Continuous On-Chip Learning in Neural Networks using SOT-MRAM based CIM Architectures

Anubha Sehgal*, Sandeep Soni*, Sumit Diware†, Alok Kumar Shukla*‡, Sourajeet Roy*, Rajendra Bishnoi†

*Indian Institute of Technology Roorkee, India, † Delft University of Technology, Netherlands,
‡ Madan Mohan Malaviya University of Technology, India
{anubha_s, sandeep_s, sourajeet.roy}@ece.iitr.ac.in, aks.ece@mmmut.ac.in, {s.s.diware, r.k.bishnoi}@tudelft.nl

2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD) | 979-8-3315-1560-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICCAD66269.2025.11240934

***Abstract*—Computational-In-Memory (CIM) is an energy-efficient paradigm that integrates computation directly within memory arrays, reducing the bottleneck associated with data transfer. This approach is beneficial for Artificial Intelligence (AI) applications that require on-chip learning for real-time processing. However, implementing on-chip learning in CIM architectures remains challenging due to limited throughput and energy-efficiency during both online training and inference. In conventional architectures, weight updates necessitate the inference process to halt to avoid unintended computation outcomes. To overcome this limitation, this paper presents a novel Spin-Orbit Torque (SOT)-based CIM architecture tailored for continuous on-chip learning applications, which enable weight updates without interrupting the inference. The proposed SOT bit-cell utilizes two read ports and one write port (2R1W) configuration, where one read port (1R) is dedicated to inference and one read and one write (1R1W) for on-chip learning that enables concurrent read and write operations. Our proposed architecture is evaluated at the system-level using the Generic-PDK 45 nm technology node, demonstrating 2.4× improvement in energy-efficiency and 5.4× improvement in throughput compared to state-of-the-art solutions, with minimal overhead.**

***Index Terms*—AI accelerators, multi-port memory, on-chip learning, spin-Orbit torque**

I. Introduction

The increasing demand for real-time machine learning applications has driven the need for energy-efficient hardware computing architectures. Traditional von Neumann architectures require frequent data transfer between processing and memory units, leading to significant energy consumption and latency due to memory bottlenecks [1]. This limitation becomes particularly severe in machine learning applications, where large amounts of data, including weights and activations, must be repeatedly fetched and updated. To overcome this issue, *Compute-in-Memory* (CIM) architectures have been introduced, integrating computation directly within memory arrays to reduce data movement [2]–[5]. Among various memory technologies explored for CIM, *Spin-Orbit Torque Magnetic Random Access Memory* (SOT-MRAM) has gained attention due to its non-volatility, high endurance, and fast switching speed [6]–[8]. These properties make SOT-MRAM a strong candidate for both inference and on-chip learning. However, traditional CIM architectures encounter a major bottleneck during on-chip learning due to the sequential nature of inference and weight updates. Since the weights updates need to be written back to memory after computation, the inference process must temporarily halt to maintain data integrity, which ultimately limits the system's throughput, energy-efficiency, and real-time learning capability.

To address this, [9] proposes a dual-port SOT-CIM architecture that enables simultaneous access, reduces read disturbance, improves bandwidth, and lowers latency, but it incurs an area overhead due to dummy bitlines and termination circuitry. To improve area efficiency, [10] uses voltage division across the track layer and leverages the resistance difference to enable low-power reads during writes, but it prioritizes write operations when both target the same bitline, limiting true parallelism. SRAM-based multi-port architectures have been explored to support parallelism and asynchronous operations in CIM designs; however, they often introduce challenges such as increased latency, energy consumption, layout complexity, and synchronization overhead [11], [12]. Additionally, other non-CIM SOT-based memory designs with dual-port or multi-port support show potential for improving parallelism but are not tailored for CIM workloads and lack compute-path integration [13], [14]. Since existing solutions to enable concurrent read-write operations either compromise on energy-efficiency, area, or scalability, there is a clear need for an optimized SOT-based CIM solution that can eliminate inference halts during training with minimal overhead while also improving energy-efficiency and throughput.

In this paper, we propose a design methodology that enables continuous on-chip learning by leveraging the decoupled read and write capabilities of the SOT device. To support this functionality, we introduce a custom bit-cell architecture equipped with an additional read port, which allows inference to proceed without disruption during training. Using this bit-cell, we develop a complete CIM architecture, along with dedicated peripheral circuits that efficiently manage current accumulation and support uninterrupted inference during ongoing training. We have also introduced a custom on-chip learning algorithm that supports simultaneous read and write operations. Simulation results demonstrate that our approach significantly boosts performance, achieves up to 2.4× higher energy-efficiency, and delivers 5.4× greater throughput com-

This work was supported by the Scheme for Promotion of Academic and Research Collaboration (SPARC) (Grant No: SPA-2136-ECD-CNA/23-24). Also, this work was partially supported by EU NEUROKIT2E (Grant No: 101112268).

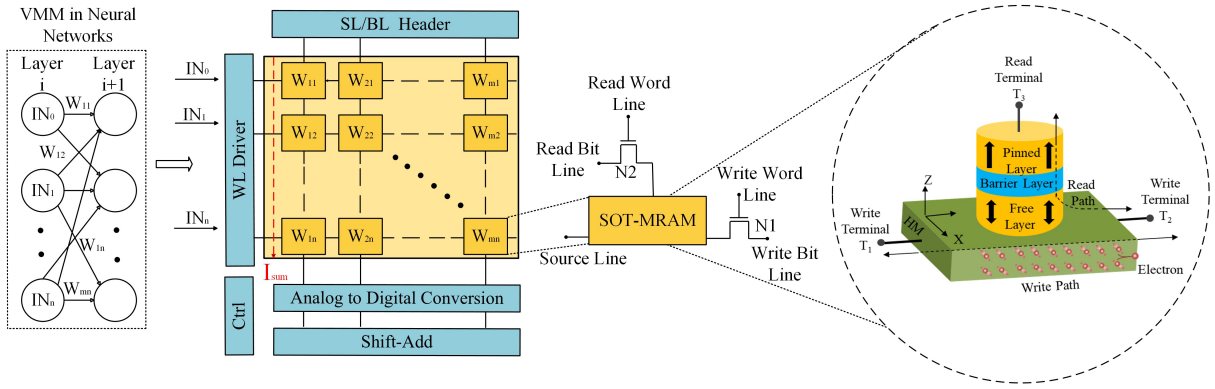


Fig. 1. Vector-matrix multiplication in Computation-In-Memory (CIM) architecture using SOT-MRAM bit-cell for neural networks.

pared to conventional SOT-MRAM-based CIM architectures.

The rest of the paper is organized as follows: Section II presents the background of CIM architecture, SOT-MRAM technology, concurrent read and write concept, and on-chip learning. Section III presents the proposed CIM architecture, and Section IV illustrates the results. Finally, Section V concludes the paper.

II. Background

A. Computation-In-Memory Architecture

CIM architecture integrates computation and memory to overcome the limitations of conventional von Neumann architecture, where frequent data transfer between memory and processors leads to high latency and energy consumption [15], [16]. By performing computations such as *vector-matrix multiplications* (VMM) directly within memory cells, CIM significantly reduces energy overhead and improves processing efficiency, making it highly suitable for Edge-AI applications [17]. A neural network framework based on CIM architecture utilizing VMM is shown in Fig. 1. This architecture uses SOT bit-cells in the crossbar array to encode network weights as resistance states. Input signals are applied concurrently across each row, where they are multiplied with the corresponding weights and produce weighted currents. After VMM operation, these currents are summed up along each column and then transformed to digital form using *Analog-to-Digital* (ADC) converters.

B. Spin-Orbit Torque Technology

SOT-MRAM is a three-terminal, non-volatile memory technology that utilizes SOT for fast and energy-efficient magnetization switching [18]. A typical SOT-MRAM consists of a *magnetic tunnel junction* (MTJ) comprising a free layer, a tunnel barrier, and a pinned/fixed layer on top of a *heavy metal* (HM) layer as shown in Fig. 1. The free layer is responsible for storing data by switching its magnetization direction, while the pinned layer maintains a constant magnetization that serve as a reference. The HM layer, typically composed of materials such as platinum (Pt), tantalum (Ta), or tungsten (W). It generates a spin current when a charge current is applied between two write terminals (T_1 and T_2) due to high spin-orbit coupling [19]. This spin current exerts torque on the free layer that switches its magnetization direction, resulting in a change in resistance due to the *spin Hall effect* (SHE) or Rashba effect in the HM layer [20], [21]. One of the key advantages of SOT-MRAM is its separate read and write paths, which enhance endurance and reliability [22]. The write operation is performed through the HM layer, while the read operation is performed independently by measuring the MTJ resistance through dedicated read terminals (T_3 and T_2). The MTJ resistance depends on the relative magnetization orientations of the free and pinned layers. When the magnetization directions are parallel (P), the MTJ exhibits low resistance, representing a logic "0", whereas an antiparallel (AP) configuration results in high resistance, corresponding to a logic "1". This resistance difference is known as *tunneling magnetoresistance* (TMR) that enables reliable data retrieval without disturbing stored information, thus significantly improving memory endurance and reducing power consumption in read operations [23].

C. Concurrent Read-Write Operations in SOT Devices

In SOT-MRAM cells, the write operation is carried out through in-plane current injection into the HM layer beneath the MTJ, which induces spin accumulation via the SHE and switches the magnetization of the free layer deterministically. In contrast, the read operation senses the MTJ resistance through a perpendicular current path that flows across the oxide barrier between the free and pinned layers, without affecting the magnetization state. This architectural decoupling effectively eliminates write-disturbance and enhances operational reliability, an issue that is commonly found in conventional two-terminal devices such as *Spin Transfer Torque STT-MRAM* [24], *Resistive Random Access Memory* (RRAM) [25], and *Phase Change Memory* (PCM) [26], etc. Due to this physical separation, read and write operations can occur concurrently on the same bit-cell without electrical or magnetic interference. As shown in Fig. 1, the write current flows laterally in the HM layer, while the read current flows vertically through the MTJ stack. As a result, their current paths are orthogonal. These demonstrations support the feasibility of concurrent operations from a device perspective. This feature is particularly advantageous for on-chip learning in neural

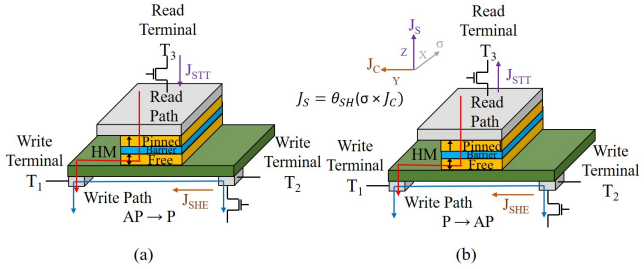


Fig. 2. SHE-assisted STT mechanism with (a) AP to P switching, (b) P to AP switching.

networks, where frequent weight updates and fast retrieval of stored values are essential for real-time learning and decision-making. Several prior works have already exploited the use of this principle through SHE-assisted STT switching [7], [27]. In this configuration, the SHE current (J_{SHE}) flows along the Y direction (from T2 to T1), resulting in spin accumulation at the interface between the free layer and the heavy metal, as depicted in Fig. 2 (a). This exerts a spin orbit torque on the free layer, rotating its magnetization from the Z direction toward the X-Y plane. Simultaneously, an STT current (J_{STT}) flowing in the Z direction (from T_3 to T_1) further assists the switching, pushing the magnetization from the X-Y plane to the +Z direction, achieving the AP to P transition. Similarly, for the P to AP transition in Fig. 2 (b), the direction of J_{STT} is reversed (T_1 to T_3), which flips the free layer magnetization back from +Z to -Z. The SHE switching mechanism helps lower the energy barrier that allows the STT to switch the magnetization with reduced current consumption. In this work, we exploit this property of SOT technology where read and write operations can pass through the device without interfering with each other's functionality.

D. On-Chip Learning

On-chip learning allows a computing system to perform data processing and machine learning tasks within the hardware. By integrating learning algorithms with memory components like SOT-MRAM, real-time updates and adaptive learning can be achieved. The training process consists of four key steps: feedforward computation, gradient computation, backpropagation error calculation, and weight updates [28], [29]. During feedforward propagation, memory cells not only store data but also execute matrix multiplications. The difference between the actual and expected output is determined by temporarily holding intermediate activations in buffers at each layer. Back-propagation then calculates weight gradients by convolving the error signal with activations from the feedforward phase. The learning rate is used to update the weights of the current layer. By using the trained model to create predictions directly for inference, the system minimizes data transfer, which lowers latency and energy usage [30].

III. Proposed SOT-based CIM Architecture

In this section, we first present an overview of our proposed CIM architecture design with a detailed explanation of the novel bit-cell design. Subsequently, we provide the architecture design details along with on-chip learning at the system-level.

A. Overview

For the on-chip learning process, the computed weight updates must be written back to the memory, which often requires a temporary pause in inference to avoid unintended computation outcomes and also requires retrieval of updated weights from memory in the next inference cycle. This leads to sequential execution, where learning and inference cannot proceed in parallel. These constraints pose a bottleneck in real-time applications like ECG monitoring, autonomous vehicle control, and drone navigation, where rapid data processing is essential. As shown in Fig. 3, during gradient calculation, the system halts the inference or forward pass, effectively blocking any further operations until the training process is complete. This results in increased application execution time due to interruption, restricting the efficiency of data processing, and limiting the feasibility of real-time operations. To address this problem, we propose a method that integrates inference directly into the on-chip training and weight update process, enabling uninterrupted inference even during training. This approach reduces execution time by eliminating system halts, making the learning process more efficient for real-time applications. To achieve this, we introduce a novel bit-cell structure along with a custom architecture, that are explained next.

B. Novel Bit-Cell Structure

The proposed SOT-MRAM based bit-cell for concurrent read and write operations is illustrated in Fig. 4 (a). The bit-cell consists of two independent read ports and one write port (2R1W). Among the two read ports, one (M4) is exclusively reserved for inference, while the other (M3) is used during training alongside the write port. This separation ensures that inference and training can occur in parallel without interfering with each other. In the SOT-MTJ cell, the write current flows

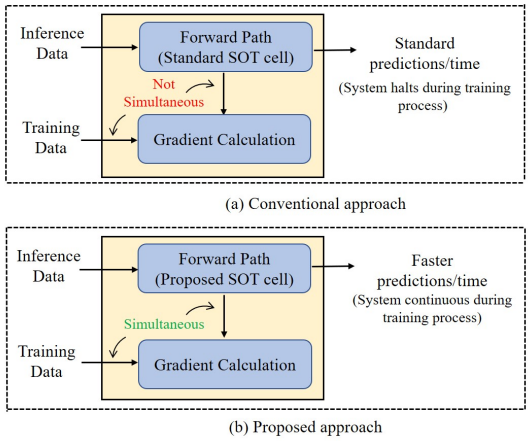
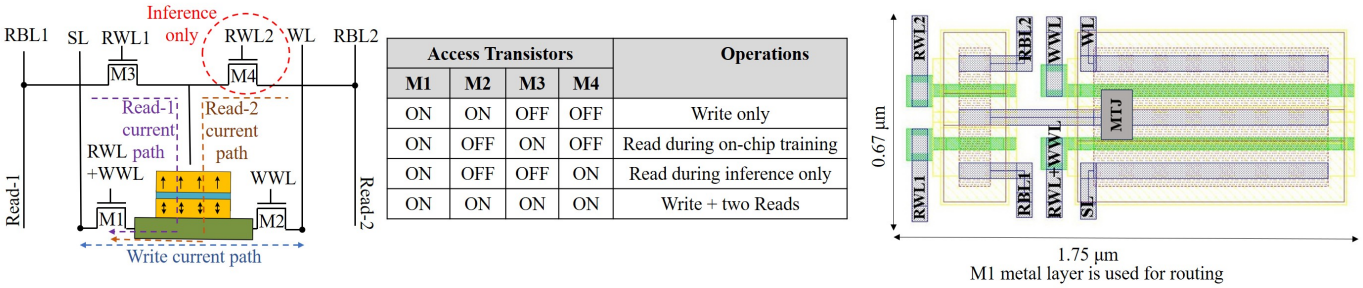


Fig. 3. Overview of the conventional and proposed on-chip learning and inference approaches.



Access Transistors				Operations
M1	M2	M3	M4	
ON	ON	OFF	OFF	Write only
ON	OFF	ON	OFF	Read during on-chip training
ON	OFF	OFF	ON	Read during inference only
ON	ON	ON	ON	Write + two Reads

(a) 1W2R Bit-cell schematic and its operation

(b) Layout design

Fig. 4. Illustration of concurrent read and write operations in an SOT-MRAM bit-cell design.

through the HM layer, not through the MTJ stack, maintaining the same current magnitude for both write-0 and write-1 operations. As shown in Fig. 4 (a), an additional write access transistor (M1) is introduced in the write path to eliminate asymmetry beyond the bit-cell level [31]. Additionally, an extra set of latches is incorporated to provide inputs during training.

During the on-chip training process, the write operation is enabled by activating M1 and M2 which allows bidirectional current to flow through the HM layer depending on the intended write value. The write circuitry drives the *Source Line* (SL) and *Write Line* (WL), establishing source and sink paths through the bit-cell. For the read operation during training, the access transistors, M3 and M1, are activated to establish the read current path. The read current passes through the oxide layer of the MTJ device, whose resistance is measured. It follows the same path as the write current of the MTJ device undergoing the write operation. The quantized read current is then sensed using ADCs. To facilitate synchronous read during training, an extra set of latches is added to provide the necessary input activations. Furthermore, a second read path is dedicated exclusively to inference, ensuring that inference operations remain isolated from training. This path is enabled by using transistors M1 and M4 to establish current flow. Then, the ADCs are used to sense the quantized current. An illustration of the two read and one write operations is shown in Fig. 4 (a). The layout of a bit-cell having a concurrent read-write design is shown in Fig. 4 (b), implemented using *Generic Process Design Kit* (GPDK) 45 nm technology and utilizing M1 metal layer.

C. Architecture Design

The macro-level architecture of a single CIM tile is shown in Fig. 5. A single CIM tile in the proposed architecture serves as a fundamental processing unit that integrates SOT-MRAM bit-cells, data converters, a control block, and weight computation/update blocks to efficiently execute VMM operations. The tile leverages SOT-MRAM bit-cells with two read ports and one write port, where one read port is dedicated to inference, and the other read and write ports are used during training. This separation ensures parallel execution of read and write operations without interference. The input data is

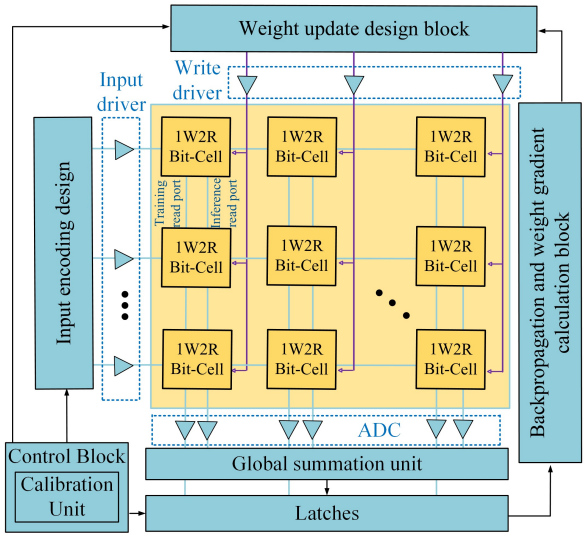


Fig. 5. Macro-level architecture of the proposed single tile.

first converted from digital to analog using *Digital-to-Analog Converters* (DACs) before being processed in the memory array, where stored weights modulate the inputs to perform the MVM operation. The resulting analog computations are then converted back to the digital domain using ADCs. During training, each tile uses a training read port to read intermediate values from the memory array. These are aggregated locally by a summation unit, then further combined across tiles via a global summation unit. The aggregated result is stored in a set of latches, and passed to the backpropagation and weight gradient evaluation block. Based on the calculated gradient, the updated weight is written back into the memory array via the dedicated write port, enabling efficient on-chip learning. During inference, the vector elements are connected to the gates of the access transistors. When a read voltage is applied through the read lines, a dot-product operation is performed, and the result is quantized using an ADC. The ADC supports shift-and-add operations and local partial sum accumulation to facilitate MAC computation within the tile. We employ a five-bit precision *successive approximation register* (SAR) ADC design in this work, as described by [32]. Based on our cell-level analysis, we can perform up to two sets of

Algorithm I: On-Chip Learning with Simultaneous Read-Write

Input: Input activations ($X^{(l)}$), Read weights during on-chip training ($W^{(l)}_{rold}$, $W^{(l)}_{rnew}$), Inference weights ($W^{(l)}_{inf}$), Target output (Y), Learning rate (η), Activation function (f), Loss function (F)

Output: Updated weights ($W^{(l)}_{w}$), Final loss (L), Predicted output ($X^{(L)}$), Gradients ($\delta^{(l)}$)

Mode:

A. **If (Inference Only):**

1. Prediction: $Z^{(l)} \leftarrow W^{(l)}_{inf} \times X^{(l)}$

B. **Else (Simultaneous Training and Inference):**

1. Prediction: $Z^{(l)} \leftarrow W^{(l)}_{rnew} \times X^{(l)}$
2. Forward Computation: $Z^{(l)} \leftarrow W^{(l)}_{rold} \times X^{(l)}$
% Inference is based on $W^{(l)}_{rold}$ to preserve accuracy during updates
3. Activation: $X^{(l+1)} \leftarrow f(Z^{(l)})$
4. Compute loss: $L \leftarrow F(X^{(L)}, Y)$
5. Output error: $\delta^{(L)} \leftarrow \frac{\delta L}{\delta X^{(L)}}$
6. Backpropagate error: $\delta^{(l)} \leftarrow \delta^{(l+1)} \times \frac{df}{dZ^{(l)}}$
7. Weight gradient: $\nabla W^{(l)} \leftarrow \delta^{(l+1)} \times (X^{(l)})^T$
8. Update weights: $W^{(l)}_{w} \leftarrow W^{(l)}_{rold} - \eta \nabla W^{(l)}$
9. Store updated weights: $W^{(l)}_{rold} = W^{(l)}_{w}$
10. Convergence check: **if** $L < \epsilon$ **then** stop training; **end**
11. **Return:** $W^{(l)}_{w}$, L, $X^{(L)}$

MAC operations (for example, executing two separate vector-matrix multiplications in parallel in the worst case) with our proposed bit-cell design while maintaining a margin of 40 mV [33]. Furthermore, once the values for all vectors are obtained, they are combined to generate the final output for the neural network. The calibration unit within the control block dynamically adjusts reference levels based on their operation mode, such as one read, two reads, combined one read and one write, or combined two reads and one write (see Fig. 8). This ensures accurate ADC behavior and reliable MAC operations across different phases, enabling continuous learning and inference without interference for real-time decision-making. Multiple CIM tiles are cascaded together to form a fully connected neural network, enhancing scalability and parallelism in on-chip learning applications.

D. On-Chip Learning at System-Level

On-chip learning involves both forward propagation (inference) and weight updates (training). In a simultaneous read-write operation, both inference and training are performed concurrently within the CIM array using a two read and one write port architecture. Algorithm I highlights the learning and inference process for on-chip learning at the system-level.

1) Forward Propagation and Error Computation: In this proposed on-chip learning architecture, inference is executed through a dedicated inference read port (RWL2) as shown in Fig. 4 (a), connected to read-only inference weights $W^{(l)}_{inf}$, for real-time applications that ensure uninterrupted computation. In contrast, the training process utilizes a separate training read port (RWL1), which accesses the previous training weights denoted as $W^{(l)}_{rold}$ during gradient computation. Input activations ($X^{(l)}$) are fetched and processed through the CIM array, where parallel MAC operations are performed using these inference weights. Since the inference path is isolated from training and write operations, it remains unaffected by any concurrent weight updates. The computed pre-activation output is passed through a non-linear activation unit (f) such as ReLU, sigmoid, to generate next layer activations ($X^{(l+1)}$). At the final layer, the predicted output ($X^{(L)}$) is compared with the target output (Y) using a loss function (F), such as cross-entropy or mean squared error, to compute the final loss (L). The corresponding output error gradient ($\delta^{(L)}$) is then calculated based on the loss and stored in temporary buffers or registers for weight updates during backpropagation. This decoupled two read and one write architecture enables simultaneous inference and training, eliminating interference during learning and enhancing parallelism and throughput.

2) Backward Propagation and Weight Update: The computed error gradients ($\delta^{(l)}$) are backpropagated layer by layer, where they are used to adjust the network weights. Each layer computes its local gradient based on the derivative of the activation function and the propagated error. The weight gradient ($\nabla W^{(l)}$) is computed as the product of the $\delta^{(l+1)}$ and the activation from the previous layer ($X^{(l)}$). These gradients are used to update the weights ($W^{(l)}_{w}$) through a standard optimizer such as stochastic gradient descent, applying the learning rate (η). The training read port is used to access previous weights ($W^{(l)}_{rold}$) during gradient computation, while the updated weights are written through the dedicated write port. Since activation values and gradients are read in parallel using read bit-cells, weight updates in the write cells do not interfere with ongoing computations. This enables simultaneous forward propagation and weight updates. This parallel, decoupled read-write architecture enables simultaneous forward and backward propagation, minimizing training latency and enhancing energy-efficiency compared to conventional sequential training methods.

IV. Results

In this section, we provide a detailed description of the setup used for calculating results, followed by a comprehensive explanation of both circuit-level and system-level evaluations.

A. Setup

Table I provides the parameters of the SOT-MRAM based model that are used for the circuit-level implementation, are stated in [34]. The layout is created using a *Generic Process Design Kit* (GPDK) 45 nm SPICE model, which is used

TABLE I
SOT-MRAM Device Parameters

Parameters	Value
Saturation Magnetization	1×10^6 A/m
Damping Factor	0.08
Spin Hall Angle	0.21
Critical Current	128 μA
Free Layer Dimensions	150 nm $\times$ 150 nm $\times$ 0.6 nm
Heavy Metal Dimensions	200 nm $\times$ 150 nm $\times$ 3 nm
Tunnel Magnetoresistance	150 % @ 300 K

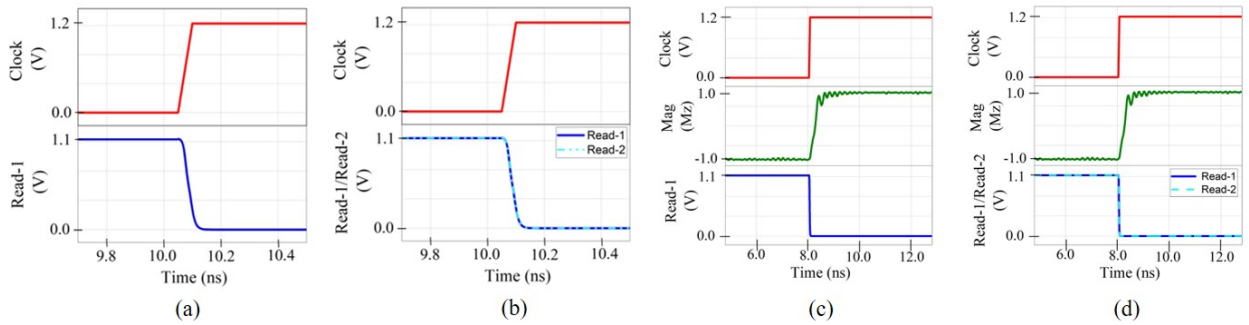


Fig. 6. Transient simulation of (a) 1R, (b) 2R, (c) 1W1R, and (d) 1W2R operations in a proposed SOT bit-cell.

to simulate the *Complementary Metal Oxide Semiconductor* (CMOS) components. In Cadence Virtuoso, simulations are carried out with a $\pm 3\sigma$ process variation and a supply voltage of 1.2 V at 27 °C . To account for worst-case scenarios at the bit-cell, row, and column levels, synthesis is carried out using *Cadence Genus*, while parasite extraction is carried out using *Calibre PEX*. The VGG-8 architecture is used to categorize 32x32 color images from the CIFAR-10 dataset for system-level assessment [35], [36]. This architecture includes two fully connected classification layers and six convolutional layers, each followed by max pooling. CIM inference and training performance are evaluated using the DNN + Neurosim V2.0 framework, which facilitates software/hardware co-design [28].

B. Circuit-Level Evaluation

The transient simulations shown in Fig. 6 validate the functionality of the proposed SOT-MRAM bit-cell under various operation modes. In Fig. 6 (a), a single read (1R) operation is performed by activating one read wordline (RWL1) as shown in Fig. 4 (a), for which the read current is sensed at Read-1. When the Clock is set to 0, the Read-1 node is precharged to 1, and when the Clock switches from 0 to 1, Read-1 discharges, validating the sensing mechanism. In Fig. 6 (b), the dual read (2R) operation is demonstrated by enabling both RWL1 and RWL2, where Read-1 and Read-2 are sensed simultaneously. The result shows that the two read operations do not interfere with each other. Fig. 6 (c) illustrates a one write and one read (1W1R) operation, where a write operation is performed by activating the write path wordline (WWL and WWL+RWL), while a read is simultaneously sensed through Read-1. The stable writing operation during the concurrent write and read operation validates the decoupled read-write paths that do not interfere with each other. Lastly, Fig. 6 (d) shows a one write two read (1W2R) operation, validating that two reads (Read-1 and Read-2) can be concurrently performed with a write without any functional disturbance.

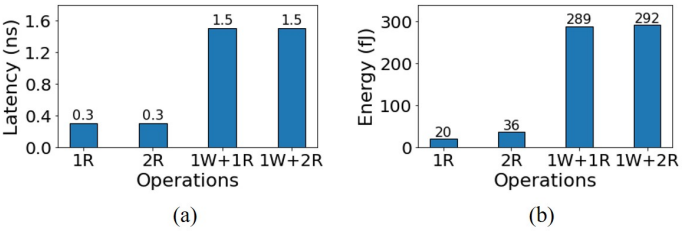


Fig. 7. (a) Write and read latency and (b) Write and read energy consumption of 1R, 2R, 1W+1R, and 1W+2R operations (W=Write, R=Read).

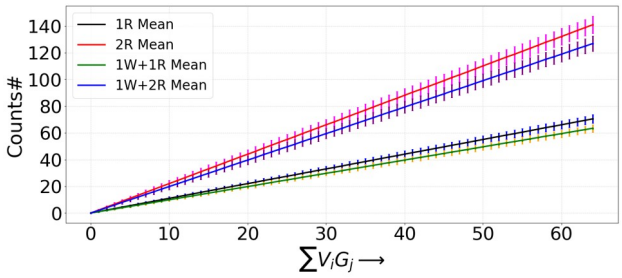


Fig. 8. Input-output characteristics of ADC having $\pm 3\sigma$ variations.

The efficiency of the proposed architecture is demonstrated in terms of circuit-level evaluation using SPICE simulations. Fig. 7 (a) and Fig. 7 (b) show the latency and energy consumption of the proposed bit-cell for various operations, respectively. The read operation exhibits a low latency of 0.3 ns for a single read (1R) and two simultaneous reads (2R), indicating minimal overhead when performing parallel read operations. The write operations (1W+1R and 1W+2R) introduce a significant delay of 1.5 ns. This is primarily due to the write operation that switches the magnetization of the free layer. Furthermore, the read and write latencies are inversely proportional to their respective current values,

TABLE II
PERFORMANCE OF VARIOUS COMPONENTS AND OPERATIONS FOR SEQUENTIAL AND SIMULTANEOUS WRITE-READ IN CIM ARCHITECTURE USING 128 × 128 ARRAY

Components	Sequential Write-Read		Simultaneous Write-Read	
	Latency (s)	Energy (J)	Latency (s)	Energy (J)
Accumulation	6.01	0.39	5.23	0.17
WG Computation	17.63	1.84	12.87	1.52
Feed-Forward	20.14	0.90	14.44	0.60
Error in BP	17.15	1.41	14.01	0.56
Weight Gradient	28.47	21.35	19.09	16.25
Weight Update	0.86	0.023	0.07	0.002

meaning that increasing the current can effectively reduce latency for both operations. Similarly, the read operations are highly energy-efficient that consumes 20 fJ for 1R and 36 fJ for 2R operations, while 1W+1R and 1W+2R operations require significantly more energy of 289 fJ and 292 fJ, respectively. In this paper, both read and write latencies are optimized for minimum energy consumption. However, these latencies can be further improved by applying a higher current, but at the cost of increased energy consumption [35].

Fig. 8 presents the input-output characteristics for an accumulation operation $\sum V_i \cdot G_j$ involving 64 rows, each with input and weight elements across different operational modes, including 1R, 2R, 1W+1R, and 1W+2R. Here, V_i denotes the input signal applied to the ith row, and G_j represents the conductance of the corresponding synaptic element in the jth column. The calibration unit within the control block dynamically adjusts ADC reference levels and sensing thresholds for each mode, thereby ensuring that the number of generated pulses remains proportional to the input despite shifts in reference levels or device non-idealities. The minimal spread observed in the $\pm 3\sigma$ error bars across all modes confirms the robustness of this adaptive referencing scheme.

C. Crossbar Sub-System Evaluation

A 128 × 128 synaptic crossbar array was implemented for each layer using Cadence Virtuoso. The output from each column was fed to a corresponding output neuron. The performance metrics, including latency and energy consumption, were evaluated over 256 training epochs. Table II provides a comprehensive hardware estimation, detailing latency and energy for key components and computational processes, including feedforward propagation, error computation in backpropagation, weight gradient calculation, and weight updates within the CIM accelerator.

The energy and latency of several components, such as ADC, accumulation module, weight update, and gradient computation, are depicted in Fig. 9 (a) and Fig. 9 (b). The feedforward propagation, weight gradient, and error computation in backpropagation are the primary contributors to overall latency and energy consumption. Among these, the weight gradient had a major effect on latency and energy since it required several read and write operations related to activation and error functions. On the other hand, weight updates required less computation time and energy since they were performed only once per batch. The area distribution of the crossbar array, including routing, ADCs, accumulation module, weight gradient, and other logic storage, is shown in Fig. 9 (c). Since the ADC and gradient computation units are independent of the CIM arrays and are implemented using CMOS technology, they occupied a significant amount of area. Other logic and storage units had a comparable area footprint, primarily for efficient data transfer. The accumulation logic included processing elements, adder trees, and gradient computation units.

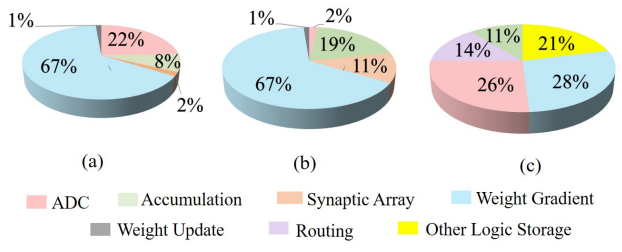


Fig. 9. Breakdown results of proposed circuit for (a) Energy, (b) Latency, and (c) Area by main components for 128 × 128 array.

TABLE III
SYSTEM-LEVEL PERFORMANCE OF DIFFERENT CIM ARCHITECTURES FOR ON-CHIP LEARNING

	[38]	[39]	[40]*	Proposed
Technology (nm)	65	65	45	45
Architecture	ResNet18	VGG16	VGG8	VGG8
Device	SRAM	SRAM	SOT	SOT
Area (mm^2)	12.0	12.0	31.7	46.9
Energy-Efficiency (TOPS/W)	9.5	6.2	20.2	48.7
Throughput (TOPS)	-	-	0.6	3.3
Training Energy-Efficiency (TOPS/W)	-	-	2.0	3.7
Training Throughput (TOPS)	-	-	0.8	2.5

* Values extracted from DNN + Neurosim V2.0 using bit-cell design in [40]

D. Comparison with State-of-the-Art

Table III presents a comparative analysis of different CIM architectures for on-chip learning across key metrics such as energy-efficiency, throughput, and area at the system-level. The proposed architecture exhibits an area overhead of 46.9 mm², primarily due to the integration of additional transistors that enable dedicated inference ports for concurrent read and write operations, thereby facilitating high parallelism. Despite the increased area, the proposed design achieves an energy-efficiency of 48.7 TOPS/W, representing a 2.4× improvement over the standard SOT-MRAM implementation (20.2 TOPS/W) due to reduced energy consumption per computation. In terms of computational throughput, the proposed architecture achieves 3.3 TOPS, which is 5.4× higher than the 0.6 TOPS evaluated for standard SOT-MRAM. Additionally, the training energy-efficiency is improved to 3.7 TOPS/W, compared to 2.0 TOPS/W in the standard SOT-based implementation, while the training throughput increases to 2.5 TOPS, outperforming 0.8 TOPS. These results highlight the effectiveness of the proposed architecture for real-time, energy-efficient, on-chip learning applications.

V. CONCLUSION

This paper presents a novel CIM architecture that enables continuous on-chip learning alongside real-time inference. The proposed architecture utilizes two read and one write port configuration, where a dedicated read port is assigned for inference, and one read and write port are utilized for on-chip learning that enhances parallelism and improves system-level performance. Simulation results demonstrate that the proposed architecture achieves a 2.4× improvement in energy-efficiency and a 5.4× enhancement in throughput compared to the standard SOT-MRAM based architecture.

REFERENCES

[1] J. W. Lee, *et al.*, "Strategic Development of Memristors for Neuromorphic Systems: Low Power and Reconfigurable Operation," *Advanced Materials*, pp. 2413916, 2025.

[2] A. S. Lele, *et al.*, "Assessing Design Space for the Device-Circuit Codesign of Nonvolatile Memory-Based Compute-In-Memory Accelerators," *Nano Letters*, vol. 25, no. 4, 2025.

[3] F. H. Meng and W. D. Lu, "Compute-In-Memory Technologies for Deep Learning Acceleration," *IEEE Nanotechnology Magazine*, vol. 18, no. 1, pp. 44-52, 2024.

[4] S. Diware, M. A. Yaldagard, and R. Bishnoi, "Hardware-Aware Quantization for Accurate Memristor-Based Neural Networks," *IEEE International Conference on Computer-Aided Design*, pp. 1-9, 2024.

[5] S. Diware, *et al.*, "Accurate and Energy-Efficient Bit Slicing for RRAM-Based Neural Networks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 1, pp. 164-177, 2022.

[6] A. Sehgal, *et al.*, "Enhancing Parallelism and Energy-Efficiency in SOT-MRAM based CIM Architecture for On-Chip Learning," *Design Automation Conference*, 2025.

[7] V. D. Nguyen, S. Rao, and S. Couet, "Recent Progress in Spin-Orbit Torque Magnetic Random-Access Memory," *NPJ Spintronics*, vol. 48, 2024.

[8] B. Wu, *et al.*, "An Energy-Efficient Computing-In-Memory (CIM) Scheme Using Field-Free Spin-Orbit Torque (SOT) magnetic RAMs," *IEEE Transactions on Emerging Topics in Computing*, vol. 11, no. 2, pp. 331-342, 2023.

[9] M. Natsui, *et al.*, "Dual-Port SOT-MRAM Achieving 90-MHz Read and 60-MHz Write Operations Under Field-Assistance-Free Condition," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 4, pp. 1116–1128, 2021.

[10] K. Zhang, B. Liu, and H. Cai, "Cache-Like Dual-Port SOT MRAM With Read-While-Write Access Strategy," *IEEE International Conference on Integrated Circuits, Technologies and Applications*, pp. 88–89, 2024.

[11] H. Fujiwara, *et al.*, "A 7nm 2.1GHz Dual-Port SRAM With WL-RC Optimization and Dummy-Read-Recovery Circuitry to Mitigate Read-Disturb-Write Issue," *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 390–392, 2019.

[12] H. Mori, *et al.*, "A 4nm 6163-TOPS/W/b 4790–TOPS/mm^2/b SRAM-Based Digital-Computing-in-Memory Macro Supporting Bit-Width Flexibility and Simultaneous MAC and Weight Update," *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 132–134, 2023.

[13] L. Chang, *et al.*, "Energy-Efficient Spin-Orbit Torque MRAM Operations for Neural Network Processor," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-5, 2021.

[14] L. Chang, *et al.*, "Multi-Port 1R1W Transpose Magnetic Random Access Memory by Hierarchical Bit-Line Switching," *IEEE Access*, vol. 7, pp. 110463-110471, 2019.

[15] M. Ali, *et al.*, "Compute-in-Memory Technologies and Architectures for Deep Learning Workloads," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1615–1630, 2022.

[16] S. Yu, *et al.*, "Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects," *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021.

[17] K. Yu, S. Kim, and J. R. Choi, "Trends and Challenges in Computing-in-Memory for Neural Network Model: A Review From Device Design to Application-Side Optimization," *IEEE Access*, vol. 12, pp. 186679–186702, 2024.

[18] R. Saha, Y. P. Pundir, and P. K. Pal, "Comparative Analysis of STT and SOT Based MRAMs for Last Level Caches," *Journal of Magnetism and Magnetic Materials*, vol. 551, no. 1, pp. 169161, 2022.

[19] Z. He, S. Angizi, F. Parveen, and D. Fan, "High Performance and Energy-Efficient In-Memory Computing Architecture Based on SOT-MRAM," *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pp. 97–102, 2017.

[20] L. Liu, *et al.*, "Spin-Torque Switching With the Giant Spin Hall Effect of Tantalum," *Science*, vol. 336, no. 6081, pp. 555–558, 2012.

[21] T. Endoh, *et al.*, "Recent Progresses in STT-MRAM and SOT-MRAM for Next Generation MRAM," *IEEE Symposium on VLSI Technology*, pp. 1–2, 2020.

[22] H. D. Kallinatha, S. Rai, and B. Talawar, "A Detailed Study of SOT-MRAM as an Alternative to DRAM Primary Memory in Multi-Core Environment," *IEEE Access*, vol. 12, pp. 7224–7243, 2024.

[23] Q. Shao, *et al.*, "Roadmap of Spin–Orbit Torques," *IEEE Transactions on Magnetics*, vol. 57, no. 7, pp. 1–39, 2021.

[24] D. C. Worledge and G. Hu, "Spin-Transfer Torque Magnetoresistive Random Access Memory Technology Status and Future Directions," *Nature Reviews Electrical Engineering*, vol. 1, pp. 730–747, 2024.

[25] S. Yu, *et al.*, "RRAM for Compute-in-Memory: From Inference to Training," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2753–2765, 2021.

[26] X. Sun, *et al.*, "PCM-Based Analog Compute-in-Memory: Impact of Device Non-Idealities on Inference Accuracy," *IEEE Transactions on Electron Devices*, vol. 68, no. 11, pp. 5585–5591, 2021.

[27] P. K. Mishra, *et al.*, "Voltage-Controlled Magnetic Anisotropy-Based Spintronic Devices for Magnetic Memory Applications: Challenges and Perspective," *Journal of Applied Physics*, vol. 135, pp. 220701, 2024.

[28] X. Peng, *et al.*, "DNN+NeuroSim V2.0: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators for On-Chip Training," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 11, pp. 2306–2319, 2021.

[29] J. Bilski, *et al.*, "A Novel Fast Feedforward Neural Networks Training Algorithm," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 11, no. 4, pp. 287–306, 2021.

[30] Z. Zhang, *et al.*, "Advances in Machine-Learning Enhanced Nanosensors: From Cloud Artificial Intelligence Toward Future Edge Computing at Chip Level," *Small Structures*, vol. 5, pp. 2300325, 2023.

[31] R. Bishnoi, F. Oboril, and M. B. Tahoori, "Low-Power Multi-Port Memory Architecture Based on Spin-Orbit Torque Magnetic Devices," *Great Lakes Symposium on VLSI*, pp. 409–414, 2016.

[32] H. Jiang, *et al.*, "Analog-to-Digital Converter Design Exploration for Compute-in-Memory Accelerators," *IEEE Design and Test*, vol. 39, no. 2, pp. 48–55, 2022.

[33] A. Singh, *et al.*, "Referencing-in-Array Scheme for RRAM-Based CIM Architecture," *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 1413–1418, 2022.

[34] K. Kazemi, *et al.*, "Compact Model for Spin–Orbit Magnetic Tunnel Junctions," *IEEE Transactions on Electron Devices*, vol. 63, no. 2, pp. 848–855, 2016.

[35] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv Preprint*, arXiv:1409.1556, 2014.

[36] R. Doon, T. K. Rawat, and S. Gautam, "CIFAR-10 Classification Using Deep Convolutional Neural Network," *IEEE PUNECON*, pp. 1–5, 2018.

[37] R. Bishnoi, *et al.*, "Improving Write Performance for STT-MRAM," *IEEE Transactions on Magnetics*, vol. 52, no. 8, pp. 1-11, 2016.

[38] J. Yue, *et al.*, "An Energy-Efficient Computing-in-Memory NN Processor With Set-Associate Blockwise Sparsity and Ping-Pong Weight Update," *IEEE Journal of Solid-State Circuits*, vol. 59, no. 5, pp. 1612–1627, 2024.

[39] J. Yue, *et al.*, "A 2.75-to-75.9TOPS/W Computing-in-Memory NN Processor Supporting Set-Associate Block-Wise Zero Skipping and Ping-Pong CIM With Simultaneous Computation and Weight Updating," *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 238–240, 2021.

[40] Y. Luo, *et al.*, "Performance Benchmarking of Spin-Orbit Torque Magnetic RAM (SOT-MRAM) for Deep Neural Network Accelerators," *Design and Applications of Emerging Computer Systems*, pp. 67–89, 2023.