IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 14 September 2023, accepted 6 October 2023, date of publication 13 October 2023,
date of current version 18 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3324375

RESEARCH ARTICLE

Framework for In-Memory Computing Based on Memristor and Memcapacitor for On-Chip Training

ANKUR SINGH, (Student Member, IEEE), AND BYUNG-GEUN LEE, (Member, IEEE)
Gwangju Institute of Science and Technology, Gwangju 61005, South Korea

Corresponding author: Byung-Geun Lee (bglee@gist.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant through the Korean Government (MSIT) under Grant 2021R1A2C22013480, and in part by the Nano-Material Technology Development Program through NRF funded by the Ministry of Science and ICT under Grant NRF-2022M3H4A1A01009658.

ABSTRACT Memristive crossbar arrays have gained considerable attention from researchers to perform analog in-memory vector-matrix multiplications in machine learning accelerators with low power and constant computational time. This work introduces a comprehensive framework for co-designing the software and hardware for deep neural networks (DNN) based on memristive and memcapacitive crossbars while considering various non-idealities. The model takes into account device-level factors, including conductance variation, cycle-to-cycle variation, device-to-device variation, peripheral circuits for error/weight gradient computation, and high tolerance. The overall neural network performance is thoroughly assessed by integrating these elements into a unified DNN training process. The proposed framework is implemented using a hybrid approach with Python and PyTorch. Performance evaluation was conducted using a simplified 8-layer VGG network on a measured 128×128 array with weight resolution. Remarkably, the memristive and memcapacitive crossbar arrays achieved outstanding training accuracies of 90.02% and 91.03%, respectively, for the CIFAR-10 dataset. Additionally, detailed hardware estimation for both mem-elements devices is provided, enabling meaningful comparisons with prior works.

INDEX TERMS Compute-in-memory, memristor, TIO_X, memcapacitor, deep neural network, neuromorphic system.

I. INTRODUCTION

Memelements [1], [2], [3] emerge as a highly promising class of devices, notably exhibiting exceptional performance when configured in a crossbar structure. The utilization of mem-elements in crossbars brings about significant improvements in the efficiency of vector-matrix multiplication (VMM) by facilitating parallel products and summations of currents flowing through the devices. Particularly in the context of convolutional neural networks (CNN), which heavily rely on extensive matrix operations during both training and inference, a VMM architecture based on mem-elements proves to be highly advantageous.

The integration of in-memory computing (IMC) architecture and the ability to tune analog memductance in mem-elements further empowers power-efficient VMM and training, allowing for the realization of a highly integrated memory architecture. As a result, a myriad of CNN hardware designs featuring mem-elements-based VMM accelerators have been proposed [4], [5], [6], and their efficacy has been impressively demonstrated.

The inherent challenge of resistive crossbars lies in their limited ability to accurately approximate the essential VMM required for optimal performance. For instance, in the process of converting digital inputs into voltages and applying them to the crossbar's rows (programmed with weights as conductances), the ensuing column currents are digitized to yield digital outputs. However, a plethora of device and circuit level nonidealities, such as driver resistance, sensing resistance, sneak paths, interconnect parasitics, analog-to-digital converter (ADC) and digital-to-analog converter (DAC) nonlinearity, stochastic write operations, and process variations, can introduce errors into the computed VMM [7], [8], [9], [10], [11]. The presence of these imperfections can

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Donato Marino.

potentially impair the accuracy of a deep neural network (DNN) implemented on a resistive crossbar system. While DNN do exhibit some level of tolerance to calculation imprecision [12], [13], [14] it is vital to acknowledge the limitations of this resilience. Thus, a thorough assessment of mem-elements crossbar nonidealities at the application level becomes imperative to determine their suitability as foundational components for DNN hardware.

Extensive research has been conducted on the influence of memristor device non-idealities on neural network performance. Existing modeling tools such as MNSIM [15] have mainly emphasized the evaluation of system area and power, potentially neglecting device variations and simplifying non-idealities calculations. On the other hand, CrossSim [16] considers more non-idealities from memristive devices but overlooks non-idealities at the array and circuit levels. Furthermore, its simulations are confined to a simple perceptron using a small dataset, MNIST [17]. The IBM Analog Hardware Acceleration Kit was introduced in [18], incorporating memristive devices into memory computing and PyTorch software simulation. However, the IBM toolkit [18] does not assess the hardware chip's runtime, latency, and power performance. While [19] utilized certain hardware constraints for software simulation, it remained quite limited in scope. An open-source framework for memristive deep learning systems is introduced in [20] and [21], addressing device-to-device variability and failures. Additionally, [22] provides a comprehensive overview of various simulation frameworks, offering detailed comparisons and insights into future modeling and simulation strategies and approaches. In [23], the introduction of capacitor-based synaptic devices is accompanied by experimental validation of their VMM function. Similarly, [24] focuses on the advancement of non-volatile capacitive devices through the utilization of ferroelectric HZO, culminating in the construction of a crossbar array tailored for in-memory computing applications. The prominence of capacitor-based IMC SRAM [25] lies in its remarkable energy efficiency within deep convolutional neural networks (DCNN), albeit with a reliance on SRAM for computation in addition to capacitive devices. Despite the limited body of research in memcapacitive-based IMC, noteworthy strides have been taken in the realm of software-hardware design. This field, however, is not void of progress.

Currently available tools do not possess the necessary capability to accurately model nonuniform conductance and capacitance distributions in memristive and memcapacitive crossbar arrays. Additionally, they often overlook or over-simplify array level non-idealities, such as line resistance and sneak path effects, in VMM simulations due to the computational complexity involved. This work explores nonideal properties crucial for in-situ training accuracy, including nonlinearity, asymmetry, device-to-device, and cycle-to-cycle variations. Consequently, there is a compelling need for a fast and accurate modeling and simulation tool that comprehensively evaluates neural network performance on memristor and memcapacitor crossbars. VMM accelerators leverage supplementary peripheral circuits for error calculation and weight gradient computation during on-chip training.

In this research, we extended the NeuroSim framework to support the evaluation of on-chip training performance in compute-in-memory (IMC) accelerators. The framework is implemented using a hybrid approach with Python and PyTorch CUDA together, incorporating an accurate VMM computation core [26]. The performance of our proposed framework was validated by implementing a simplified 8-layer VGG network with a 128×128 TiO_X-based memristive and a Silicon (Si)-based memcapacitive crossbar array, both derived from hardware data. The results demonstrated an impressive 90.02% and 91.03% inference accuracy with memristive and memcapacitive crossbar array on the CIFAR-10 dataset. Our proposed method offers a potential solution for holistically evaluating neural network performance and highlights the feasibility of utilizing TiOx based memristive and Si based memcapacitive crossbars for DNN applications.

The article is structured as follows: In Section II, the NeuroSim framework structure for IMC based on mem-elements is introduced, along with detailed explanations of the architectures supporting feedforward and back-propagation computation in deep CNN. In Section III, the impact of non-idealities, sneak paths, and information about peripheral circuits is presented. Section IV discusses the benchmark and hardware results of IMC accelerators for on-chip training, providing a detailed comparison with previous works. Finally, Section V concludes the paper.

II. PROPOSED FRAMEWORK DESIGN

A. IMC MAPPING ALGORITHM

In this approach, the weights of each kernel are intelligently divided into sub-matrices based on their spatial locations, resulting in K×K sub-matrices with a size of D×N. This division leads to a total weight matrix size which is kernel of K×K×D×N, as shown in Fig. 1. Simultaneously, the input data assigned to various spatial locations within each kernel is routed to the corresponding sub-matrices. Through parallel computation, partial sums are derived from these sub-matrices. These partial sums are then efficiently aggregated using an adder tree. By doing so, a processing element (PE) is defined as a group of sub-arrays equipped with essential input and output buffers, along with accumulation modules. The kernels are divided into multiple PEs based on their spatial locations, allowing the input data to be assigned accordingly. This strategic division of kernels and input data enables the reusability of input data among PEs, removing the need to revisit upper-level buffers. Consequently, a direct transfer of input data between PEs is facilitated, streamlining the processing flow, and optimizing computational efficiency.

The chip hierarchy is organized into several tiles, each of which houses processing elements (PEs) along with synaptic sub-arrays, accumulation modules, and output buffers. The transfer of inputs/activations from one memory array to another is facilitated by interconnects within each tile. In terms of the assumed interconnect topology, using an H-tree structure for routing within each hierarchy implies

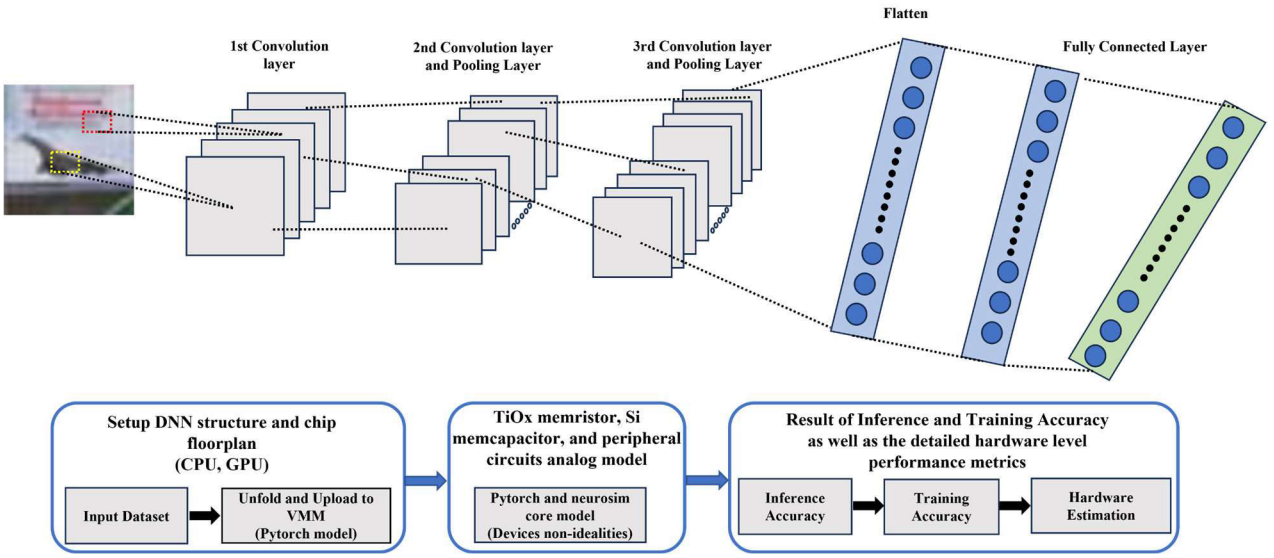

FIGURE 1. The architecture of the convolutional layer and fully connected layer integrated with memristor and memcapacitor crossbars, accompanied by the flowchart of the proposed framework.

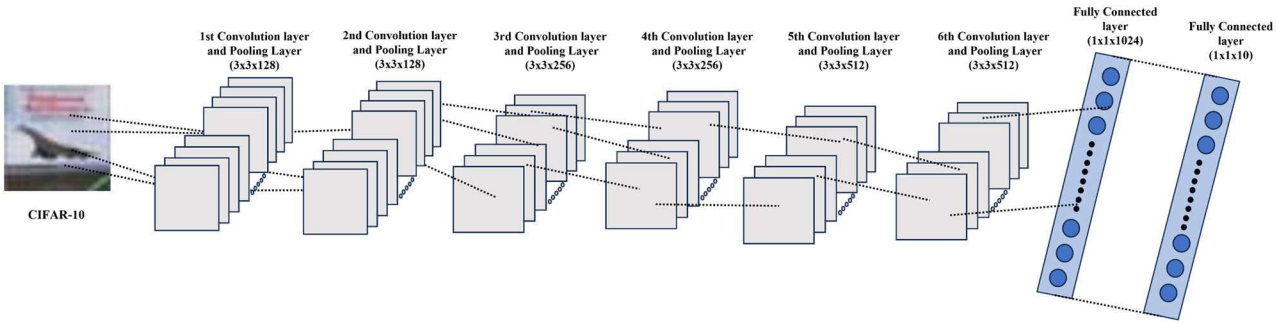

FIGURE 2. The simplified 8-layer VGG network comprises 6 convolutional layers and 2 fully connected layers.

that the interconnections within individual tiles adhere to an H-tree topology. Each layer functions as an individual pipeline stage, and the system clock cycle for the pipeline is determined by the longest latency observed among all the layers. This setup implies that the layers are sequentially processed in a pipelined manner, where the output of one layer serves as the input to the subsequent layer. The framework employs an off-chip offloading model where a portion of the neural network layers is loaded into on-chip memory arrays while the remaining layers are stored in off-chip memory. Offloading entails transferring these layers between on-chip and off-chip memory, resulting in potential performance and power overheads. We have yet to develop a comprehensive analysis of the performance and power overheads associated with offloading, but we are actively working on it to obtain all the necessary details. Nevertheless, the offloading process would introduce additional latency and energy consumption due to the data transfer between on-chip and off-chip memory.

NeuroSim efficiently calculates the weight-matrix size for each layer in the pre-defined network structure using the weight mapping method. The process of iteratively reducing the matrix size involves the following sequence of actions: Initially, the tile size is configured to accommodate the largest weight matrix among all layers. Subsequently, the framework calculates memory utilization by dividing the memory mapped by synaptic weights by the total chip memory. The tile size is then gradually decreased while monitoring its impact on memory utilization. The aim is to achieve optimal memory utilization. This stepwise reduction in tile size contributes to refining memory allocation for improved efficiency. The weights are programmed using the conductance of the memory devices. When input vectors are encoded using read voltage signals, the weighted sum operation is performed in parallel, resulting in currents at the end of each column. The read voltage applied at the input of transmission gates passes through the WL, and the parallel readout of weighted sums occurs through the BL. In cases where input vectors are larger than 1 bit, encoding necessitates multiple clock cycles. The network employs a unit cell arrangement, and for encoding the inputs, 8 bits are used. This choice is influenced by the nature of the CIFAR-10 dataset used in the experiments, which comprises $32 \times 32 \times 3 = 3072$ input features. Utilizing lower bit resolutions for encoding inputs

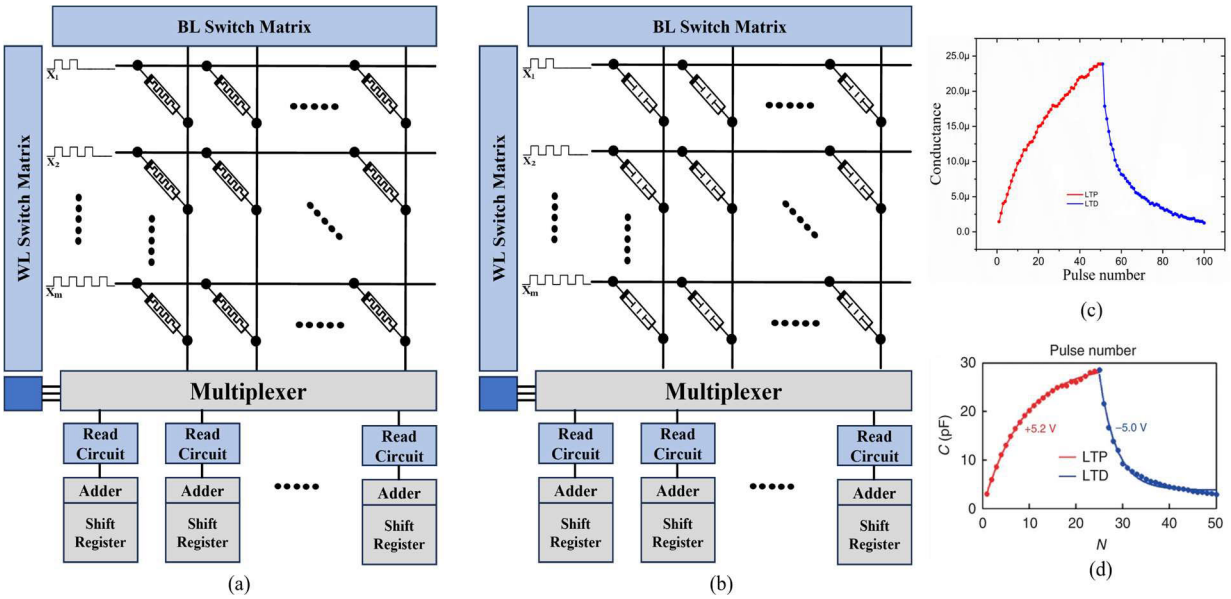

FIGURE 3. Crossbar structure defined in the simulator using (a) memristor device, and (b) memcapacitor devices. Measured memductance data of the (c) memristor device with TiOx material, (d) memcapacitor device with Si material.

would likely result in a degradation of network accuracy. In cases where negative inputs are encountered, they are encoded using the two's complement representation. In this representation, a negative value is represented by taking the complement of its positive counterpart and adding 1. Subsequently, the corresponding output is decoded using the same two's complement representation method.

Fig. 3 (a) and (b) depicts a typical design of a memristive and memcapacitive crossbar array utilized for realizing VMM. The design comprises a 2-D array of synaptic devices, digital-to-analog converters (DACs), analog-to-digital converters (ADCs), and write peripheral circuitry. As proposed in the framework [27], the NeuroSim core is enveloped by Python and PyTorch, allowing for the facilitation of flexible network topologies. The model used is 8 layer VGG network for CIFAR-10. However, the framework also supports larger models such as ResNet, AlexNet, GoogleNet, or users have the option to define arbitrary CNN topologies.

B. MEMRISTOR AND MEMCAPACITOR BASED VMM ACCELERATOR

Fig. 3 (a) and (b) depicts a typical design of a memristive and memcapacitive crossbar array utilized for realizing VMM. This framework comprises two parts: one implemented in Python and the other in PyTorch CUDA. For the evaluation of metrics such as nonlinearity, asymmetry, device-to-device variation, cycle-to-cycle variation, IMC area, latency, and energy. To assess the area, latency, dynamic energy, and leakage associated with interconnects, we assume that routing among modules within each hierarchy follows an H-tree structure. The latency and energy breakdown analysis reveals that, due to substantial on-chip data transfer, the primary bottlenecks are buffer latency and DRAM energy consumption. The estimated training dynamic energy per epoch amounts to 108.36 J, while the training latency per epoch is calculated at 104.31 sec. Although the paper mentions using mem-elements-based IMC arrays for weight gradient computation, it does not explicitly specify the exact on-chip storage capacity required for storing intermediate results or facilitating routing to multiple arrays. The size of the on-chip storage would be contingent upon factors such as the neural network's dimensions and the nature of the operations being conducted.

We consider six convolutional layers and two fully connected layers, each serving as dedicated computation units designed specifically for weighted sum and weight update operations as shown in Fig. 2. In the forward convolutional layer, the analog weights are initially mapped to memductance, with the line resistance serving as the memductance weight. The input data is then fed from the input layer and travels forward through a series of weighted sum operations and neuron activation functions until reaching the output layer. In the fully-connect layer, a similar process occurs, where the analog weights are first mapped to memductance. The VMM is performed on the input vector with the crossbar array assigned weights. During back-propagation, the error is propagated backward from the output layer to adjust the weights of each layer, minimizing the prediction error.

In the backpropagation step, there are two VMM steps. First, multiplying the weight matrix with gradients. It can be inferred that the second Vector-Matrix Multiplication (VMM) is executed by retrieving the activations from off-chip memory and then multiplying them with the gradients. When computing weight gradients, activations are fetched from off-chip memory and conveyed to on-chip buffers before reaching the weight gradient computation units. This implies that the activations are present in on-chip

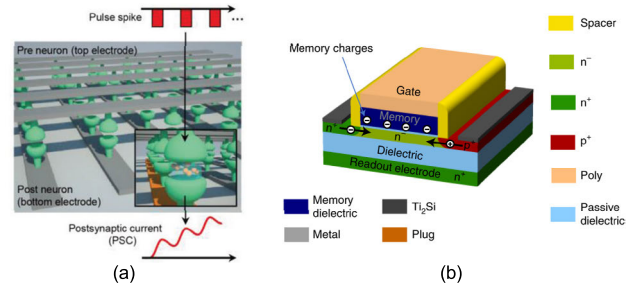

FIGURE 4. Device structure (a) TiOx-based memristor device (b) Si-based memcapacitor device.

buffers and can thus be employed for the second VMM operation.

The TiOx memristive device characteristics encompass operational yield and uniformity, symmetrical analog switching, functional stability, and adjustable learning rates. The TiOx memristor array achieves a remarkable operational yield exceeding 99%, displaying exceptional uniformity in its switching threshold. Its symmetrical analog switching behavior enables both conductance potentiation and depression, essential for implementing synaptic functions in artificial neural networks. Notably, the device exhibits high functional stability, maintaining repeatability over 3000 programming cycles and remaining operational for six months. In essence, the TiOx memristive device showcases reliable symmetrical analog switching traits, operational uniformity, and functional stability, rendering it a promising candidate for effective in situ training within neuromorphic computing systems [1]. Fig. 4 (a) illustrates the device structure and image of a TiOx memristor [1]. This memristor features a crossbar array with nodes based on TiOx. The individual memristor cells are positioned at the intersections of Al electrode lines on a glass substrate, with each Al electrode line having a width of 100 μm.

The characteristics of the Si memcapacitive device, as described in [2], make it well-suited for neuromorphic computing applications. Notable features include a high dynamic range, which enables precise analog signal processing, and low power operation through adiabatic charging, enhancing energy efficiency. The device's scalability down to around 45 nm and its crossbar array architecture further support its integration into compact and energy-efficient neuromorphic systems. This architecture facilitates parallel multiply-accumulate (MAC) operations, ideal for neural network training and pattern recognition tasks. Overall, the Si memcapacitive device's dynamic range, low power operation, scalability, and crossbar array structure position it as a promising choice for energy-efficient neuromorphic computing systems [2]. The Si memcapacitive device comprises a layered structure with a gate electrode, shielding layer, and readout electrode. The gate electrode applies input signals, and the readout electrode reads accumulated charge. The shielding layer between them significantly affects capacitance modulation. The device structure includes a lateral pin junction and electron and hole injection, depicted in Fig. 4 (b).

The expected relationship between weight increase long-term potentiation, (LTP) and weight decrease long-term depression, (LTD) should be linearly dependent on the number of write pulses. However, real-world devices, as described in existing literature, often deviate from this ideal trajectory. In practice, the memductance tends to undergo rapid changes during the initial stages of LTP and LTD, eventually reaching a saturation point, as depicted in Fig. 3 (c) and (d) for TiOx-based memristive and Si-based memcapacitive devices, respectively.

For the TiOx-based memristive VMM [1] and, X_{LTP} and X_{LTD} represent the conductance values for LTP and LTD, respectively and the following are the equations:

$$X_{LTP} = B\left(1 - e^{\left(-\frac{P}{A}\right)}\right) + X_{\min} \tag{1}$$

$$X_{LTD} = -B\left(1 - e^{\left(\frac{P-P_{\max}}{A}\right)}\right) + X_{\max} \tag{2}$$

$$B = X_{\max} - X_{\min} \Big/ 1 - e^{\left(\frac{-P_{\max}}{A}\right)} \tag{3}$$

Similarly, the equations mentioned above can be applied to the Si-based memcapacitive VMM [2]. In this scenario, X_{LTP} and X_{LTD} denote the capacitance values associated with LTP and LTD, respectively.

The parameters X_{max}, X_{min}, and P_{max} are directly obtained from experimental hardware data and correspond to the maximum conductance and capacitance, minimum conductance and capacitance, and the maximum pulse number needed to switch the device between its minimum and maximum conductance states. The parameter A governs the nonlinear behavior of weight update and can be either positive (blue) or negative (red). In Fig. 3 (c) and (d), both LTP and LTD have the same magnitude but opposite signs for the parameter A. B, on the other hand, is a function of A designed to fit the functions within the range of X_{max}, X_{min}, and P_{max}.

III. IMC RESULTS WITH NON-IDEALITIES

This section focuses on investigating nonidealities in memristive and memcapacitive crossbars and analyzing how they affect VMM.

A. CROSSBAR AND DEVICE NONIDEALITIES

Due to fabrication imperfections, non-ideal behaviors are observed in memristor and memcapacitor devices. These include variations in conductance, capacitance, device-to-device (D2D), and cycle-to-cycle (C2C), as well as nonlinearity and programming failure [28]. Consequently, it is crucial to consider nonuniformly distributed levels and conductance variations in the simulation of DNN. Devices are assigned to different levels based on the conductance and capacitance distribution in the crossbar array to assess the degradation of training accuracy under nonideal properties. The nonlinearity and asymmetry model can be represented by

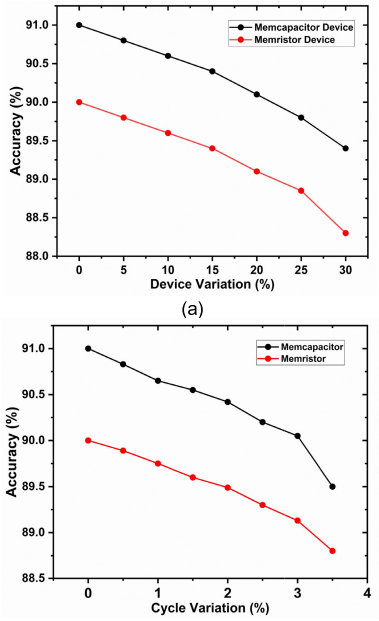

FIGURE 5. Training accuracy comparison of non-idealities (a) different device-to-device memductance variation (b) different cycle-to-cycle variation.

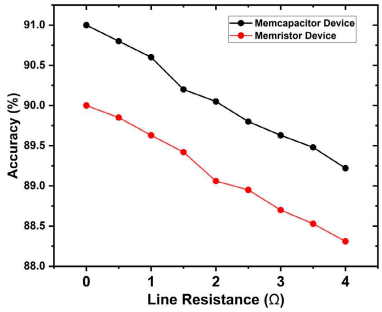

FIGURE 6. The accuracy is influenced by the line resistance and the sneak paths.

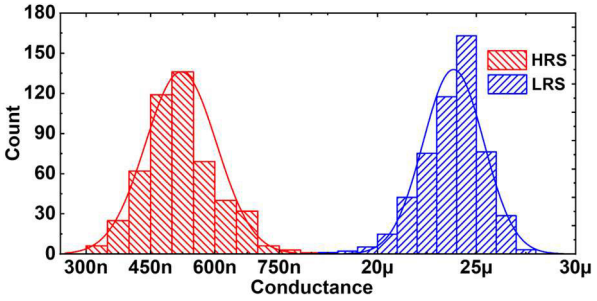

FIGURE 7. Distribution of the TiOx memristor conductance in the HRS and LRS.

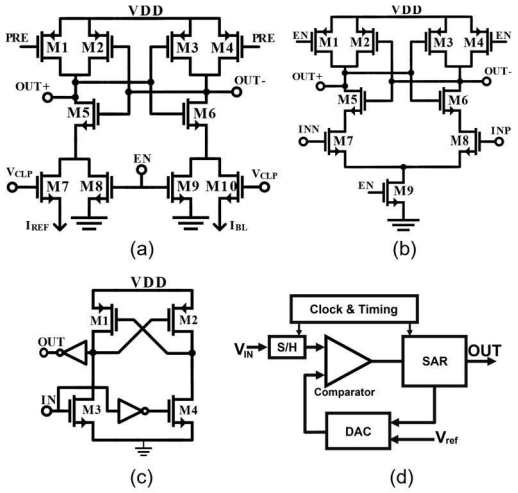

FIGURE 8. Transistor level used in the framework (a) current sense amplifier (CSA), (b) voltage sense amplifier (VSA), (c) level shifter, and (d) successive approximation register (SAR) ADC.

equations 1 to 3. The parameter A determines the degree of nonlinearity in weight update, with a value range of $(0, +\infty)$, where smaller A values indicate a more nonlinear weight update behavior. The device's conductance is programmed from a high resistance state (HRS) to a low resistance state (LRS) and shown in Fig. 7 for the memristive device. It is crucial to emphasize that while the failure mask undergoes updates in each programming cycle, the stuck mask remains unchanged throughout both training and inference. This is due to the inability to fix stuck devices, resulting in their fixed position after array testing.

In the weight update process, D2D variation leads to varying nonlinearities in different synaptic devices. To create a behavior model, we randomly generate the nonlinearity factors for different synaptic weights, using a standard deviation (σ) with respect to the mean nonlinearity value (μ). The results depicted in Fig. 5 (a) highlight the significant impact of device variation, showcasing the remarkable accuracy maintenance achieved by our proposed method using both devices. To investigate the effects of C2C variation, we created a behavior model similar to the one used for device-to-device variation. C2C variation pertains to the variability in conductance change with each programming pulse. As shown in Fig. 5 (b) C2C variation does not degrade the performance of the system. Therefore, we can represent the cycle-to-cycle variation standard deviation (σ) as a percentage of the entire weight range.

In an ideal scenario, currents in resistive crossbars should flow from left to right along the rows and from top to bottom through the columns. Nonidealities, including wire resistances, cause variations in the actual voltage across the memristor and memcapacitor VMM accelerator, resulting in a lower voltage than the theoretical value. This reduction is due to the accumulated voltage drop on the connecting traces and sneak pathways [29]. The presence of line resistance and sneak paths impacts the training accuracy of the model, as depicted from Fig. 6. In our results, we observed that the training accuracy of both VMM approaches is more significantly influenced by the line resistance than by the variations between individual devices and cycles. This suggests that the impact of line resistance plays a more prominent role in affecting the training accuracy compared to the inherent variability between devices and cycles.

B. PERIPHERAL CIRCUITS

The VMM employs various peripheral circuit modules, including a switch matrix, multiplexer, adder, shift register, driver, and ADC [27]. In this framework, these peripheral circuits are designed using transistor parameters directly extracted from the TSMC 22-nm PDK, as shown in Fig. 8, and specifically set in the NeuroSim transistor library. These parameters encompass device W/L, supply voltage (V_{DD}), threshold voltage (V_{TH}), gate and parasitic capacitance, and NMOS/PMOS on/off current density. By utilizing these parameters, the area and intrinsic RC/power model of standard logic gates can be analytically calculated using specific formulas, as discussed in prior works. This enables the estimation of performance metrics for each sub-circuit. The transistor W/L for the ADC, multiplexer, switch matrix, and drivers are predefined based on the required drivability, while the transistor W/L for other logic gates is set at a fixed size. The capacitances at the logic gate level are also improved, and their transistors' sizing is known. This allows for the calculation of $\tau = RC$ and CV_{DD}^2 to estimate module delay and dynamic energy consumption.

Switch matrices are vital components that facilitate fully parallel voltage input to the rows or columns of the array. These matrices are composed of transmission gates connected to all the bit lines (BLs), with their control signals stored in registers. The traditional crossbar word line (WL) decoder has been modified to activate all the WLs, making all the transistors transparent for weighted sum. This enhanced crossbar WL decoder integrates follower circuits into each output row of the conventional decoder. Additionally, a multiplexer is employed to distribute the read periphery circuits among the synaptic array columns, optimizing the utilization of resources as the array cell size is significantly smaller than the size of read periphery circuits. Placing all the read periphery circuits at the edge of the array would not be area efficient. Hence, the multiplexer efficiently addresses this challenge.

We have incorporated quantization noise for the ADC. To address the potential effects of ADC truncation on partial sums, we adopt a nonlinear quantization approach utilizing several quantization edges, each indicative of different levels of ADC precision. These edges are determined based on the distribution pattern of partial sums. Currently, we have not incorporated a read noise model. However, we intend to incorporate such a model in future endeavors.

The CMOS transistor parameters are extracted from TSMC's PDK and integrated into the Framework. Components like current sense amplifiers (CSA), voltage sense amplifiers (VSA), level shifters, and switch matrices are realized using these CMOS parameters. However, achieving the same training accuracy in the hardware implementation may be challenging due to factors like fabrication mismatches, ambient noise, and other variables that can impact system performance.

The current sense amplifier (CSA) as shown in Fig. 8 (a) serves to amplify and convert small current signals into voltage signals, a critical function in precise analog-to-digital conversions within flash-ADCs. Fig. 8 (b) shows the voltage sense amplifier (VSA) plays a pivotal role by amplifying and converting minute voltage signals into digital outputs, essential for accurate conversion of analog voltages in ADCs, particularly in flash-ADCs. Additionally, a level shifter functions as a peripheral module facilitating translation of signal voltage levels across different logic domains as shown in Fig. 8 (c). This ensures seamless communication and signal compatibility among various parts of integrated circuits operating at diverse voltage levels. Lastly, in Fig. 8 (d) the successive approximation register (SAR) ADC operates by employing a binary search algorithm to determine the digital representation of analog input signals. Through iterative adjustment of the digital output, the SAR ADC converges to the closest digital representation of the input signal, making it suitable for various applications due to its moderate conversion speed and relatively low power consumption. The active blocks in the design operated at 1.1 V for VDD. Level shifters were incorporated into the design, particularly for the WL (Word Line), BL (Bit Line), and SL (Sense Line) signals within the crossbar array. These level shifters were employed to convert the voltage levels of these signals to the necessary levels essential for the correct functioning of the crossbar array.

To extract and process partial sums for subsequent logic modules, a group of flash-ADCs with multilevel successive approximation (S/A) using varying references is employed at the end of the synaptic lines (SLs) to produce digital outputs. In the simulator, a conventional current-sense-amplifier (CSA) based on transistor is utilized as the unit circuit module for building the multilevel S/A, as depicted in Fig. 8 (a). At the bottom of the synaptic core, an adder and shift register pair are utilized to execute shift and addition operations on the weighted sum result during each input vector bit cycle, resulting in the final weighted sum. The bit-width of the adder and shift register may need to be extended based on the precision requirements of the input vector.

TABLE 1. DNN neurosim experiment hardware configurations.

Configuration	Value
Operation mode	Conventional Parallel
ADC Precision	5-bit
Crossbar Size	128 x 128
Clock Frequency	10^9
Temperature	300 K
Roff (Ω)	25 x 10^6
Ron (Ω)	10^4
Coff (C)	30 x 10^{-12}
Con (C)	2 x 10^{-12}
Device Type	Memristor, Memcapacitor
Technology	22 nm

IV. HARDWARE RESULTS OF IMC

In this section, we examine detailed analysis for accuracy, area, throughput, and energy efficiency with our Python framework designed to evaluate large-scale memristive and

TABLE 2. Comparison with prior works.

	This Work		[30]	[31]	[32]	[33]	[34]	[35]
Device	Memristor (TiO_X)	Memcapacitor (Si)	Ag:a-Si	Digital ReRAM	EpiRAM	HZO FeFET	PCMO	AlO_x/HfO_2
Network Structure	VGG-8		VGG	BNN-9	VGG-8	VGG-8	VGG-8	VGG-8
Crossbar Size	128 × 128		128 × 128	128 × 128	784 × 300	128 × 128	128 × 128	128 × 128
# of Conductance States	32		97	-	64	32	50	40
ADC precision	5-bit		6-bit	-	6-bit	6-bit	6-bit	6-bit
Weight/ Cell precision	5-bit/ 1-bit		6-bit	-	6-bit	5-bit	5-bit	5-bit
Ron (Ω)/Con (C)	$10^4/2 \times 10^{12}$		50×10^6	50×10^3	81×10^3	240×10^3	23×10^6	16.9×10^6
On/Off Ratio	10	10	12.5	16	50.2	100	6.84	4.43
Device Variation (3σ/μ)	30%	30%	-	12 %	-	-	-	-
Line resistance (Ω)	0.5	0.5	-	0.1	-	-	-	-
Area (mm^2)	29.4	47.1	48.29	0.78	48.59	48.29	48.29	49.88
Memory Utilization (%)	88.59 %	88.59 %	88.59 %	-	88.59 %	88.59 %	88.59 %	88.59 %
Training Accuracy (%)	90.02 %	91.03 %	49.00 %	92 %	85.00 %	91.00 %	56.00 %	37.00 %
Training Throughput (TOPS)	1.51	1.54	0.14	0.792	0.95	1.04	0.03	0.30
Training Energy Efficiency (TOPS/W)	2.10	2.32	2.00	176	2.00	2.01	2.00	1.98
Training Peak Throughput (TOPS)	3.81	3.85	0.16	-	2.68	3.57	0.03	0.38
Training Peak Energy Efficiency (TOPS/W)	18.98	19.11	20.54	-	20.11	20.57	20.50	17.27

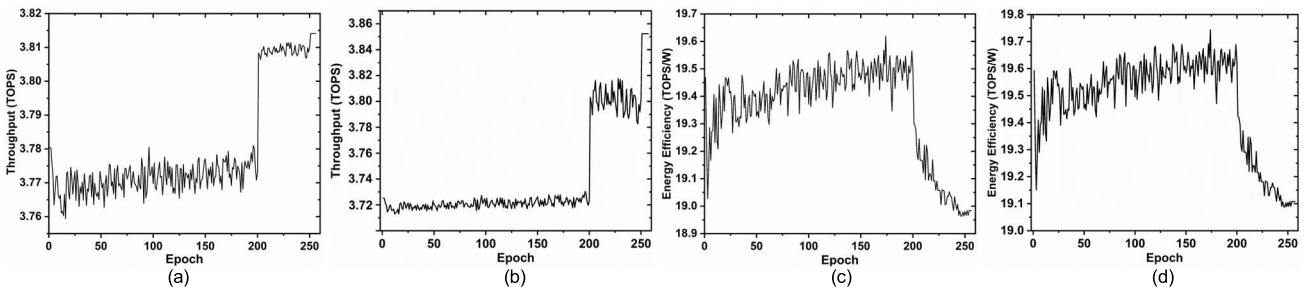

FIGURE 9. The framework was trained using the CIFAR-10 dataset for 256 epochs. Training throughput (TOPS) of VMM with (a) memristor device (b) memcapacitor device. Training Energy Efficiency (TOPS/W) of VMM with (c) memristor device (d) memcapacitor device.

memcapacitive VMM accelerator, and the hardware configuration for NeuroSim is in Table 1. We particularly emphasize the significance of on-state resistance (Ron), on-state capacitance (Con), and ADC precision in inference-only VMM accelerators, as these hardware factors play a crucial role in determining the accuracy and performance of the system. Our focus lies in analyzing data extracted from TiOx material memristive and Si material mem-capacitive devices based on the measurement results of 128 × 128 crossbar array, with specific attention to nonlinearity, crossbar sneak paths, asymmetry, device-to-device, and cycle-to-cycle variation for in-situ training. For benchmarking purposes, we utilize the 8-layer VGG network which is shown in Fig. 2 with the CIFAR-10 dataset across various device technologies. In Table 2 each cell precision refers to the number of bits employed to represent the conductance of each individual memristor cell. For instance, a system utilizing 5-bit precision memristor or memcapacitor cells signifies that each memristor can express 32 distinct conductance levels. When we accumulate data from 128 of these cells, we effectively combine their conductance values to calculate the weighted sum of inputs. The utilization of 5-bit ADC precision in this scenario implies that the analog sum of these 128 cells is being discretized into 5-bit digital values. Nevertheless, it's crucial to recognize that the selection of ADC precision can vary based on the specific application's requirements and the desired level of accuracy. Increasing the bit precision can potentially enhance accuracy as well.

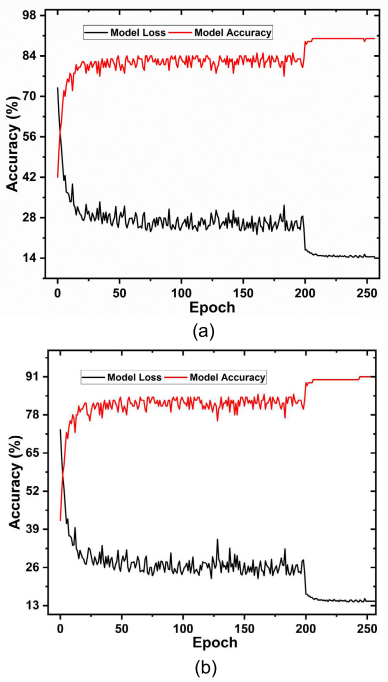

FIGURE 10. The training accuracy of the VMM accelerators with (a) memristor device, (b) memcapacitor device.

The framework also supports larger models as per the requirement. Simulation using the ResNet-34 network yielded training accuracies of approximately 86.14% for the memristor device and 85.81% for the memcapacitor device. These values were notably lower compared to the results from the VGG-8 network. However, for future iterations of the framework, we intend to include a wider range of networks and more complex datasets to enhance the comprehensiveness of our analysis. To determine whether the network is overfitting, we used a validation dataset to prevent overfitting, which is separate from the training dataset. We periodically monitored the model's performance on this validation dataset using metrics such as validation loss. The weights that minimized the loss or optimized the chosen metric were selected as the final trained weights. This approach ensured that the model did not overly adapt to the training dataset and could generalize well to new, unseen data.

The proposed framework's performance is thoroughly assessed using system throughput (TOPS) and energy efficiency (TOPS/W). TOPS measures computational performance in terms of the number of trillion operations a system can perform per second. On the other hand, TOPS/W is the ratio of throughput to power consumption, indicating how efficiently the system performs computations per unit of power consumed. Higher TOPS and energy efficiency values are desirable for achieving more powerful and energy-efficient computing systems. Detailed information of the VMM accelerators on TOPS and TOPS/W can be found in Fig. 9 and Table 2. The proposed approach is also evaluated by analyzing the training accuracy curves of both VMM accelerators, achieving approximately 90.02% and 91.03% accuracy, as shown in Fig. 10. Table 2 provides a detailed summary of the proposed method's performance, including information about both VMM accelerators, and compares it with other recently published DNNs implemented with mem-elements crossbar systems, specifically for the CIFAR-10 image classification task. Notably, with a significantly smaller network size, the proposed framework achieves comparable inference accuracy with 8-layer VGG network using both memristive and memcapacitive devices. Table 2 provides a concise comparison between the proposed work and previously published works. The size of the crossbar in [30] and [32] is large, leading to impractical area consumption, whereas our proposed VMM accelerator achieves a smaller footprint compared to [32], [33], [34], and [35]. Additionally, our proposed work exhibits higher training accuracy compared to [30], [32], [34], and [35]. Furthermore, our approach demonstrates better performance in terms of throughput (TOPS) and energy efficiency (TOPS/W) compared to earlier works. Unlike [31], which utilizes digital ReRAM with certain limitations, our proposed work utilizes hardware-extracted data based on memristor and memcapacitor, thereby addressing nonidealities more effectively. The comparison demonstrates the superior effectiveness of the proposed VMM accelerator, showcasing its successful utilization of in-memory computing for on-chip training.

V. CONCLUSION

In this paper, we introduced a comprehensive Python framework for evaluating large-scale deep neural network (DNN) on memristive and memcapacitive crossbar systems, taking into account various non-idealities. The framework considers device-level factors such as conductance, capacitance cycle-to-cycle, and device-to-device variations. The proposed approach was tested using a simplified 8-layer VGG network on a 128 $\times$ 128 RRAM array. The memristive and memcapacitive vector-matrix multiplication (VMM) accelerators achieved impressive training accuracies of 90.02% and 91.03% on the CIFAR-10 dataset, respectively. We also observed that training accuracy is influenced by line resistance, cycle-to-cycle, and device-to-device variations, which are well managed. Furthermore, the framework provides detailed hardware estimation for TiOx-based memristor and Si-based memcapacitor VMM accelerators.

REFERENCES

[1] J. Jang, S. Gi, I. Yeo, S. Choi, S. Jang, S. Ham, B. Lee, and G. Wang, "A learning-rate modulable and reliable TiO_x memristor array for robust, fast, and accurate neuromorphic computing," *Adv. Sci.*, vol. 9, no. 22, Aug. 2022, Art. no. 2201117.

[2] K.-U. Demasius, A. Kirschen, and S. Parkin, "Energy-efficient memcapacitor devices for neuromorphic computing," *Nature Electron.*, vol. 4, no. 10, pp. 748–756, Oct. 2021.

[3] C. Volos and V.-T. Pham, *Mem-Elements for Neuromorphic Circuits With Artificial Intelligence Applications*. San Diego, CA, USA: Academic Press, 2021.

[4] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nature Commun.*, vol. 9, no. 1, p. 2385, Jun. 2018.

[5] Z. Wang, C. Li, P. Lin, M. Rao, Y. Nie, W. Song, Q. Qiu, Y. Li, P. Yan, J. P. Strachan, N. Ge, N. McDonald, Q. Wu, M. Hu, H. Wu, R. S. Williams, Q. Xia, and J. J. Yang, "In situ training of feed-forward and recurrent convolutional memristor networks," *Nature Mach. Intell.*, vol. 1, no. 9, pp. 434–442, Sep. 2019.

[6] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020.

[7] K. Moon, E. Cha, J. Park, S. Gi, M. Chu, K. Baek, B. Lee, S. H. Oh, and H. Hwang, "Analog synapse device with 5-b MLC and improved data retention for neuromorphic system," *IEEE Electron Device Lett.*, vol. 37, no. 8, pp. 1067–1070, Aug. 2016.

[8] W. Sun, B. Gao, M. Chi, Q. Xia, J. J. Yang, H. Qian, and H. Wu, "Understanding memristive switching via in situ characterization and device modeling," *Nature Commun.*, vol. 10, no. 1, p. 3453, Aug. 2019.

[9] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature Nanotechnol.*, vol. 8, no. 13, pp. 13–24, Dec. 2012.

[10] S. Yu, B. Gao, H. Dai, B. Sun, L. Liu, X. Liu, R. Han, J. Kang, and B. Yu, "Improved uniformity of resistive switching behaviors in HfO_2 thin films with embedded Al layers," *Electrochem. Solid-State Lett.*, vol. 13, no. 2, p. H36, 2010.

[11] H. Li, P. Huang, B. Gao, X. Liu, J. Kang, and H.-S. P. Wong, "Device and circuit interaction analysis of stochastic behaviors in cross-point RRAM arrays," *IEEE Trans. Electron Devices*, vol. 64, no. 12, pp. 4928–4936, Dec. 2017.

[12] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan, "AxNN: Energy-efficient neuromorphic systems using approximate computing," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2014, pp. 27–32.

[13] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," 2015, *arXiv:1502.02551*.

[14] H. Tann, S. Hashemi, R. I. Bahar, and S. Reda, "Hardware-software codesign of accurate, multiplier-free deep neural networks," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2017, pp. 1–6.

[15] Z. Zhu, H. Sun, K. Qiu, L. Xia, G. Krishnan, G. Dai, D. Niu, X. Chen, X. S. Hu, Y. Cao, Y. Xie, Y. Wang, and H. Yang, "MNSIM 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in *Proc. Great Lakes Symp. VLSI*, Sep. 2020, pp. 83–88.

[16] S. Agarwal, S. J. Plimpton, D. R. Hughart, A. H. Hsia, I. Richter, J. A. Cox, C. D. James, and M. J. Marinella, "Resistive memory device requirements for a neural algorithm accelerator," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 929–938.

[17] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[18] M. J. Rasch, D. Moreda, T. Gokmen, M. Le Gallo, F. Carta, C. Goldberg, K. El Maghraoui, A. Sebastian, and V. Narayanan, "A flexible and fast PyTorch toolkit for simulating training and inference on analog crossbar arrays," in *Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2021, pp. 1–4.

[19] M. Le Gallo, C. Lammie, J. Buechel, F. Carta, O. Fagbohungbe, C. Mackin, H. Tsai, V. Narayanan, A. Sebastian, K. El Maghraoui, and M. J. Rasch, "Using the IBM analog in-memory hardware acceleration kit for neural network training and inference," 2023, *arXiv:2307.09357*.

[20] C. Lammie, W. Xiang, B. Linares-Barranco, and M. R. Azghadi, "MemTorch: An open-source simulation framework for memristive deep learning systems," *Neurocomputing*, vol. 485, pp. 124–133, May 2022.

[21] C. Lammie and M. R. Azghadi, "MemTorch: A simulation framework for deep memristive cross-bar architectures," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[22] C. Lammie, W. Xiang, and M. Rahimi Azghadi, "Modeling and simulating in-memory memristive deep learning systems: An overview of current efforts," *Array*, vol. 13, Mar. 2022, Art. no. 100116.

[23] S. Hwang, J. Yu, G. H. Lee, M. S. Song, J. Chang, K. K. Min, T. Jang, J.-H. Lee, B.-G. Park, and H. Kim, "Capacitor-based synaptic devices for hardware spiking neural networks," *IEEE Electron Device Lett.*, vol. 43, no. 4, pp. 549–552, Apr. 2022.

[24] J. Hur, Y.-C. Luo, A. Lu, T.-H. Wang, S. Li, A. I. Khan, and S. Yu, "Nonvolatile capacitive crossbar array for in-memory computing," *Adv. Intell. Syst.*, vol. 4, no. 8, Aug. 2022, Art. no. 2100258.

[25] B. Zhang, J. Saikia, J. Meng, D. Wang, S. Kwon, S. Myung, H. Kim, S. J. Kim, J.-S. Seo, and M. Seok, "A 177 TOPS/W, capacitor-based in-memory computing SRAM macro with stepwise-charging/discharging DACs and sparsity-optimized bitcells for 4-bit deep convolutional neural networks," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2022, pp. 1–2.

[26] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, "DNN+NeuroSim V2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2306–2319, Nov. 2021.

[27] A. Lu, X. Peng, W. Li, H. Jiang, and S. Yu, "NeuroSim simulator for compute-in-memory hardware accelerator: Validation and benchmark," *Frontiers Artif. Intell.*, vol. 4, Jun. 2021, Art. no. 659060.

[28] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 3, pp. 570–579, Sep. 2019.

[29] T. Cao, C. Liu, W. Wang, T. Zhang, H. K. Lee, M. H. Li, W. Song, Z. X. Chen, V. Y. Zhuo, N. Wang, Y. Zhu, Y. Gao, and W. L. Goh, "A non-idealities aware software–hardware co-design framework for edge-AI deep neural network implemented on memristive crossbar," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 4, pp. 934–943, Dec. 2022.

[30] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, Apr. 2010.

[31] L. Ni, Z. Liu, H. Yu, and R. V. Joshi, "An energy-efficient digital ReRAM-crossbar-based CNN with bitwise parallelism," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 3, pp. 37–46, 2017.

[32] S. Choi, S. H. Tan, Z. Li, Y. Kim, C. Choi, P.-Y. Chen, H. Yeon, S. Yu, and J. Kim, "SiGe epitaxial memory for neuromorphic computing with reproducible high performance based on engineered dislocations," *Nature Mater.*, vol. 17, no. 4, pp. 335–340, Apr. 2018.

[33] M. Jerry, P.-Y. Chen, J. Zhang, P. Sharma, K. Ni, S. Yu, and S. Datta, "Ferroelectric FET analog synapse for acceleration of deep neural network training," in *IEDM Tech. Dig.*, Dec. 2017, p. 6.

[34] S. Park, A. Sheri, J. Kim, J. Noh, J. Jang, M. Jeon, B. Lee, B. R. Lee, B. H. Lee, and H. Hwang, "Neuromorphic speech systems using advanced ReRAM-based synapse," in *IEDM Tech. Dig.*, Dec. 2013, p. 25.

[35] J. Woo, K. Moon, J. Song, S. Lee, M. Kwak, J. Park, and H. Hwang, "Improved synaptic behavior under identical pulses using AlO_x/HfO_2 bilayer RRAM array for neuromorphic systems," *IEEE Electron Device Lett.*, vol. 37, no. 8, pp. 994–997, Aug. 2016.

ANKUR SINGH (Student Member, IEEE) received the B.Tech. degree in electronics and communication engineering from the Indian Institute of Information Technology Guwahati, Guwahati, India, in 2021. He is currently pursuing the master's degree with the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju, South Korea.

His research interests include neuromorphic systems and in-memory computing.

BYUNG-GEUN LEE (Member, IEEE) received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2000, and the M.S. and Ph.D. degrees in electrical and computer engineering from The University of Texas at Austin, in 2004 and 2007, respectively.

From 2008 to 2010, he was a Senior Design Engineer with Qualcomm Inc., San Diego, CA, USA, where he has been involved in the development of various mixed-signal ICs. Since 2010, he has been with the Gwangju Institute of Science and Technology (GIST). He is currently a Professor with the School of Electrical Engineering and Computer Science. His research interests include high-speed data converters, CMOS image sensors, and neuromorphic system design.