15] C. Liu *et al.*, "A spiking neuromorphic design with resistive crossbar," in *Proc. 52nd ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, 2015, pp. 1–6.

[16] F. Chen, L. Song, H. H. Li, and Y. Chen, "ZARA: A novel zero-free dataflow accelerator for generative adversarial networks in 3D ReRAM," in *Proc. 56th Annu. Design Automat. Conf.*, 2019, p. 133.

[17] Q. Zheng *et al.*, "MobiLattice: A depth-wise DCNN accelerator with hybrid digital/analog nonvolatile processing-in-memory block," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2020, pp. 1–9.

[18] B. Yan *et al.*, "RRAM-based spiking nonvolatile computing-in-memory processing engine with precision-configurable in situ nonlinear activation," in *Proc. Symp. VLSI Technol.*, 2019, pp. T86–T87.

[19] C. Liu *et al.*, "A memristor crossbar based computing engine optimized for high speed and accuracy," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, 2016, pp. 110–115.

[20] B. Yan, A. M. Mahmoud, J. J. Yang, Q. Wu, Y. Chen, and H. H. Li, "A neuromorphic asic design using one-selector-one-memristor crossbar," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2016, pp. 1390–1393.

[21] B. Liu *et al.*, "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2014, pp. 63–70.

[22] J. Zhou, K.-H. Kim, and W. Lu, "Crossbar RRAM arrays: Selector device requirements during read operation," *IEEE Trans. Electron Devices*, vol. 61, no. 5, pp. 1369–1376, May 2014.

[23] Q. Hua *et al.*, "A threshold switching selector based on highly ordered ag nanodots for x-point memory applications," *Adv. Sci.*, vol. 6, no. 10, 2019, Art. no. 1900024.

[24] G. L. Zhang *et al.*, "Reliable and robust RRAM-based neuromorphic computing," in *Proc. Great Lakes Symp. VLSI*, 2020, pp. 33–38.

[25] Y.-C. Chiu *et al.*, "A 4-kb 1-to-8-bit configurable 6t SRAM-based computation-in-memory unit-macro for CNN-based AI edge processors," *IEEE J. Solid-State Circuits*, vol. 55, no. 10, pp. 2790–2801, Oct. 2020.

[26] X. Si *et al.*, "A twin-8t SRAM computation-in-memory unit-macro for multibit CNN-based AI edge processors," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 189–202, Jan. 2020.
[27] X. Si *et al.*, "A local computing cell and 6t SRAM-based computing-in-memory macro with 8-b MAC operation for edge AI chips," *IEEE J. Solid-State Circuits*, vol. 56, no. 9, pp. 2817–2831, Sep. 2021.
[28] A. Biswas and A. P. Chandrakasan, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, Jan. 2019.
[29] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute," *IEEE J. Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, Jun. 2019.
[30] Z. Chen *et al.*, "CAP-RAM: A charge-domain in-memory computing 6T-SRAM for accurate and precision-programmable CNN inference," *IEEE J. Solid-State Circuits*, vol. 56, no. 6, pp. 1924–1935, Jun. 2021.
[31] P.-C. Wu *et al.*, "A 28nm 1Mb time-domain computing-in-memory 6T-SRAM macro with a 6.6 ns latency, 1241GOPS and 37.01 TOPS/W for 8b-MAC operations for edge-AI devices," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 65, 2022, pp. 1–3.
[32] Y.-D. Chih *et al.*, "An 89TOPS/W and 16.3 TOPS/mm 2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 64, 2021, pp. 252–254.
[33] H. Kim, Q. Chen, T. Yoo, T. T.-H. Kim, and B. Kim, "A 1–16B precision reconfigurable digital in-memory computing macro featuring column-MAC architecture and bit-serial computation," in *Proc. ESSCIRC IEEE 45th Eur. Solid State Circuits Conf. (ESSCIRC)*, 2019, pp. 345–348.
[34] S. Kvatinsky *et al.*, "MAGIC–Memristor-aided logic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.
[35] S. Gupta, M. Imani, and T. Rosing, "FELIX: Fast and energy-efficient logic in memory," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2018, pp. 1–7.
[36] X. Yang, B. Yan, H. Li, and Y. Chen, "ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration," in *Proc. 39th Int. Conf. Comput.-Aided Design*, 2020, pp. 1–9.
[37] S. Shirinzadeh, M. Soeken, P.-E. Gaillardon, and R. Drechsler, "Logic synthesis for RRAM-based in-memory computing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1422–1435, Jul. 2017.
[38] Q. Zheng *et al.*, "Lattice: An ADC/DAC-less ReRAM-based processing-in-memory architecture for accelerating deep convolution neural networks," in *Proc. 57th ACM/IEEE Design Automat. Conf. (DAC)*, 2020, pp. 1–6.
[39] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Compute caches," in *Proc. IEEE Int. Symp. High Perform. Comput. Architect. (HPCA)*, 2017, pp. 481–492.
[40] A. Dhar *et al.*, "FReaC cache: Folded-logic reconfigurable computing in the last level cache," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitect. (MICRO)*, 2020, pp. 102–117.
[41] C. Eckert *et al.*, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Architect. (ISCA)*, 2018, pp. 383–396.
[42] S. Lee *et al.*, "Hardware architecture and software stack for pim based on commercial DRAM technology: Industrial product," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Architect. (ISCA)*, 2021, pp. 43–56.
[43] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Architect.*, 2017, pp. 1–12.
[44] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1B-to-16B fully-variable weight bit-precision," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2018, pp. 218–220.
[45] B. Yan *et al.*, "A 1.041-Mb/MM 2 27.38-TOPS/W signed-INT8 dynamic-logic-based ADC-less SRAM compute-in-memory macro in 28nm with reconfigurable bitwise operation for AI and embedded applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 65, 2022, pp. 188–190.
[46] J. Park, S. Lee, and D. Jeon, "A neural network training processor with 8-bit shared exponent bias floating point and multiple-way fused multiply-add trees," *IEEE J. Solid-State Circuits*, vol. 57, no. 3, pp. 965–977, Mar. 2022.
[47] I. A. Papistas *et al.*, "A 22 nm, 1540 TOP/S/W, 12.1 TOP/S/MM 2 in-memory analog matrix-vector-multiplier for DNN acceleration," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, 2021, pp. 1–2.
[48] K. Ueyoshi *et al.*, "DIANA: An end-to-end energy-efficient digital and analog hybrid neural network SoC," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, vol. 65, 2022, pp. 1–3.