Abstract: Recently, numerous studies have investigated computing in-memory (CIM) architectures for neural networks to overcome memory bottlenecks. Because of its low delay, high energy efficiency, and low volatility, spin-orbit torque magnetic random access memory (SOT-MRAM) has received substantial attention. However, previous studies used calculation circuits to support complex calculations, leading to substantial energy consumption. Therefore, our research proposes a new CIM architecture with small peripheral circuits; this architecture achieved higher performance relative to other CIM architectures when processing convolution neural networks (CNNs). We included a distributed arithmetic (DA) algorithm to improve the efficiency of the CIM calculation method by reducing the excessive read/write times and execution steps of CIM-based CNN calculation circuits. Furthermore, our method also uses SOT-MRAM to increase the calculation speed and reduce power consumption. Compared with CIM-based CNN arithmetic circuits in previous studies, our method can achieve shorter clock periods and reduce read times by up to 43.3% without the need for additional circuits.

Keywords: convolution neural network; computing in memory; processing in memory; distributed arithmetic; MRAM; SOT-MRAM

1. Introduction

Machine learning (ML) models, such as convolutional neural networks (CNNs) and deep neural networks (DNNs), are widely used in real-world applications. However, neural network structures have also increased in size, causing a bottleneck in the Von Neumann accelerator architecture. More specifically, the CPU must retrieve data from memory before processing it and then transfer it back to memory at the end of the computation in a Von Neumann architecture. This leads to additional energy consumption during data transfer, which reduces the energy efficiency of computing devices [1]. Furthermore, limited memory bandwidth, high memory access latency, and long memory access paths limit inference speeds and cause substantial power consumption regardless of the performance of the logic circuit. However, in-memory computing can effectively overcome the bottlenecks in the Von Neumann architecture. The CIM architecture can achieve low memory access latency, parallel operation, and ultra-low power consumption, and close access to the arithmetic logic unit of the CIM architecture can overcome the bottlenecks of the Von Neumann architecture [2].

The error rate of visual recognition by CNNs declined from 28% in 2010 to 3% in 2016 to become better than the 5% error rate in manual (i.e., human) visual recognition [3]. CNNs have been integrated into embedded systems to solve image classification and pattern recognition problems. However, large CNNs may have millions of parameters.
and require up to tens of billions of operations to process an image frame [4]. Therefore, accelerating the convolution operation yields the greatest improvement in performance. Iterative processing of the CNN layers is a common design feature of CNN accelerators. However, the intermediate data are too large to fit on the chip’s cache, and accelerator designs must, thus, use off-chip memory to store intermediate data between layers after processing. Due to the computational requirements of Internet of Things (IoT) and artificial intelligence (AI) applications, the cost of moving data between the central processing unit (CPU) and memory is a key limiter of performance.

CPU and GPU performance growth is approximately 60% per year; however, the performance increases in memory reach only up to 7% each year [5]. The data transfer rate in memory is insufficiently fast for the computational speed of the CPU; thus, the CPU is typically “data hungry.” Although deep learning processor performance has grown exponentially, most power consumption occurs during the reading and writing of data. Thus, the efficiency of the accelerator has little effect on performance.

In the field of hardware design, the computing units, such as GPUs, CPUs, and isolated memory modules, are interconnected with buses; this design has entailed multiple challenges, such as long memory access latency, limited memory bandwidth, substantial energy requirements for data communication, congestion during input and output (I/O), substantial leakage power consumption when storing network parameters in volatile memory. Additionally, because the memory used for AI accelerators is volatile memory, data are lost if power is lost. Therefore, overcoming these challenges is imperative for AI and CNN applications.

To design a hardware CNN accelerator with improved performance and reduced energy consumption, CIM CNN accelerators [5–7] constitute a viable method to overcome the “CNN power and memory wall” problem; these accelerators have been researched extensively. The key concept of CIM is the embedding of logic units within memory to process data by leveraging inherent parallel computing mechanisms and exploiting higher internal memory bandwidth. CIM could lead to remarkable reductions in off-chip data communication latency and energy consumption. In the field of algorithm design, several methods have been proposed to break the memory wall and power wall; these include compressing pretrained networks, quantizing parameters, using binarization, and pruning. Additionally, Intel’s Movidius Neural Compute Stick is a hardware NN accelerator for increased computing performance. In contrast, our approach is based on the MRAM CIM architecture, but the Movidius Neural Compute Stick is an onboard computing architecture. Compared to onboard computing architectures, MRAM CIM-based architectures significantly reduce the costs associated with data exchange between storage and memory. Our architecture has several key advantages, including non-volatility (no data loss in the absence of power), lower power consumption, and higher density. With the increasing demand for on-chip memory on AI chips, MRAM is emerging as an attractive alternative.

This paper follows the same assumptions in the existing works [5,8] and primarily focuses on methods of reducing hardware power consumption in edge computing without software algorithms. We designed a CIM CNN accelerator that is compatible with all the aforementioned algorithms without modifying the hardware architecture. Notably, we do not tackle the influence of slower peripherals to CNN.

Our contributions can be summarized as the follows:

• Integrate a DA architecture with the CIM to achieve faster speeds, fewer reads and writes, and lower power consumption.

• Optimize CNN operations and complete calculations in fewer steps.

• Integrate the DA architecture with the CIM and magnetic random access memory (MRAM) techniques to replace the original circuit architecture without off-chip memory. All calculations are performed on the cell array; thus, low latency can be achieved.

• Parallelize the CIM process using calculations in a sense amplifier to reduce power consumption and accelerate calculations.
The rest of this paper is organized as follows. Section 2 describes background and related work. We then describe the details of proposed architecture in Section 3. We provide the experimental process and results in Section 4. Finally, the conclusion is presented in Section 5.

2. Background and Related Work

2.1. CNN

CNN [9] is a combination of a feature extractor and a category classifier. The architecture uses shared kernel weights, local receptive fields, and spatial and temporal pooling to ensure invariance with respect to shift, scale, and distortion. Moreover, novel layers have been developed, such as the normalization layer and the dropout layer. CNN models typically have a feed-forward design; each layer uses the output of the previous layer as its input, and its calculation are output to the next layer. CNNs typically comprise three primary types of layers: the convolution (CONV) layer, the pooling layer, and the fully connected (FC) layer.

The convolutional layer is the main layer of a CNN. Each output pixel is connected to a local region of the input layer; this connection is called the receptive field. The receptive field can be defined as the window size of the region in the local input that produces a feature. These connections scan the entire image in the input feature map by extending a fixed-size window along the length and width of the entire image. The displacement of the window (i.e., the overlap of the receptive fields in both the height and width) typically has a value of 1 and is shared with the kernel weight. The processes of convolution is a 2D operation in which the shared kernel weight is multiplied element-wise with the corresponding receptive field. These element-wise operations require numerous executions of the multiplication and addition operations.

An input layer typically contains multiple channels, and the sum of all the channels is the result of the convolution. Pixel \( y \) at position \((x, y)\) in the convolution result for \( n \) is given as follows.

\[
\text{sum} = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \sum_{k=0}^{K_z-1} w(n)[i][j][k] \times \text{in}[i][x+j][y+k] \\
y(n)[x][y] = f(\text{sum} + b[n])
\] (1)

Therefore, if input data are extended to three dimensions of length, width, and depth, each 2D kernel must correspond to a depth.

The pooling layer can reduce the results of feature extraction but retain important features, typically reducing the image size by half. The pooling layer is generally placed following the convolutional layers. Average pooling and max pooling are two common pooling methods. In average pooling, the average value of the local field in each input feature map is calculated, whereas in max pooling, the maximum of the local field parameter is selected and pixels are output. Moreover, the number of output feature maps in the pooling layer must be equal to the number of input feature maps. Reducing the parameters can increase the efficiency of system operations; thus, a pooling layer is typically used when building neural networks.

The pooling layer is typically follows the convolutional layer, and the fully connected layer generally constitutes the final layers. The fully connected layer is usually a classifier that flattens the result to one dimension by converting it into a single vector that is used as the input of the next layer. The weight of the next FC layer is used to predict the correct label, and the output of the last fully connected layer is the final probability of each label.

Each of the three CNN layer can perform useful calculations; thus, our research combined these three layers to construct a highly accurate CNN model.

2.2. Spin-Orbit Torque MRAM

Spin-orbit torque MRAM (SOT-MRAM) [10] is the generation of MRAM following spin transfer torque MRAM (STT-MRAM). The main difference between STT-MRAM and
SOT-MRAM is that SOT-MRAM uses a more energy-efficient material called spin hall metal (SHM). SHM causes a rotating Hall effect on the application of a write current; this Hall effect creates a spin-torque switch on the magnetic channel of the free layer. SOT-MRAM does not require substantially more write current than STT-MRAM does because the area in which the current flows through SHM is relatively small. SOT-MRAM also has separate read and write paths, which can improve read and write speeds. Thus, we used SOT-MRAM circuit architecture.

An SOT-MRAM cell comprises two word lines, namely the read word line (RWL) and write word line (WWL); two bit lines, namely the read bit line (RBL) and write bit line (WBL); one source line (SL); and two access transistors. The details of SOT-MRAM operation are as follows.

On a rising write signal, the write current from the WBL flows in, and the WWL signal simultaneously activates the access transistor. Thus, the write current can flow through the access transistor. For a written value of 0(1), the current changes from SL(WBL) to WBL(SL). The direction of the free layer’s magnetic field can be changed by the spin Hall effect, which is generated by the different current directions.

If the direction of the changed magnetic field is parallel (anti-parallel) to the fixed magnetic field, the effective resistance of the MTJ is $R_P$ ($R_{AP}$), which has low (high) impedance. By connecting SL to GND and connecting the switch voltage source ($V_{write}$) to WBL, the direction of the write current can be changed directly.

On a rising read signal, the induced current passes through RBL. The sensing current then passes through the bit cell when the RWL signal switches on the transistor on the RBL side. To read the bit cell, the sense amplifier senses the voltage of the BL. The sensed current and the resistance of the bit cell are known when SL is grounded. Finally, the voltage value of BL can be calculated by the product of the sensing current and the effective resistance ($R_P$ and $R_{AP}$) of the unit.

2.3. Memory Comparison

Table 1 presents the characteristics of different types of memory for comparison [11]. The read/write speed of MRAM is similar to that of SRAM and DRAM, but the read power consumption is substantially lower than both DRAM and SRAM. In addition, MRAM is nonvolatile memory; thus, MRAM does not consume any energy outside of I/O operations, and data are not lost when power is disconnected. MRAM also has advantages in its manufacture over DRAM and SRAM in that it can be combined with an original digital circuit by adding a masking layer. In the future, MRAM may be able to replace the cache or flash memory in microcomputer units (MCUs). Therefore, MRAM is suitable for the design of CIM circuit architecture.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MRAM</th>
<th>SRAM</th>
<th>DRAM</th>
<th>Flash</th>
<th>FeRam</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Speed</td>
<td>Fast</td>
<td>Fast</td>
<td>Medium</td>
<td>Fast</td>
<td>Fast</td>
</tr>
<tr>
<td>Write Speed</td>
<td>Fast</td>
<td>Fast</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Array Efficiency</td>
<td>Med/High</td>
<td>High</td>
<td>High</td>
<td>Med/Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Future Scalability</td>
<td>Good</td>
<td>Good</td>
<td>Limited</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Cell Density</td>
<td>Med/High</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Nonvolatile</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Endurance</td>
<td>Limited</td>
<td>Infinite</td>
<td>Infinite</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Cell Leakage</td>
<td>Low</td>
<td>Low/High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>limited</td>
<td>Yes</td>
<td>Yes</td>
<td>Limited</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Complexity</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
</tbody>
</table>
2.4. Computing Bit-Wise Logical Operations in SOT-MRAM

2.4.1. Energy Efficient Method of AND/OR Operations in SOT-MRAM

A physics-based compact model for a three terminal PMTJ is proposed in [12], which models the magnetic, electrical, and thermal behaviors of a PMTJ controlled through SOTs. It considers the effects of both damping-like and field-like SOTs on device behavior. Moreover, the model tackles the dynamic behavior of the self-heating process within the device. However, the compact model does not design in-memory computing architecture for a convolutional neural network based on spin-orbit torque MRAM. After that, an energy efficient method of AND/OR operations in SOT-MRAM [8] is proposed, which is the first generation of MRAM using a changing magnetic field to change its internal resistance to store a bit (0 or 1). When a fixed current is input to read 0 or 1, different voltages can be obtained. Thus, if a current is input to two MRAMs, four different voltages are obtained, as presented in Figure 1b. The AND and OR results of these two cells can be obtained and can achieve CIM in the design of sensing amplifiers. However, this method of reading two cells simultaneously has disadvantages. As presented in Table 2, the voltage difference between two cells, namely voltage gap, is approximately 1 mV but can be as low as 0.5 mV. This slight gap is a substantial challenge for designing sense amplifier (SA) and also reduces the robustness of this circuit. Therefore, we adopted the circuit presented in Figure 1a to overcome this problem.

![SOT-MRAM circuit for CIM](image)

**Figure 1.** SOT-MRAM circuit for CIM [8]. (a) depicts an improved circuit with lower read and write current; the circuit is also more robust when executing CIM operations. (b) presents a conventional circuit that requires a larger current for reading and writing when performing CIM operations to achieve the same robustness as the improved circuit.

<table>
<thead>
<tr>
<th>Read Value of Two Cells</th>
<th>Parallel Resistance (ohm)</th>
<th>Read Voltage (mV)</th>
<th>Voltage Gap (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1500</td>
<td>1.5</td>
<td>-</td>
</tr>
<tr>
<td>01</td>
<td>2000</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>3000</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

First, the voltage of the cell is measured. This voltage is used to determine whether the input current is \( I_1 \) or \( I_0 \), and the ratio of \( I_1 \) to \( I_0 \) is equivalent to the ratio of \( R_{AP} \) to \( R_P \). As indicated in Table 3, the voltage difference is larger, and the calculation can, thus, be
more robust. Figure 2 reveals that, for the same read current, the voltage gap required by
the circuit of Figure 1a is greater than Figure 1b, indicating that the circuit has a greater
robustness. Therefore, low current and low power consumption can be used to set the same
voltage threshold.

![Figure 2. Differences between circuits in (a,b) during execution and operation.](image)

**Table 3.** Read result of the circuit of Figure 1a (1: \(R_{AP} = 6000 \) (ohm), 0: \(R_P = 3000 \) (ohm), \(I_1 = 1 \) µA, 0 = 0.5 mA).

<table>
<thead>
<tr>
<th>Reading Value of Two Cells</th>
<th>Reading Voltage (mV)</th>
<th>Voltage Gap (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1.5 mV</td>
<td>-</td>
</tr>
<tr>
<td>01</td>
<td>3 mV</td>
<td>1.5 mV</td>
</tr>
<tr>
<td>10</td>
<td>3 mV</td>
<td>3 mV</td>
</tr>
<tr>
<td>11</td>
<td>6 mV</td>
<td>3 mV</td>
</tr>
</tbody>
</table>

2.4.2. Majority Operation

The majority function returns an output of 1 if more than two signals of the three inputs
are 1. The truth table of the majority logic operation is presented in Table 4. According to
Table 4, the result of majority logic is equivalent to the Cout of a full adder. Therefore, this
characteristic can be used to implement a full adder in memory.

**Table 4.** Truth table of the majority function.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

2.4.3. Majority Decision in Memory

Kirchhoff’s circuit law indicates that the input current at a node is equal to the
output current at a node; this property can be used to implement a current adder. By
matching with a corresponding \(V_{ref}\), the majority result can be obtained, as presented in
Figure 3. Therefore, Cout in memory can be quickly obtained after the majority operation
is performed.
Figure 3. Voltage distribution of majority, where \( V_{\text{ref}} = 1.35 \text{ mV} \).

2.5. IMCE

IMCE [5] is a method from a paper published by Angizi et al. in 2018 [5]. The method uses bitwise in-memory computing to execute calculations. The method requires \( 3^2 \) steps to execute an AND operation for a 3-bit value. For each row, a bit count and shift are required. The total is then summed to complete a \( 4 \times 4 \) convolution. In addition, the bitwise in-memory computing requires additional circuits that increase the power consumption of the critical path. The advantage of the method proposed in this paper is that weight and data are stored in the same memory to facilitate calculation. However, this method requires additional circuits and more cycles to complete the calculation in memory. Overall, the power consumption and critical path of ICME are much greater than those of our method.

2.6. Energy-Efficient CIM Architecture

Kim et al. formulated another method in 2019 [8]. Their method first executes an AND operation to obtain partial sums, and it then uses a full adder circuit to complete all steps in sequence. If multiple bit lines are executed together, the final result must be calculated through the outer processing of \( \text{tempsum1} + \text{tempsum2} \ll 1 \). The advantage of the method is that it can run the full adder in memory without the use of additional circuits, but it requires more cycles to complete. The read and write operations must be executed numerous times, and the control of the method is also complicated. These shortcomings cause the data to be read slowly from the CPU, which is partly because the data are stored in the same destination address, meaning that each address can only read one data.

3. Proposed Architecture

Distributed Arithmetic (DA) was first introduced by Croisier et al. in 1989 [13]. It is an effective method of operations based on memory access and is effectively a bit-serial operation. The execution time depends on the clock speed, read/write speed of the memory, and the length of the operation bit. Figure 4 presents a DA circuit.

![DA Circuit](image)

Let us consider the convolution of two \( N \)-point vectors \( x_i \) and a fixed coefficient vector \( h_i \), which is expressed as follows:
\[ y = \sum_{i=0}^{N-1} x_i h_i, \]  
where \( h = [h_0, h_1, h_2, \ldots, h_{(n-1)}] \) and the input vector \( x = [x_0, x_1, x_2, \ldots, x_{(n-1)}] \). Let us assume that \( x_i \) is expressed in \( B \)-bit two’s complement representation as follows.

\[ x_i = -x_i + \sum_{j=1}^{B-1} x_{ij} 2^{-j} \]  

By substituting (3) in (4), the output \( y \) can be expressed in an expanded form as follows.

\[ y = \sum_{i=0}^{N-1} x_i 0 h_i + \sum_{j=1}^{B-1} \left[ \sum_{i=0}^{N-1} x_{ij} h_i \right] 2^{-j} \]  

Because \( h_i \) is constant, there exist \( 2^n \) possibilities for \( \sum_{i=0}^{N-1} x_{ij} h_i \) for \( j = 1, 2, \ldots, (B-1) \). However, these values can be calculated and stored in memory ahead of time. Thus, we can obtain a partial sum by the bit sequence as the address of the read memory. Therefore, the inner product can be calculated through an accumulation loop of \( B \) shifter-adders and by reading the value of the corresponding bit sequence. In our method, DA and the CIM structure are combined to overcome the challenges of the aforementioned model [14].

3.1. Integral Architecture

Figure 5 presents a memory circuit comprising eight banks. Each bank comprises 16 mats, and each mat has four cell arrays of size \( 16 \times 1024 \) for a total of 1 megabyte. The control circuit can only control eight banks simultaneously. Each operation can run 16 parallel mats operations, and each mat has 4 cell arrays. In addition, each cell array can perform four \( 3 \times 3 \) convolutions simultaneously; thus, the memory architecture can execute 64 convolutions in parallel.

Figure 5. Circuit architecture of memory.
3.2. Achievements Made by the New Architecture

Figure 6 presents our proposed CIM circuit architecture integrating a DA circuit architecture in memory without any digital circuits, such as a full adder or shifter circuit, to implement a DA calculation algorithm. In addition, the CIM architecture requires no additional weighting data; correspondingly, placing the results data and the buffer register on the same cell array can reduce both data access time and power consumption. The execution speed depends on the clock frequency, read speed of memory, and length of the calculation unit. Therefore, the novel CIM architecture performs faster than the traditional DA architecture and has lower power consumption because of the operations performed in memory.

Due to the advantages of the DA algorithm, the precalculated partial sum can be stored in the memory, and the shifter adder can then be used to accumulate the sum of each part. Therefore, our approach uses only a shifter adder and does not require a multiplier with a long critical path or a large area. Our new CIM architecture avoids the lengthy execution steps and additional circuits required by previous methods.

The main components of the DA architecture circuit are the read-only memory (ROM), reg buffer, full adder, and shifter. The following section describes the structure, operation, and implementation of these components to achieve the DA architecture in memory.

3.2.1. Build ROM and Register (Reg) Buffer in the Memory

MRAM is nonvolatile memory, and its read speed is similar to that of DRAM; thus, MRAM is suitable as the storage unit for a DA architecture. To increase the efficiency of in-memory calculation execution and to achieve lower latency and read/write power consumption, the weighted memory and buffer register stored in the CIM are placed on the same cell array shown, as presented in Figure 7. In addition, these defined memory sizes can be changed because the entire memory space, not only one specific part of the memory, can complete CIM operations.
3.2.2. Shifter

Because the shifter is unavailable in traditional memory, our method adds N-Metal-Oxide Semiconductor (NMOS) and P-Metal-Oxide Semiconductor (PMOS) to the SA circuit architecture, as presented in Figure 8. This change enables the output of the SA to be written into different columns based on shifter control without reading data out of the cell array or rewriting; these processes would otherwise extend the read/write time.

3.2.3. Shifter Full-Adder

This unit is used to complete a full adder operation. First, it calculates MAJ(A,B,Cin) to obtain Cout and obtain a sum in parallel in the following step. Then, \((A \oplus Cin) \oplus B\)
is performed to obtain sum-reg. Finally, the left shift to the sum is executed for the next shift-adder operation, as presented in Figure 9.

Figure 9. Steps of the shift adder.

4. Experimental Process and Result

4.1. Experimental Process

The simulation was divided into three stages, as presented in Figure 10. First, the Hspice tool was used to obtain a circuit level result. Second, the result was sent to the Nvsim model for simulation by using the memory architecture-level data obtained in stage 1. Subsequently, read/write power consumption and the memory delay were sent to GEM5 to execute system-level calculations. After these three stages of simulation, the written LeNet-5 algorithm could be executed in GEM5 to obtain the read/write power consumption of the entire algorithm. In our simulation, the parameters are set as follows. We considered the setup of our SPICE and process file, referring to references [15,16], respectively, with read voltage = 6 mV, current = 1 uA, and register = 6 kΩ. For the setup of NVSIM, we choose 1 MB memory with 8 banks, in which each bank has 16 MATs; each MAT has 4 cell arrays; and each cell array is 16 × 1024 bits in size. Additionally, we considered accelerators using 16-bit gradients; we select MNIST as our benchmark dataset, along with the LeNet-5 NN architecture. For the CNN layers of each 32 × 32 image, we developed a bitwise CNN with six convolutional layers, two average pooling layers and two FC layers, which are equivalently implemented by convolutional layers. After collecting this information, we could conduct comparisons using the experimental data. The detailed process of each stage is described in the following paragraph.

Figure 10. Experimental architecture.
4.1.1. SOT-MRAM Simulation

The SOT-MRAM model is built at the circuit level. The MRAM model used in our research is the same as that presented in [17]. That study’s authors provided the SOT-MRAM Verilog-A model file that is used to facilitate simulations and verifications of the real-world performance of MRAM memory. Figure 11 presents the simulation results of MRAM Verilog-A in Cadence Virtuoso.

![Figure 11. Simulated waveform of an MRAM cell.](image)

4.1.2. Process and Sensing Amplifier (SA)

We used NCSU FreePDK 45 nm [16] to simulate the SA circuit architecture and the digital circuit synthesis in Hspice. In addition, we chose StrongARM Latch [18], which consumes zero static power and has low latency. Thus, it is suitable for edge computing in the CIM architecture presented as Figure 12.

![Figure 12. StrongARM Latch.](image)

4.1.3. Nvsim

Nvsim [19] is a circuit-level model used to estimate the performance, energy, and area of new nonvolatile memory (NVM). Nvsim supports various NVM technologies, including STT-MRAM, PCRAM, ReRAM, and traditional NAND flash; thus, it was used and was modified to match the architectures we chose for simulation.
4.1.4. Gem5

Gem5 [20] is a modular discrete event-driven simulator for a full-system that combines the advantages of M5 and GEMS. M5 is a highly configurable simulation framework that supports a variety of ISAs and CPU models. In addition, GEMS complements the features of M5 by providing a detailed and flexible memory system, including multiple cache consistency protocols and interconnection models. It is a highly configurable architecture simulator that integrates multiple ISAs and multiple CPU models. In our experiment, we used a single-core Arm A9 CPU clocked at 2 Ghz as the CIM CPU for simulation analysis. Figure 13 presents the entire simulation process in Gem5. First, C code was compiled into a binary file, and Gem5 was then used to simulate the binary file and obtain a states.txt file containing data on the simulated CPU cycles and the read/write times of memory. Then, CPU power consumption could be obtained with the Mcpat tool.

![Simulation process diagram](image)

Figure 13. Simulation of an A9 processor running with LeNet-5.

4.2. Experimental Result

4.2.1. IMCE vs. Our Method

We analyzed the number of access times in reading, writing, and overall reading/writing separately; the results were then compared with those in previous IMCE studies. Figure 14 presents the convolution algorithm used for this comparison. The input image data and weight were both 8-bit. Figure 15 present three comparisons of reading and writing times. For reading times only, as presented in Figure 15a, we observed that our method is 49.9% faster that IMCE. As presented in Figure 15b, our method was 22.7% faster than IMCE in writing times. Finally, with regard to overall reading/writing times, as indicated in Figure 15c, our method was 43.3% faster than IMCE overall. This improvement was due to the use of the DA algorithm to substantially reduce the number of reads and, thus, reduce the power consumption during CIM by replacing multiplication with a lookup table.

![Convolution process diagram](image)

Figure 14. Convolution process.
4.2.2. Traditional Non-CIM Architecture and CIM Architecture

To determine whether the CIM architecture can effectively reduce computing power consumption, we used a non-CIM architecture and the CIM architecture to run the same NN as presented in Figure 16; this algorithm was also used in our experimental analysis. The main focus of our research is the ConV operation; therefore, the FC layer in the NN handled the CPU operation, and the CIM circuit architecture handled the ConV operation. Table 5 presents a comparison of the power consumption of the two architectures. Figure 17a shows the comparison of CPU power consumption. The difference is primarily because the convolution operation consumes the most power. In the CIM architecture, the convolution operation is moved to memory. Thus, the CPU does not perform the convolution operation, greatly reducing power consumption. Figure 17b illustrates the memory power consumption for comparison. CIM architecture can minimize the data transmission path and more greatly reduce the total memory power consumption. Moreover, because convolution has been moved to memory and also further reduced power consumption, the total read/write power consumption of the memory was lower. Figure 17c presents a comparison of the total power consumption, revealing that the CIM circuit architecture has lower overall power consumption.

Figure 15. (a) Reading times, (b) writing times, and (c) comparison.

Figure 16. LeNet-5 NN architecture.
Figure 17. (a) CPU, (b) memory, and (c) overall.

Table 5. Comparison of non-CIM architecture and CIM architecture.

<table>
<thead>
<tr>
<th></th>
<th>Arm A9</th>
<th>Arm A9 + CIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>23.43</td>
<td>0.5837034</td>
</tr>
<tr>
<td>CPU Memory</td>
<td>0.23</td>
<td>0.0006997</td>
</tr>
<tr>
<td>CIM</td>
<td>0</td>
<td>0.1459011</td>
</tr>
<tr>
<td>Total Power</td>
<td>23.66</td>
<td>0.7303042</td>
</tr>
</tbody>
</table>

4.3. Discussion

Our CIM architecture can be used for CPUs, GPUs, FPGAs, and ASIC in different design manners. In-memory computing has two advantages: making computing faster and scaling it to potentially support petabytes of in-memory data. In-memory computing utilizes two key technologies: random-access memory storage and parallelization. When the CPU/GPU processes data from the main memory, frequently used data are stored in fast, energy-efficient caches to enhance performance and energy efficiency. However, in applications that process large amounts of data, most data are read from the main memory because the data to be processed are very large compared to the size of the cache. In this case, the bandwidth of the memory channel between the CPU/GPU and the main memory becomes a performance bottleneck, and a lot of energy is consumed to transfer data between the CPU/GPU and the main memory. To alleviate this bottleneck, the channel bandwidth between the CPU/GPU and main memory needs to be extended but if the current CPU/GPU’s number of pins has reached its limit, further bandwidth improvement faces technical difficulties. In a modern computer structure where data storage and data calculation are separated, such a memory wall problem will be inevitably raised. Our CIM architecture is used to overcome the aforementioned bottleneck by performing operations in memory without moving data to the CPU/GPU. Additionally, our CIM architecture can also be implemented in FPGA or as an ASIC design under the assumption that the MRAM can be well taped out.

5. Conclusions

We proposed a new SOT-MRAM-based CIM architecture for a CNN model that can reduce both power consumption and read/write in comparison with conventional CNN CIM architectures. In addition, our method does not require additional digital circuits, enabling the MRAM cell to retain the advantages of memory for data storage. By conducting a series of experiments, compared with the ICME method [5], our proposed method reduces read times by 49.9%, write times by 22.7%, and overall read/write times by 43.3%.
Additionally, we evaluated that a CIM model running on an Arm A9 CPU can significantly reduce power consumption. In this paper, we did not tackle the changing magnetic field of MRAM. We used highly configurable architecture simulators SPICE, NVSIM, and GEM5 models to evaluate our proposed SOT CIM-based architecture. Quantifying the changing/switching magnetic field of the MRAM is an open issue. In the future, we will collaborate with industries to realize it.


**Funding:** This work is supported by the Ministry of Science and Technology, Taiwan, under grant MOST 109-2923-E-035-001-MY3, MOST 110-2112-M-033-013, and MOST 110-2221-E-035-034-MY3.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**


