

Received February 15, 2022, accepted March 8, 2022, date of publication March 16, 2022, date of current version April 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3159967

MagCiM: A Flexible and Non-Volatile Computing-in-Memory Processor for Energy-Efficient Logic Computation

VAHID JAMSHIDI¹, AHMAD PATOOGHY², AND MAHDI FAZELI³

¹Department of Computer Engineering, Shahid Bahonar University of Kerman (SBUK), Kerman 76169-14111, Iran

²Department of Computer Systems Technology, North Carolina Agricultural and Technical State University, Greensboro, NC 27411, USA

³Department of Information Technology, Halmstad University, SE-301 18 Halmstad, Sweden

Corresponding author: Vahid Jamshidi (vjamshidi@uk.ac.ir)


ABSTRACT This paper presents a high-performance and energy efficient processor exploiting a Magnetoresistive-based Computing-in-Memory array architecture (so-called MagCiM processor), to perform Boolean logic functions on operands stored in a memory array. The proposed processor efficiently addresses the memory wall and the leakage power consumption problems in conventional processors. The MagCiM processor utilizes mCell memory, a class of Magnetoresistive memory employing only Magnetic Tunnel Junction (MTJ) devices, to realize both computation-in-memory and on-chip instruction and data memories. The mCell memory is characterized by almost zero leakage power, high integration density, high level of reliability, and compatibility with the CMOS VLSI fabrication process. The circuit-level simulation results through comparisons with the previous work reveal that the MagCiM processor provides low occupation area, low power, and energy consumption and offers Normally-off instant-on computing capability, which makes it very suitable for embedded system applications. Based on our evaluations, a conventional processor based on the well-known MIPS architecture consumes about 13 times more energy while having 1.5 times more delay than the MagCiM processor.

INDEX TERMS Magnetic tunnel junction (MTJ), computation-in-memory (CIM), memory bottleneck, non-volatile memory.

I. INTRODUCTION

Big data processing applications with more random and less local access patterns ended up having unbalanced workloads on multi-core processors [1] that intensifies the memory wall problem. One effective solution to alleviate the communication costs is to place computation units in the proximity of memory units [2], the so-called Near Memory Processing (NMP). Instead of having close but distinct computation and memory units, the computations can be performed within memory units, known as Computing In Memory (CIM). Having memory units capable of both storing the data and performing simple computations, one can deal with the memory wall problem with the minimum required data movements [2].

SRAM technology might not be a fit candidate for CIM architectures due to its large area occupation (>30%) [3],

The associate editor coordinating the review of this manuscript and approving it for publication was Gian Domenico Licciardo .

high power consumption [3], and low reliability [4]. These issues worsen in data-intensive applications that demand larger memories to minimize the communication cost between off-chip and on-chip memories. Therefore, there are serious barriers against the deployment of SRAM memories for the design of memory-centric computing architectures such as CIM. Instead, magnetic tunnel junction (MTJ) based memories like STTRAMs feature small dimensions/high density, nearly zero leakage power, non-volatility, compatibility with semiconductors [5], and being robust against particle strikes. Various studies have been conducted on the design of MTJ-based memories [6], [7] and logic circuits [8], [9]. The non-volatility of MTJ cells can be perfectly used for helping the Normally-off/Instant-on computing paradigm [10].

In general, the application of non-volatile memories such as ReRAM [11], [12] and STTRAMs [13]–[15] in the design of CIM architectures can be categorized into Computing in Memory Arrays (CIMA) [11], [13]–[19] and Computing in Memory Edge (CIME) [20]. In a typical CIMA architecture

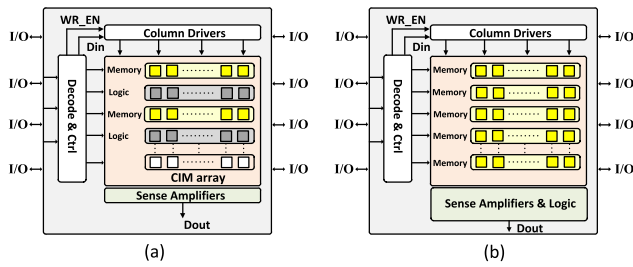


FIGURE 1. Computing in memory: (a) computing in memory arrays (CIMA). (b) Computing in memory edge (CIME).

(see Fig.1(a)), the computational logic is scattered between rows/columns of the memory and the sense amplifiers are placed at the edge of the memory to read data. In contrast to the CIMA architectures, in CIME architectures (Fig.1.b), computations are performed in a peripheral circuit at the edge of the memory unit. In the former, the logic/arithmetic operations are done within memory arrays. In contrast, some extra computation units (attached to the memory units as peripheral circuits) perform the logic/arithmetic operations in the latter. [16], [20].

The main shortcomings of the previous CIM architectures can be listed as follows.

I) ReRAM-based CIM architectures [11], [12] suffer from the higher operating voltage and higher read/write access delay of ReRAMs (w.r.t STTRAMs); and also suffers from low endurance (about 10^8) which makes ReRAMs an inappropriate candidate for on-chip memories.

II) The need for sense amplifiers to read data in STT-MRAM-based memories makes them vulnerable to decision failure. This is due to 1) temperature-related resistance variations in MTJs that may result in decision failure, especially for modern MTJs with low contrast between their high and low resistances. 2) having bit line's resistance significantly bigger than MTJ's resistance happens when the bit line length grows in bigger memories.

III) STT-MRAM-based memories suffer from the read disturb problem in which a read current may unwantedly change the magnetization direction of an MTJ free layer, i.e., the data is corrupted during the sensing process [21]. This problem does not apply to Spin-Orbit Torque Magnetic Random Access Memory (SOT-MRAM) as they use different write and read current paths. Overall, the previous CIM architectures are vulnerable to the decision failure mechanism regardless of the used technology [15], [19]. Though, the STT-MRAM-based architectures are also vulnerable to the read disturb failure as well [13], [16]–[18], [20].

IV) Although MTJs are experimentally proved to be robust against radiation-induced faults [22], the peripheral circuits in both STT-MRAM and SOT-MRAM based memories are still vulnerable to soft errors [23]. Consequently, all previously proposed CIM architectures based on either STT-MRAM or SOT-MRAM cells are vulnerable to radiation-induced errors. It should be noted that SOT-MRAM has been proposed to

mitigate such STT-MRAM challenges as 1) STTMRAM cannot reliably operate at sub-ns scales due to long incubation delays [6], making it an unsuitable solution to tackle L1/2 SRAM cache replacement and non-volatile logic; 2) The shared read/write path can impair the read reliability, while the write current can impose severe stress for the memory cell, which results in a possible time-dependent degradation of the MTJ.

V) The previously proposed STT-MRAM/SOT-MRAM based CIMA architectures are only capable of performing logic bit-wise operations [11], [13]–[15], [17]–[19].

To the best of our knowledge, the research problem of designing a full MTJ-based processor is still not addressed and this paper is the first research work in this context. The main contributions of this paper include:

- 1) An all magnetic computing in the memory array (CIMA) architecture using MTJ cells (the so-called MagCiM) is proposed. This is the first CIM architecture capable of performing both bit-wise logic and arithmetic operations within a memory array to the best of our knowledge. The proposed MagCiM architecture is robust against radiation-induced errors. We have eliminated the transistor-based read peripheral circuits, either the sense amplifiers or additional circuits used by CIME architectures for arithmetic operations. Eliminating these circuits also significantly reduces leakage power consumption. The MagCiM is immune against decision failure as it does not use sense amplifiers. Read disturb does not occur in the MagCiM, as the read and write current paths are disjoint.
- 2) Using MagCiM, we have designed and implemented an energy-efficient processor, the so-called MagCiM-processor that performs all arithmetic and logic operations in memory arrays without needing any register file or arithmetic/logic unit.
- 3) To support normally-off instant-on computing, the MagCiM-processor can completely switch off when it is in the idle mode. If needed, it operates instantly with maximum performance. This capability will help battery-operated embedded systems save more energy by switching to power-off more frequently without storing the system state.

The paper is organized as follows. Section II reviews related work. Section III provides an overview of mCell device. Section IV presents the proposed majority gate. Section V describes the CIM array used in the proposed processor. Section VI explains the structure of the proposed processor. Section VII describes normally-off computing in the MagCiM processor. Section VIII explains the pipelined design of the proposed processor. Section IX discusses the analysis and evaluation results. Finally, Section X presents the conclusion.

II. RELATED WORK

As mentioned in the introduction section, previous research work in the design of STT-MRAM/SOT-MRAM CIM

TABLE 1. The previously proposed computation-in-memory architectures (SA: sense amplifier, TD: threshold detector).

		Ref. [13]	Ref. [14]	Ref. [15]	Ref. [16]	Ref. [17]	Ref. [18]	Ref. [19]	Ref. [24]	Ref. [25]	Ref. [26]	MagCiM
Year		2018	2022	2018	2021	2018	2017	2018	2020	2018	2019	2022
Integrated With a Processor		NO	NO	NO	NO	NO	YES	NO	NO	NO	YES	YES
Memory Technology		STT-MRAM	STT-MRAM	SOT-MRAM	SOT-MRAM	STT-MRAM	SHE-DWM	SOT-MRAM	STT-MRAM	GSHE-MTJ	STT-MRAM	mCell
Data Extraction		SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	TD
Supported Computation	Logic Bit-wise Operations	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
	Arithmetic Operations	NO	NO	YES	YES	YES	YES	NO	YES	NO	YES	YES
CIM Type	CIMA	YES	NO	YES(Logic)	YES	YES	NO	YES	NO	YES	NO	YES
	CIME	NO	YES	YES(Arith)	NO	NO	YES	NO	YES	NO	YES	NO
Failure Mechanism	Decision Failure	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Robust
	Read Disturb	Vulnerable	Robust	Robust	Vulnerable	Vulnerable	Robust	Robust	Vulnerable	Robust	Vulnerable	Robust
	Single Event Upsets	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Robust

architectures can be classified into two broad categories: a) Computing-In-Memory Arrays (CIMA), and b) Computation-In-Memory Edge (CIME). Table 1 lists recently proposed CIMA and CIME architectures and compares these architectures based on various features including the integration of the architecture in a processor, the used technology, the data extraction approach, failure mechanisms, logical bit-wise operation support, arithmetic operation support, and the type of CIM architecture. To make it easier to follow the paper, we also have added our proposed (MagCiM) architecture in the table. As shown in the table, the proposed MagCiM is the CIMA architecture that performs both logic and arithmetic operations within memory arrays. The MagCiM architecture is also highly robust against various failure mechanisms, and we have integrated it within a processor.

According to Table 1, the mentioned methods are not architecturally the same. Some of them are processors, and others are just a CIM. In addition, some perform only logic bit-wise operations, while others perform both logic bit-wise operations and arithmetic operations (by adding power-hungry modules). Considering these differences in their architectures, comparing the methods in terms of power consumption and computational complexity load in table 1 is not fair. Therefore, in Table 1, the common features of the architectures have been stated.

The major shortcoming of the previously proposed CIMA architectures can be summarized as follows. 1) As most of them are designed based on STT-MRAM devices, they have reliability issues in decision making and read disturb. 2) Although computations are done within the memory, they still require complex extra MOS-based circuits and different control signals to connect specific rows/columns. Such circuits impose static and dynamic power overheads and negatively impact the reliability of the CIM architectures. 3) The most crucial part of designing a CIM architecture is to try to integrate the CIM in a processor (this may need the design of corresponding instruction and hardware set architectures). Most of the previous CIMA architectures have ignored such an integration.

Although STT-MRAM based CIME architectures can perform both logic and arithmetic operations, they suffer from three problems: 1) The used fairly-complex

circuit at the memory peripherals that performs arithmetic and/or logic operations imposes notable static and dynamic power consumption. The computing circuit is also prone to radiation-induced faults and may end up in a single point of failure. 2) The CIME architectures require additional signals across the memory's columns/rows that consume more energy when the memory capacity grows. 3) Like standard STTRAMs, CIME architectures are also subject to the common failure mechanisms such as decision failure and read disturb.

III. OVERVIEW OF mCell DEVICE

The storage cell used in the mCell-MagCiM array is a spintronic device called mCell [7] (shown in Fig. 2). mCell is a four-terminal device with electrically separated read path (R, R*) and write path (W-, W+). mCell works based on magnetic domain wall in the write path which is displaced back and forth by positive and negative pulsed current, respectively. When the write path is magnetically coupled to the free layer of a magnetic tunnel junction, read path resistance changes as determined by the element's tunneling magnetoresistance. In our work, two terminals of mCell comprise a write-path, wherein the direction of flowing input current charges the digital state of the device. The other two terminals include a read-path that is electrically separated from the write-path. The state of the device is detected as a high or low resistance through the read-path terminals. As shown in Fig. 2(b) and Fig. 2(c), an input current (I_{mCell}) larger than or equal to threshold current (I_C) passing from W- to W+ changes the mCell read path resistance from a value between R and R* to high (RH) value, and a reverse current changes it to low (RL). Therefore, the input to the mCell are currents $+I_{mCell}$ or $-I_{mCell}$. The resistance of mCell can be measured by passing a sense current from R to R*. These two states are non-volatile, i.e., they are preserved even if the supply voltage is disconnected.

Table 2 lists all essential parameters values which are used in simulations and analysis of the MagCiM processor in this paper.

IV. THE PROPOSED MAJORITY GATE

It has been proven that the logic of a majority gate is functionally complete [28] i.e., it covers every logical function.

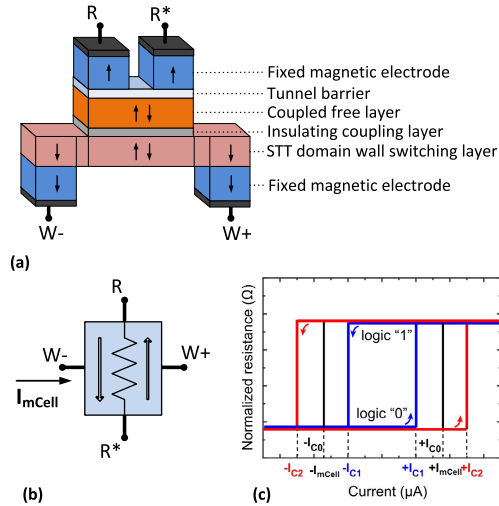


FIGURE 2. Four-terminal mCell device: (a) 3-D view, (b) schematic symbol, (c) illustration of mCell-driven switching under different threshold currents. The read path of the mCell is through the (R, R*) terminals and the write path is through the (W-, W+) terminals [7].

TABLE 2. The mCell specifications used in the MagCiM processor [27].

Threshold Current Density [MA/cm ²]	4
MTJ Resistance*Area [ohm*μm ²]	2
Tunnel Magnetoresistance Ratio (TMR)	200%
Read Path Low Resistance [Ω]	1.25 k
Read Path High Resistance [Ω]	3.75 k
Write Path Resistance [Ω]	120
Spin Polarization	120%
Domain Wall Depinning Time [ps]	100

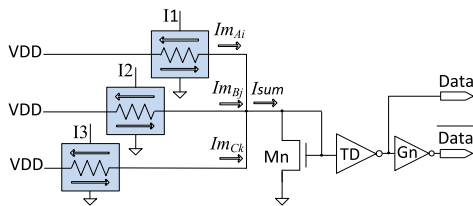


FIGURE 3. The proposed majority gate.

Fig. 3 shows our proposed majority gate used for performing logical operations in the MagCiM ALU. As shown in this figure, the inputs are of current, whereas the outputs are of voltage. The Mn transistor converts the resultant of Im_A , Im_B , and Im_C into a voltage and transfers it to an element called the voltage threshold detector (TD). The TD is a CMOS inverter that uses the midpoint voltage, V_M (defined as $V_M = V_{in} = V_{out}$), to obtain the threshold. At the midpoint voltage, both NMOS and PMOS transistors of the inverter operate in the saturation region. So, considering $V_{GS,N} = V_M$ and $V_{SG,P} = V_{DD} - V_M$, Eq. 1 and Eq. 2 obtain the drain currents.

$$I_{D,N}(SAT) = \frac{K_N}{2} (V_{GS,N} - V_{TN})^2 = \frac{K_N}{2} (V_M - V_{TN})^2 \quad (1)$$

$$I_{D,P}(SAT) = \frac{K_P}{2} (V_{SG,P} + V_{TP})^2 = \frac{K_P}{2} (V_{DD} - V_M + V_{TP})^2 \quad (2)$$

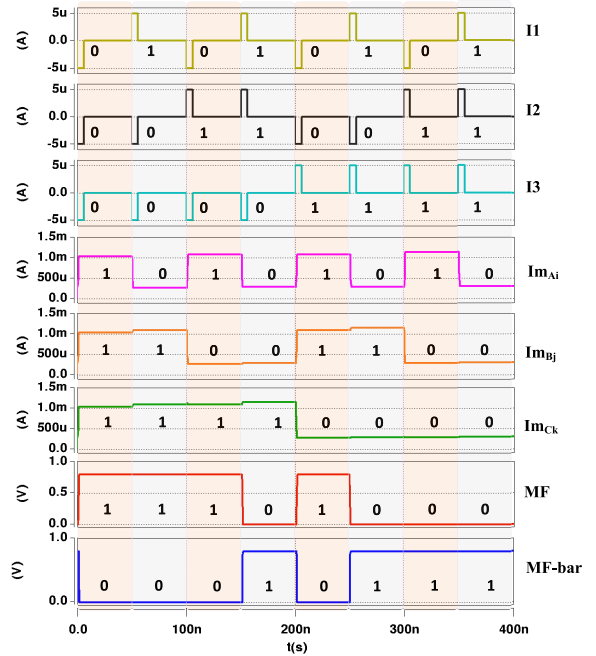


FIGURE 4. The output waveforms of the proposed majority gate.

where $k_N = k'_N (\frac{W}{L})_N$ and $k_P = k'_P (\frac{W}{L})_P$. k'_P and k'_N are the parameters of process transconductance. Equating drain currents and solving equation, V_M can be obtained by Eq. 3:

$$V_M = \frac{V_{DD} + V_{TP} + V_{TN} \sqrt{\frac{k_N}{k_P}}}{1 + \sqrt{\frac{k_N}{k_P}}} \quad (3)$$

The midpoint voltage is adjusted by changing $(\frac{W}{L})_N$ and $(\frac{W}{L})_P$; to obtain the necessary threshold. For the majority gate to work correctly, the threshold is adjusted to satisfy the following condition: the output will change from high to low every time the result becomes greater than '1.5'. Therefore, we have:

$$\begin{cases} R_1 = RL || RL || RH \\ R_2 = RL || RH || RH \end{cases} \Rightarrow R_{Mid} = \left(\frac{R_1 + R_2}{2} \right) \quad (4)$$

$$V_M = V_{DD} - I_{Mn}(Sat) \times R_{Mid} \quad (5)$$

Considering Eq. 3 and Eq. 5, it is concluded that the resistances of mCells (RL, RH) affect k_N and k_P values. The Gn inverter inverts the TD output so that both *data* and *data* will appear at the output. Fig. 4 shows the normal operation of proposed majority gate, extracted by HSPICE simulations.

As VDD, transistor specifications, and mCell specifications are not same in different technology, the value of Midpoint Voltage (VM) would be also different, and its value should be obtained based on Eq. 3 to Eq. 5 for each technology.

V. MagCiM: THE PROPOSED MAGNETIC BASED COMPUTING IN MEMORY ARRAY

Fig. 5 shows the structure of the proposed Magnetoresistive-based Computation-In-Memory (MagCiM) array. The

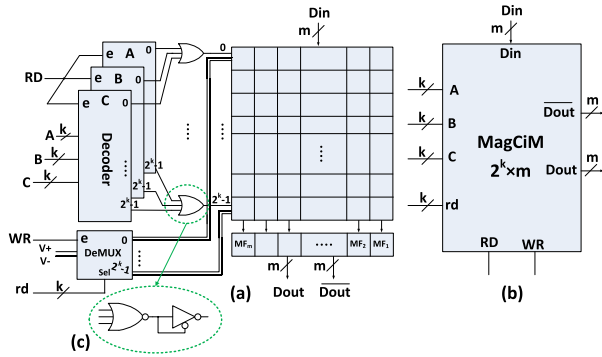


FIGURE 5. The proposed magnetic based computing-in-memory array (MagCiM) (a) MagCiM structure. (b) Schematic view. (c) OR structure for preventing the sneak current problem.

MagCiM uses three decoders (A, B, and C) and one de-multiplexer (DeMUX). Outputs of decoders are connected to an OR gates' input to allow concurrent read of three rows in memory. The DeMUX module makes it possible to write data on the selected row. The RD signal is applied to the enable pins of decoders (A, B, and C), whereas the WR signal enables the DeMUX module. When the RD signal is set to '0', the decoders' outputs become '0', and when the WR signal is '0', the outputs of DeMUX become '0'. In MagCiM architecture, a computation process follows two steps. In the first half-cycle, the RD and WR signals are set to '1' and '0', respectively to read the memory. Then, three decoders are activated, and three stored bits of each column are selected simultaneously and applied to the majority gate to perform the computation on the selected bits. The result finally appears in the MagCiM output (Dout and \overline{Dout}). The majority gate, responsible for all computations, performs computations simultaneously with the read operations in the first half-cycle. In the second half-cycle, RD and WR signals are set to '0' and '1', respectively. The input data (Din) is then written into a MagCiM memory row determined by DeMUX.

Fig. 6 shows the MagCiM array, which uses the all-magnetic mCell-based bit-cells proposed in [7]. Locations denoted as $M0$, $M1$, and $M2$ are reserved locations of the MagCiM array to achieve logic bit-wise operations. The $M0$ cells are set to "0...00", whereas $M1$ cells are set to "1...11". For performing arithmetic operations, $M2$ cells are reserved for storing the result of the ripple carries. In the following, we explain the operations of these components in detail.

A. ALL-MAGNETIC mCell-BASED BITCELL

As every memory cell consists of three mCells, two of which form a buffer called 'driving buffer'. The third mCell keeps data and transfers it to the output in the reading phase.

B. WRITING INTO THE DRIVING BUFFER

Unlike conventional memories, the WBL(i) signal is of the current i.e., the direction of the WBL(i)'s current determines the status of driving buffer mCells (Fig. 7).

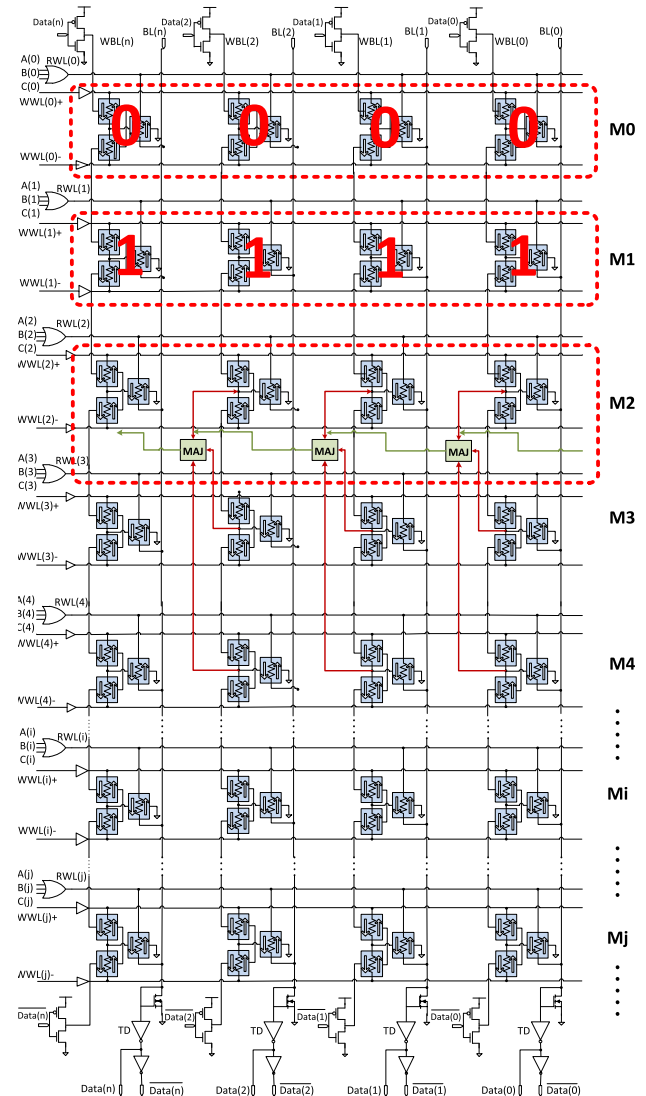


FIGURE 6. The proposed $M \times N$ mCell-MagCiM array.

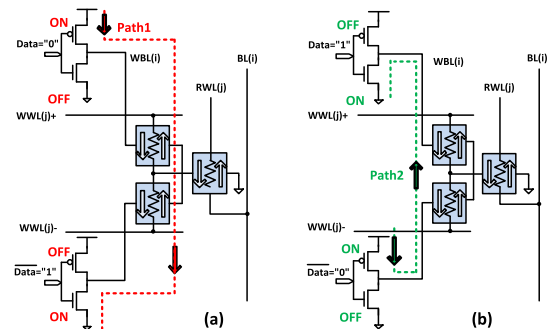


FIGURE 7. Two data writing paths are made by the Data and \overline{Data} signals. (a) The path-1 performs a write operation of logical value of '0'. (b) The path-2 performs a write operation of logical value of '1'.

1) WRITING INTO A MEMORY CELL

To write in a memory cell, the WWL(j)+ and WWL(j)- writing lines are set to V+ and V- values,

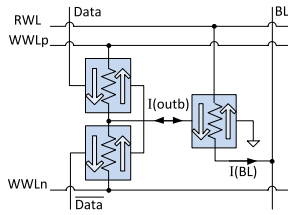


FIGURE 8. Structure of an all-magnetic mCell-based bitcell.

respectively. This causes the electrical current to flow through the writing path of the memory cell. The direction and the magnitude of the writing current depend on the pull-up and pull-down resistors of the driving buffer reading path. In Fig. 7(a), the pull-up resistor of the driving buffer is greater than the respective pull-down resistor on Path-1. If the reading path of the driving buffer is activated (by activating WWL(j)+ and WWL(j)- lines), a right-to-left current on the writing path sets the value of the memory cell to zero. In Fig. 7(b), the pull-up resistor of the driving buffer becomes smaller than the corresponding pull-down resistor on Path-2. If the reading path of the driving buffer is activated (by activating WWL(j)+ and WWL(j)- lines), a left-to-right current on the writing path of the memory cell sets the value to '1'.

2) READING FROM A MEMORY CELL

To perform a reading operation, the RWL(j) line is initialized with the value V+, and the output current is transferred to the BL(i) line through the reading path of the memory cell. Given the fact that a mCell has the two low resistors (Logic '0') and high resistor (Logic '1') states, the current transferred to the RBL through the memory cell can also have two values. When the value of the storing element is '1', the output current will be low (IOL), otherwise when the value of the storing element is '0', the output current will be high (IOH).

Fig. 8 shows an MagCiM memory cell, and Fig. 9 indicates its corresponding waveforms.

C. THE mCell-MagCiM ARRAY

Similar to other random access memories (DRAM, MRAM, etc.), the MagCiM is divided into individual rows and columns. Each row has a dedicated word-line, and each column has a dedicated bit-line. When data is being written on the WBL(i) line, all of the buffers connected to WBL(i) will be initialized concurrently. However, as the storing element is separated from the WBL signal in each memory cell, it will not be affected. After activating the WWL(j) line, only the driving buffer data connected to WWL(j) will be transferred to the storing element. When RWL(j) is activated, the storing element's data will be read and transferred to BLs.

WBL voltage depends on the array size, i.e., the equivalent resistance of all reading paths of all driving buffers and the wire loads along with the bit-line when the WBL voltage can provide at least a 5μA. The bit-line current can be set higher than 5μA to reduce the probability of write failure

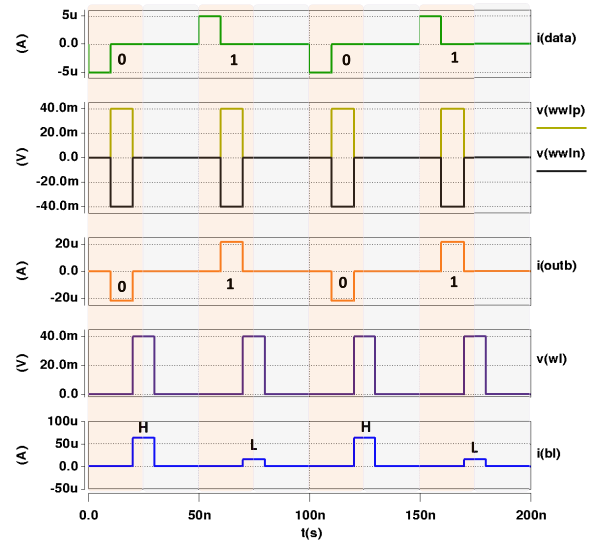


FIGURE 9. Inputs/outputs waveforms of an all-magnetic mCell-based bitcell.

(by providing safe margins through separated writing and reading paths in the mCell). WWL voltages can also be increased to improve the switching of the storing element in the writing process. Given the fact that these voltages decline toward reading the driving buffers, their values should be determined to ensure that storing elements are switched correctly despite the voltage drop.

Along with the reading operations, an appropriate voltage is applied to the word-line of reading. As a result, the current flows through the reading path of the storing element. The magnitude of this current depends on the resistance of the storing element (RH or RL). This current is regarded as one of the majority gate inputs.

D. ARITHMETIC AND LOGIC UNIT (ALU)

In the following, we describe how the proposed processor can do arithmetic and logic operations.

1) LOGIC UNIT

Accordingly, when three memory cells of BL(l) (m_A(i, l), m_B(j, l), and m_C(k, l)) are selected as the inputs of the majority gate in the MagCiM array (Fig. 6), the reading path current of each mCell (Im_i) can be IOH or IOL based on the cell state (RH or RL). According to Kirchhoff's current law (KCL), the sum of currents entering any junction is equal to the sum of currents leaving the junction. Therefore, the output current of BL(l) will be the resultant of Im_A(i, l), Im_B(j, l), and Im_C(k, l) (I_{sum} = Im_A(i, l) + Im_B(j, l) + Im_C(k, l)). I_{sum} is converted into voltage through Mn transistor, then the result of the majority gate is revealed through the voltage threshold detector.

2) ARITHMETIC UNIT

As mentioned earlier, all previously proposed CIMA architectures only support logic bit-wise operations inside the

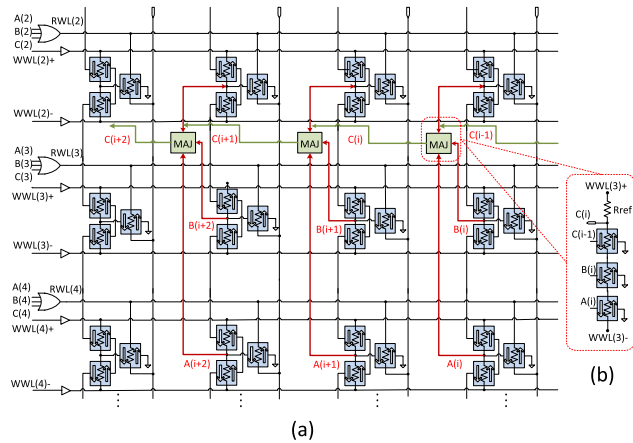


FIGURE 10. Carry computation and propagation for ADD operation.

memory. The previous CIM architectures commonly embed arithmetic circuits in the peripheral circuitry of the memory to perform arithmetic operations. The arithmetic circuits are either based on CMOS or hybrid spin/CMOS circuits. However, the MagCiM processor supports both logic and arithmetic bit-wise operations using only magnetic cells. In the following text, we will explain how the MagCiM performs primary arithmetic operations, including addition, subtraction, multiplication, and division.

a: ADDITION OPERATION

A ripple carry adder adds the bits of the same significance and the carry-in from the previous stage and propagates the carry-out to the next stage. The relations between the inputs and the outputs of any stage are expressed as $SUM(i) = A(i) \oplus B(i) \oplus C(i - 1)$ and $C(i) = A(i) \bullet B(i) + A(i) \bullet C(i - 1) + B(i) \bullet C(i - 1)$.

In our processor, $C(i)$ is directly generated in memory, and $SUM(i)$ is generated using XOR operation performed in memory without requiring any peripheral circuit at the edge of memory. Fig. 10 shows the hardware and wiring to carry computation and propagation. Location $M2$ of the memory is considered as special location to store the result of the ripple carries. According to Eq. 6, the majority circuit calculated the carry of i -th stage.

$$C(i) = Majority(A(i), B(i), C(i - 1)) \quad (6)$$

The majority circuit used in MagCiM (shown in Fig. 10(b)) serially connects three mCells $A(i)$, $B(i)$, $C(i-1)$ with the R_{ref} of 562Ω . In fact, the majority circuit consists of two separate parts: a reference resistance (R_{ref}) and a pull-down resistance. It also includes two opposite source/sink output currents (I_{Src} , I_{Sink}). Source and sink currents are obtained by comparing the pull-down resistance with the Ref resistance. Higher and lower values than the reference resistance directly influence the output ($C(i)$). If pull-down resistance is more than Ref resistance, the reference current would be greater than pull-down current; so that the output current turns into a source current ($I_{OUT} = I_{Src}$). Likewise, if the pull-down resistance is less than Ref resistance, the output current turns

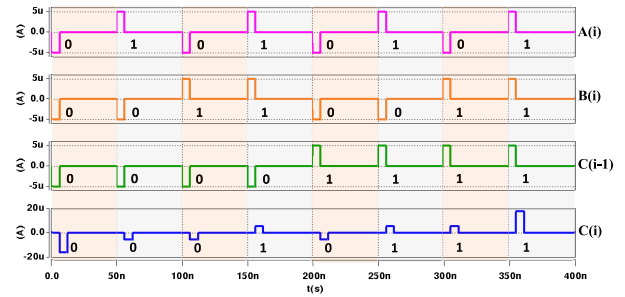


FIGURE 11. Normal operation of an MagCiM majority circuit.

into a sink current ($I_{OUT} = I_{Sink}$). The reference resistance provides a current between $WWL(3)+$ and the output whenever the output of the majority circuit becomes “1” (based on the inputs). Similarly, the pull-down resistance provides a current between the output and $WWL(3)-$ whenever the output of the majority becomes “0” (based on the inputs). Fig. 11 shows the regular operation of a MagCiM majority circuit.

In order to do an ADD operation, in the first step $M4$ (A) is initialized; $M3$ (B) is then initialized and $M2$ (Carry) is directly generated. According to Eq. 7, the SUM can calculated using three MAJ operations.

$$SUM(i) = Majority[A(i), \overline{C(i)}, Majority[\overline{C(i)}, B(i), C(i - 1)]] \quad (7)$$

Table 3 demonstrates how the ADD operation can be done using a majority instruction. The result of the addition is stored in a location of memory, such as Mj .

b: SUBTRACTION OPERATION

In MagCiM, we convert the subtraction operation into add operations using 2’s complement. Difference $A - B$ is calculated as addition i.e., $A + (-B)$ that is equal to $A + \overline{B} + 1$.

c: MULTIPLICATION AND DIVISION OPERATIONS

To avoid design complexity in CIM architecture, the multiplication/division operations are performed by repeated additions/subtractions. $A \times B$ is done by adding A repeatedly B times. Likewise, $A \div B$ is performed by subtracting the divisor until the remainder is less than the divisor.

d: OTHER OPERATIONS

Other operations such as Square Root, Logarithmic, Exponential, Hyperbolic, and Trigonometric functions can be done by the CORDIC (COordinate Rotation DIgital Computer) algorithm [29]. CORDIC is a simple and efficient algorithm to calculate the mentioned operations using only iterative shift-add operations.

3) COMPUTATION FLOW IN MagCiM

To explain the computation flow of the MagCiM processor, we have set up two subsequent case studies on the execution of i) logic operations (AND/NAND and OR/NOR operations)

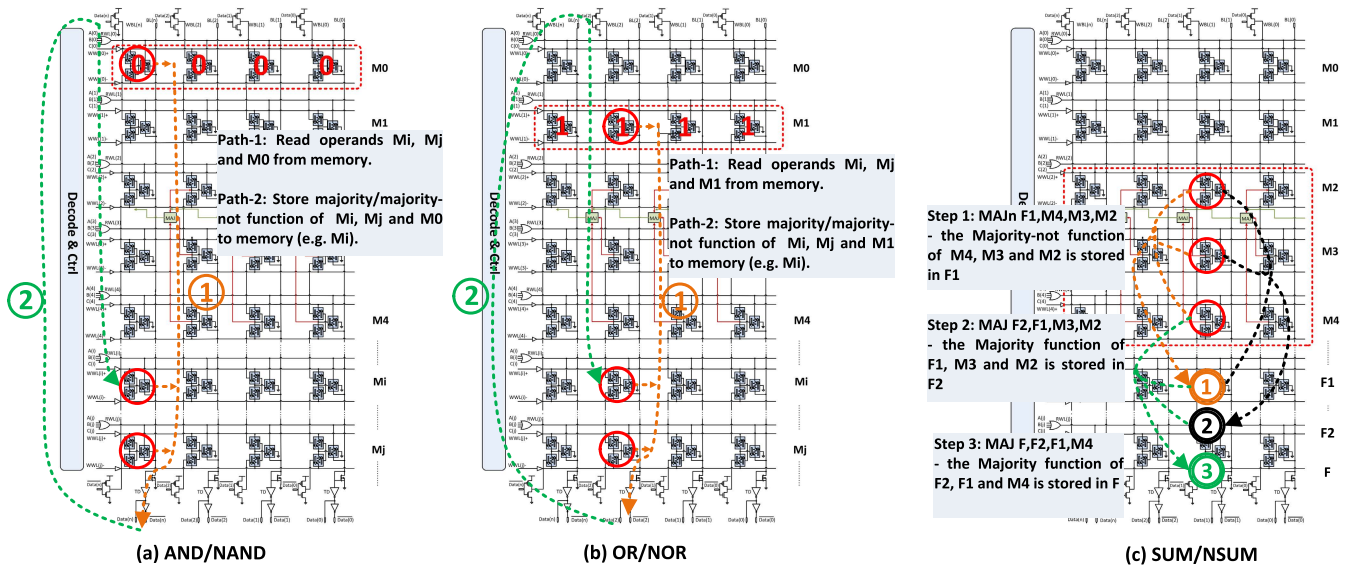


FIGURE 12. MagCiM computation flow: (a) AND/NAND current path. (b) OR/NOR current path. (c) SUM/NSUM current path.

and ii) arithmetic operations (SUM/NSUM operations) on the MagCiM processor.

a: CASE STUDY ON LOGIC OPERATIONS

For the sake of clarity, we briefly explain the overall computation flow for AND/NAND operation in MagCiM. The majority operation performs AND/NAND operations on operands stored in M_i and M_j (first it performs on M_0 , M_i , and then M_j). M_0 is reserved in MagCiM to always keep an all '0' value. Using the decoders A, B, and C, the currents associated with M_0 , M_i , and M_j are passed through BLs. The MAJ operation is performed by the Threshold Detector (TD) through path 1. Finally, by enabling $WWL(i)$, the result is stored in M_i via path 2. The same scenario performs an OR/NOR operation, except the M_1 entry that holds all '1' values used in the MAJ operation (See Fig. 12(b)).

b: CASE STUDY ON ARITHMETIC OPERATIONS

Fig. 12(c) shows the computation flow for a SUM operation used in the ADD instruction. An addition operation has two outputs, namely, SUM and Carry-out. The SUM is computed by XORing the inputs and Carry-in. We need to perform XOR operation on three operands for producing SUM, e.g., M_2 , M_3 , and M_4 . MagCiM performs this in two steps. In the first step, a majority operation processes M_2 and M_3 , and stores the result in M_i (Path 1 of Fig. 12(c), 2). In the second step, the second majority operation processes M_i and M_4 , and stores the result in M_j (Path 2 and 3 in Fig. 12(c)).

VI. THE ARCHITECTURE OF THE PROPOSED MagCiM PROCESSOR

Fig. 13 indicates the structure of the proposed MagCiM single-cycle processor designed using all-magnetic magnetoresistive RAM. The MagCiM is a 32-bit RISC processor

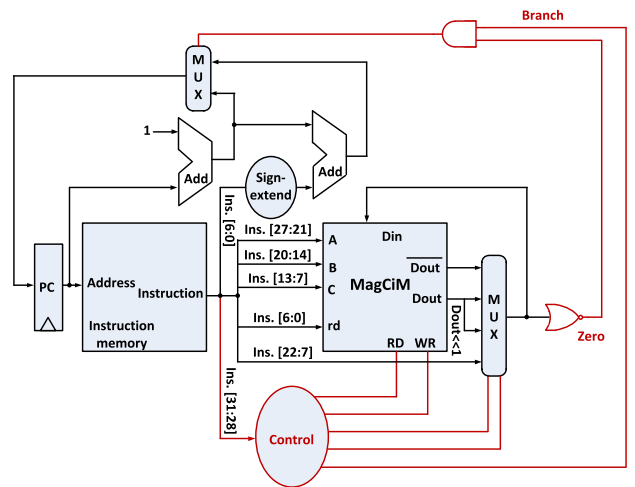


FIGURE 13. The structure of the proposed magnetoresistive-based computing-in-memory processor (MagCiM).

that executes instructions without needing any register file and ALU. Based on the fact that the majority operation is a logically complete operation, i.e., other operations can be restated using only majority expressions [28], the MagCiM processor uses the majority gates for its computations. Table 3 shows the basic instructions of the proposed MagCiM processor based on the majority gate. The MagCiM single-cycle processor performs all tasks of every instruction in one clock cycle. A new instruction is fetched only when the execution of the previous is finished. Execution of every instruction takes the following four steps.

- Instruction fetch (IF): the instruction is fetched from the instruction memory, and the address of the next instruction is generated.
- Instruction decode (ID): the instruction is decoded, and the operands are being read.

TABLE 3. The basic instructions of the proposed MagCiM processor.

Function	High-level language	Assembly language	Description	Encoding	B	OP
No-operation	NOP;	NOP	No Operation Instruction	-	-	111
NOT	F=not A;	MAJn F,A,M0,M1	Majority-not function	M	0	000
AND	F=A and B;	MAJ F,A,B,M0	Majority Function	M	0	001
NAND	F=A nand B;	MAJn F,A,B,M0	Majority-not function	M	0	000
OR	F=A or B;	MAJ F,A,B,M1	Majority Function	M	0	001
NOR	F=A nor B;	MAJn F,A,B,M1	Majority-not function	M	0	000
SLL	F=A<<1;	MAJs F,A,0,M1	Shift Left Logical	M	0	010
LW	F=A;	MAJ F,A,M0,M1	Load Word	M	0	001
Li	F=imm;	Li F,imm	Load Immediate	I	0	011
BEQ	If (A==B) goto L1;	MAJ F1,A,B,M0	Majority Function	M	0	001
		MAJn F2,A,B,M1	Majority-not function	M	0	000
BNE	If (A!=B) goto L1;	jMAJz L1,F1,F2,M1	Jump if MAJ=0	B	1	100
		MAJ F1,A,B,M0	Majority Function	M	0	001
		MAJn F2,A,B,M1	Majority-not function	M	0	000
ADD	F=A+B;	jMAJnz L1,F1,F2,M1	Jump if MAJ !=0	B	1	101
		MAJn F1,M4,M3,M2	Majority-not function	M	0	000
		MAJ F2,F1,M3,M2	Majority Function	M	0	001
		MAJ F,F2,F1,M4	Majority Function	M	0	001

- Memory access and EXecution (MEX): the memory (MagCiM) is accessed based on the computed addresses, and the majority function is executed.
- Write back (WB): the load operation is completed by writing the result from memory (MagCiM) into the memory (MagCiM) itself.

All the essential components of the MagCiM processor (including PC, Instruction Memory, and MagCiM) are designed non-volatile to support normally-off computing [30]. In the following, we will describe the functionality and structure of these components in detail.

The proposed processor has two memories: MagCiM and instruction memory. However, in the pipeline version, the existence of only one memory for both data and instructions may result in Structural Hazards. To tackle this issue, we have used two separated memories one for data and one for the instruction.

A. NORMALLY-OFF COMPUTING

Normally-off computing (NoC) is one of the promising power-saving techniques that is capable of offering extra power saving on top of existing low-power techniques such as dynamic voltage and frequency scaling, clock gating, and power gating can. NoC has zero standby power consumption during idle time and instant-on characteristics [30] that are commonly used in numerous Internet of Things (IoT) applications with long-term sleep duration e.g., wearable health-care devices, wireless sensing, etc.

In general, the total power consumption of an IoT device can be described with Eq. 8, where P_{Active} , P_{Sleep} , and P_{Wakeup} are respectively the power consumption during active mode, sleep mode, and wake-up transition:

$$P_{Total} = P_{Active} + P_{Backup} + P_{Sleep} + P_{Wakeup} + P_{Restore} \quad (8)$$

The weight of each parameter in the total power consumption severely depends on the application. In some applications, the

system will spend most of its time in sleep mode. Accordingly, a low sleep-power consumption is more critical design issue. On the other hand, for applications such as data loggers, the device will often switch between active and sleep modes. In that case, the wake-up energy has to be reduced. When non-volatile memory is utilized in the computation device, P_{Backup} and $P_{Restore}$ are completely removed. This significantly decrease the total energy consumption when the frequency of switching between the sleep and the wake-up state are high. Briefly, the total energy consumption when using MagCiM is given by Eq. 9:

$$E_{MagCiM} = P_{Active} \times T_{Active} + P_{Wakeup} \times T_{Wakeup} \quad (9)$$

where T_{Active} and T_{Wakeup} are the average time duration in which the device is in active and wake-up state. The total energy consumption when using MIPS is given by Eq. 10:

$$E_{MIPS} = P_{Active} \times T_{Active} + P_{Backup} \times T_{Backup} + P_{Sleep} \times T_{Sleep} + P_{Wakeup} \times T_{Wakeup} + P_{Restore} \times T_{Restore} \quad (10)$$

Comparing Eq. 9 with Eq. 10, it can be seen that the MagCiM will is more energy efficient essentially with increasing T_{Sleep} .

B. INSTRUCTION SET ARCHITECTURE

The 32-bit instruction set architecture (ISA) of the MagCiM processor contains three classes: M – type (for Memory), B – type (for Branch), and I – type (for Immediate). The MagCiM instruction format for each of the classes is shown in Fig. 14. Common fields for all three formats include i) the four most-significant bits of the instruction that are used for the operation code and ii) the branch bit. The instructions OP bits [30:28] are sent to a control unit to determine the type of instruction; the type of instruction then determines which control signals are to be set, i.e., the instruction decode. If the

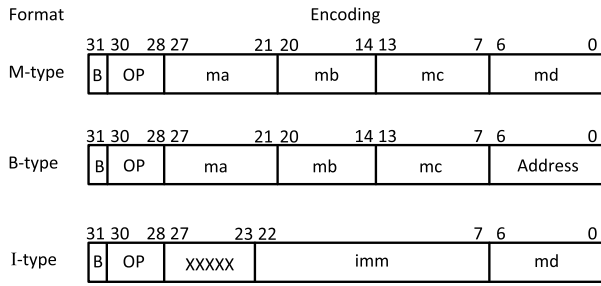


FIGURE 14. MagCiM instruction formats.

TABLE 4. The XOR and XNOR operations based on majority function.

Function	XNOR ($F=A \odot B$)	XOR ($F=A \oplus B$)
Instruction set	MAJ F1,A,B,M0	MAJ F1,A,B,M0
	MAJn F2,A,B,M1	MAJn F2,A,B,M1
	MAJ F,F1,F2,M1	MAJn F,F1,F2,M1

branch bit is set to '1', the content of *md* field is set to the PC-relative address of a memory location.

M-type MagCiM instructions are used for arithmetic and logic operations. These instructions consist of four memory location fields. *ma*, *mb*, and *mc* are three sources fields, and the *md* is the only destination field. The memory address fields *ma* (bits [27:21]), *mb* (bits [20:14]), *mc* (bits [13:7]), and *md* (bits [6:0]) are used to address the MagCiM memory. The MagCiM memory performs three independent memory read operations and one memory write operation in one clock cycle. Read operations are performed in the first half of the clock cycle while write operations are done in the second half.

B-type MagCiM instructions are used for implementing conditional operations and loops. The format of B-type instructions is similar to that of the M-type instructions unless the *md* address field of M-type instructions here is interpreted as the PC-relative address of a memory location.

I-type MagCiM instructions are used for writing data in a MagCiM location. Per this format, the immediate data (a 16-bit constant value) is stored in bits [22:7].

Table 3 shows the basic instructions the MagCiM processor supports. Worth to mention that other operations can be done as a combination of instructions of Table 3. Table 4 demonstrates how, for example, XNOR and XOR operations can be done using a majority instruction. An XOR gate is used for the implementation of the BNE instruction. If two operands are the same, the XOR operation returns zero that, according to control signals (see Fig. 13, the NOR-gate output), the branch will not be taken. The SLL instruction (shift left logical, $Dout \ll 1$ in Fig. 13) is done by locating a wire of '0' at the least significant bit of *Dout* and ignoring the most significant bit of *Dout*.

C. MagCiM ADDRESSING MODES

The ISA specifies the mechanisms by which a processor accesses operands for calculations. Fig. 15 shows how operands are identified for three addressing modes of the

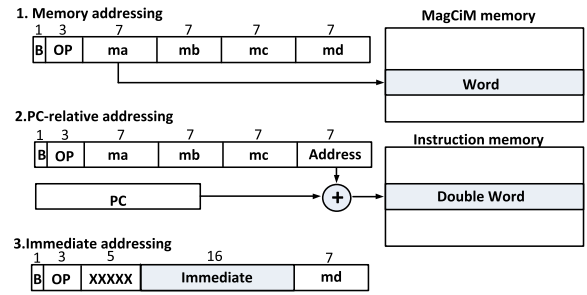


FIGURE 15. Illustration of the three MagCiM addressing modes. The operands are shaded in color.

MagCiM processor. Notable to mention that the MagCiM supports having more than one addressing mode in a single instruction. For example, jMAJz instruction uses both PC-relative and memory addressing modes.

1. In the *memory addressing mode*, the operand is stored in a memory location.
2. In *PC-relative addressing mode*, the branch address is obtained as the summation of PC and a constant value embedded in the instruction.
3. In *immediate addressing mode*, the full target address is embedded as a constant value within the instruction.

VII. NORMALLY-OFF COMPUTING IN MagCiM PROCESSOR

To support normally-off computing in the MagCiM processor, we have proposed a non-volatile instruction memory (IM) and the program counter (PC), which are shown in Fig. 16 and Fig. 17. The IM cells are similar to MagCiM cells. The IM array benefits from one decoder. This decoder's output allows the stored bits of a row to be selected simultaneously and applied to current-to-voltage (I-V) converters. In the I-V converter (Fig. 16(c)), the Mn transistor converts Im_{Ai} into a voltage and transfers it to an element called the voltage threshold detector (TD). The result finally appears in the IM output.

The proposed 8-bit Non-volatile PC is composed of four components: 1. Write circuits to program mCell-devices; 2. mCell-devices to store the address; 3. Sense Amplifiers (SA) to read address stored in mCell-devices according to their resistances; 4. SR-latches (Slave) to provide the next address. The mCell device is utilized as a storage element in the PC register. The writing and reading process of the PC register is controlled by the signal "CLK". When "CLK" is high, the address is stored in mCells by write circuits in the form of different resistance levels, e.g., high resistance state stands for logic '1' and vice versa; meanwhile, the slave latches keep the precedent address. When "CLK" is low, the sense amplifiers read the stored address, and the slave latches become transparent and update their output address.

VIII. PIPELINING IN MagCiM PROCESSOR

To enhance the MagCiM processor's performance, we have also designed a 3-stage pipelined architecture for the MagCiM processor (see Fig. 18).

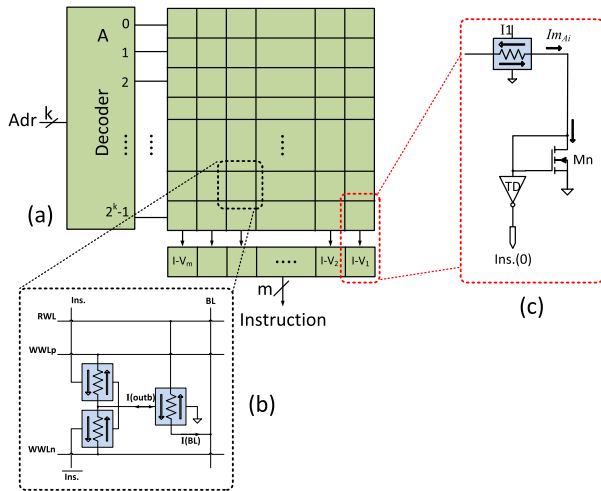


FIGURE 16. The proposed instruction memory (IM) employed in our proposed MagCiM processor. a) IM structure. b) Bit-cell structure. (c) Current-to-voltage (I-V) converter.

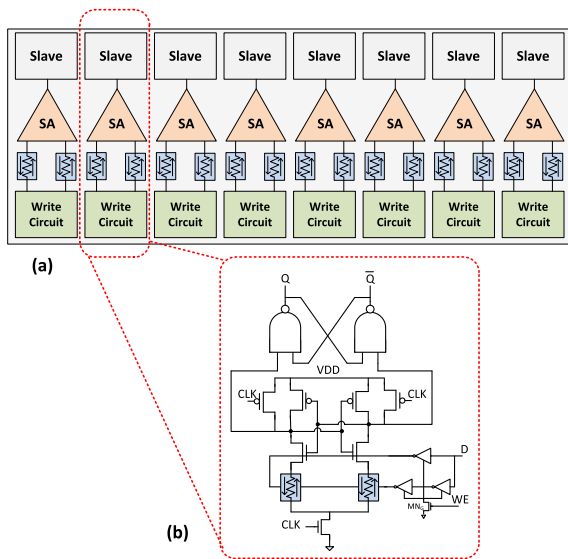


FIGURE 17. The proposed non-volatile PC. (a) Schematic view. (b) Bit-cell structure.

TABLE 5. Data fields in the pipeline registers.

Pipeline register	Data fields
IFD/MXW	PC+1, sign extended dest, ma, mb, mc, md, imm, control signals
MXW/BR	PC+1, sign extended dest, branch signal

- 1) Instruction Fetch and Decode (IFD) stage,
- 2) Memory, eXecution, and Writeback (MWX) stage,
- 3) Branch (BR) stage.

In Fig. 18, IFD/MXW and MXW/BR are pipeline registers that connect consecutive pipeline stages by holding data only for one clock cycle. Table 5 shows data fields held by each pipeline register.

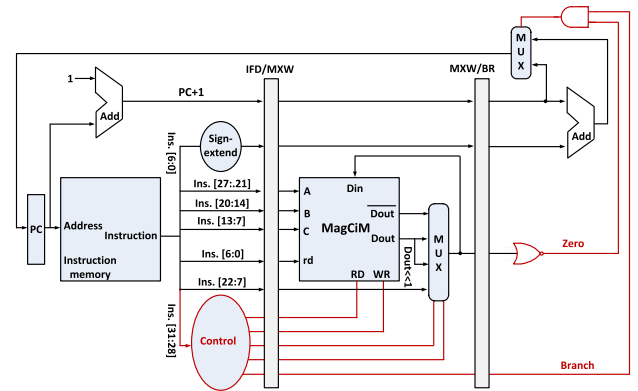


FIGURE 18. The pipelined datapath of Fig. 13, with the control signals connected to the control portions of the pipeline registers.

A. IFD STAGE

The IFD stage consists of the PC register, a simple adder, the instruction memory, and the control unit. At each clock cycle, the PC increments to address the instruction memory to fetch a new instruction. This stage also decodes the operands according to the operation code. The control signals are generated by the control unit and passed to the MXW stage along with PC+1 via the IFD/MXW pipeline register.

B. MXW STAGE

Based on the operation code and other instruction bits, the controller generates A, B, and C address lines for MagCiM. The data D1, D2, D3 are then passed to majority gates. Finally, the majority operation is performed. This process is done in the first half of the clock cycle when the MagCiM read signal (RD) is high. The result is then written into the MagCiM, corresponding to the write address rd, during the second half-cycle when the MagCiM write signal (WR) is high. The address, PC+1, and the branch signal are all written into the MXW/BR pipeline register.

C. BR STAGE

This stage is used to obtain the next instruction address for the branch instructions. In this stage, the zero flag and branch address are calculated, and the result is written into the PC.

Like any other pipelined processor, there are situations in which the MagCiM pipeline needs to be stalled due to the structural, control, and data hazards. The straightforward solution to resolve data or branch hazards is to insert NOP ('no operation' in Table 3) instructions at the assemble/link time. The assembler/compiler inserts NOPs after branch and before data-dependent instructions.

IX. SIMULATION SYSTEM SETUP AND RESULTS

To evaluate the MagCiM processor, we have employed the same method as [18] (See Fig. 19). First, Circuit level simulations are done using a Verilog-A model proposed in [31] for mCells. Then, mCells and interface CMOS circuits are co-simulated in Cadence Spectre and HSPICE tools. The MagCiM processor is developed at the circuit level

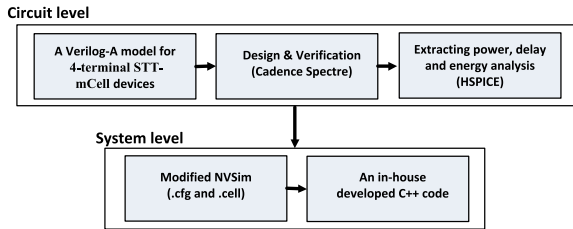


FIGURE 19. Circuit and system level simulation framework.

TABLE 6. Effect of process variation on MagCiM.

variation	$\pm 0\%$	$\pm 5\%$	$\pm 10\%$	$\pm 15\%$	$\pm 20\%$	$\pm 25\%$
% Failures	0.00%	0.00%	0.18%	5.31%	12.43%	21.58%

using the HSPICE tool. Simulations are carried out using 32nm Predictive Technology Model (PTM) technology with a nominal voltage of 0.9v. For the system-level simulations, we employ a modified self-consistent NVSim [32] along with an in-house developed C++ code for our power consumption and occupied area evaluations.

A. RELIABILITY ANALYSIS

In order to analyze the read failures under process variation of mCells for MagCiM operations, we performed a Monte-Carlo circuit-level simulation. We increase the amount of process variation from $\pm 5\%$ to $\pm 25\%$ and run 100,000 simulations for each level of process variation. Table 6 shows the percentage of iterations in which MagCiM operates incorrectly for each level of variation. Two conclusions are reached: First, as expected, up to $\pm 5\%$ variation, there are zero errors in MagCiM; and second, even with ± 10 and ± 15 variation, the percentage of erroneous MagCiM across 100,000 iterations each is just 0.18% and 5.31%. These results show that MagCiM is reliable even in the presence of significant process variation.

The reliability of an STT-MRAM device depends on the following failure mechanisms: read disturb (MTJ flipping during read), write failure (incorrect write operation), and decision failure (incorrect sensing of MTJ value). An efficient way to solve the read disturb problem is to separate the read and write paths of MTJ devices [33]. For this purpose, we have used “mCell”. Write failure occurs when the MTJ device’s write current is less than its critical current. Write failure rate can be decreased by increasing the write pulse duration or the write current value [33]. In mCell, the critical current is 5uA. we have set the write current to 10uA to reduce the write failure rate in the MagCiM processor. Decision failure occurs in STT-MRAMs in which sense amplifiers are used to read the values of the MTJs by comparing the MTJs resistance with a reference resistance [33]. Decision failure will not happen in MagCiM because sense amplifiers are not used to extract the result of the cells.

B. MEMORY PERFORMANCE EVALUATION

According to Table 7, the proposed MagCiM shows the least read dynamic energy in comparison to other designs.

TABLE 7. SRAM, DRAM, and MagCiM memory validation and comparison for a sample 4MB memory.

Metrics	SRAM		STT-MRAM		MagCiM	
	Write	Read	Write	Read	Write	Read
Average Latency (ns)	1.53		11.13	2.62	1.84	1.13
Dynamic Energy (pJ)	297.46	312.56	1219	791.13	442.75	287.35
Leakage Power (mW)	5258		830.3		627.82	
Area (mm ²)	10.544		4.611		6.562	

In addition, MagCiM reduces the total leakage power compared to SRAM. Although, the proposed MagCiM shows longer average latency compared to SRAM due to the longer write latency of magnetic memory storage. Moreover, the area overhead of the proposed MagCiM is 42.31% more than STT-MRAM but still 37.76% less than SRAM design. It should be noted that the first and foremost advantage of spintronic memories, compared to SRAM, is their non-volatility with almost 10 years’ retention time [5].

C. MagCiM VERSUS A TYPICAL MIPS

We have compared the MagCiM processor with a typical MIPS processor [34] under various instructions (ADD, SUB, MUL, DIV, SQRT, and EXP) in terms of instruction execution latency, the leakage power consumption, and the energy consumption. To this end, we have implemented a typical MIPS processor at the circuit level using the HSPICE tool under 32nm PTM CMOS technology.

As the MagCiM processor does not have any register file and ALU, it is highly area-efficient compared to a typical RISC processor like MIPS. Using 32nm technology, MagCiM exhibits about 76% less occupied area than the MIPS processor. The MagCiM exploits mCells (a particular type of MTJ-based memory cell) to cope with the leakage power consumption issue. Fig. 20.a shows the Leakage power consumption of the MagCiM versus its competitor, the MIPS processor. According to our results, almost 98% leakage power saving is achieved by the MagCiM processor, mainly because of magnetic memory and computation. In addition, as the instruction memory, the data memory, and the program counter are non-volatile, the MagCiM is a normally-off enabled processor.

Latency in our evaluations is measured as the product of the number of clock cycles required for executing an instruction and the clock signal period. Fig. 20.b confirms that the MagCiM processor offers a significant lower delay in all instructions than the MIPS processor. On average, it exhibits 58% improvement in latency as compared to the MIPS processor. The energy per instruction is measured by the product of the execution latency and the instruction’s power consumption. According to Fig. 20.c, the MagCiM processor consumes much less energy than that of the MIPS processor.

D. COMPARISON WITH THE PREVIOUS WORK

Although NVM-based CIM architectures provide substantial performance improvement and energy efficiency for data-intensive applications, they are vulnerable to Data

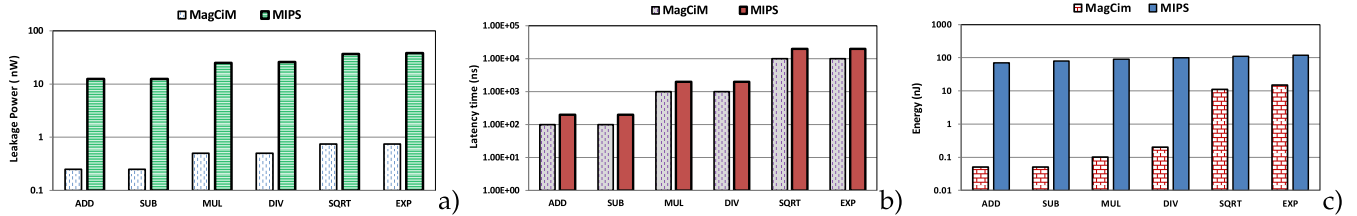


FIGURE 20. Operation leakage power (a), latency (b), and energy (c) for the MagCiM and MIPS processors.

TABLE 8. The clock-rate and clocks comparison of different implementations for 128-bit AES.

Parameters	GPP [35]	CMOS ASIC [36]	Pipelined ASIC [37]	Memristive CMOL [38]	Baseline DW [39]	Pipeline DW [39]	Multi-issue DW [39]	Parallelism level P1 [18]	Parallelism level P2 [18]	Parallelism level P4 [18]	MagCiM	Pipeline MagCiM
Clocks	2309	336	672	470	1022	2652	1320	4176	2168	1084	905	1215
Clock-rate	2GHz	30MHz	30MHz	33MHz	30MHz	30MHz	30MHz	32MHz	32MHz	32MHz	1.5GHz	1.5GHz

Remanence Attacks (DRA) as they may hold persistent data. Therefore, it makes sense to employ some sort of encryption algorithm. In this subsection, we compare the MagCiM processor with different architectures proposed in the previous work. To keep results comparable, we use the same case study benchmark as the earlier works have used, i.e., Advanced Encryption Standard (AES) algorithm. The selected architecture for this comparison are:

- I) The proposed Computing-in-memory Edge (CIME) architecture in [18], which is based on 4-terminal spin Hall effect-driven domain wall motion devices. In this work, three different parallelism levels for the proposed CIME architecture have been proposed, which are referred to as **P1**, **P2**, and **P4** configurations. Similar to our work, a Verilog-A model of their MTJ-based devices is developed to co-simulate with the interface CMOS circuits in Cadence Spectre and SPICE. Authors have also used the 45nm North Carolina State University (NCSU) Product Development Kit (PDK) library for their SPICE simulations [18].

II) **ASIC**: The proposed AES hardware accelerator in [36] fabricated in 22nm CMOS technology is the next candidate for this comparison experiment. Although this research work is not in the context of CIM, we have intentionally selected it to compare the MagCiM processor’s efficiency with ASIC designs.

III) **CMOL** is a hybrid CMOS-Nano circuit proposed in [38] for the implementation of the AES algorithm. To evaluate CMOL circuit, HSPICE simulations were conducted using a CMOS 45nm technology.

IV) A special-purpose spintronic based CIME architecture designed to perform AES encryption [39], the so called **DW-AES**. To evaluate DW-AES, authors have used SPICE simulations using 32nm technology. Three different configurations of DW-AES have been proposed in this research namely, **Baseline**, **Pipeline**, and **Multi-issue DW-AES** [39].

V) The MIPS [34] and the pipeline MIPS processors [34] that we used in the previous subsection are used as in this comparison as well.

VI) Finally, a software implemented AES algorithm on a general purpose processor (GPP) [35] is also used in our comparisons.

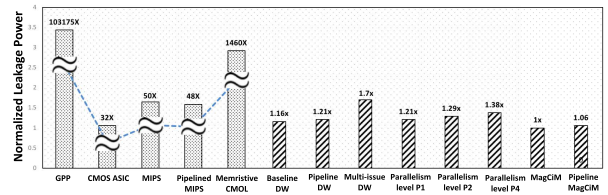


FIGURE 21. Normalized leakage power consumption in different architectures using AES encryption algorithm.

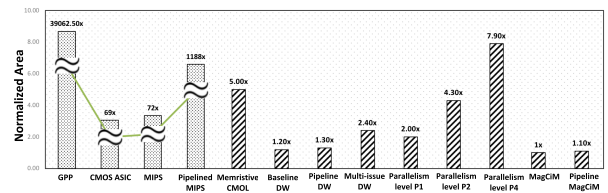


FIGURE 22. Normalized occupied area in different architectures using AES encryption algorithm.

We have compared the MagCiM processor with the mentioned architectures in terms of energy consumption, occupied area, number of clock cycles, and leakage power dissipation. In our comparisons with the mentioned designs, we have used the results reported in their corresponding papers. As mentioned earlier, for the MIPS processors, we have used the simulation results of our MIPS implementations. To be consistent with the previous work, we have carried out our SPICE simulations for the 32nm Predictive Technology Model (PTM) for MagCiM and MIPS processors.

The Fig.21, Fig.22, and Fig.23 show the leakage power dissipation, the occupied area, the energy consumption results, respectively (all results are normalized to that of MagCiM processor). As shown in Fig.21, Fig.22, and Fig.23, both MagCiM and the Pipeline MagCiM consume less leakage power, occupies less area, and dissipate less energy than all of the other architectures. To accurately compare the MagCiM processor with different architectures in terms of performance, we should measure the total execution time, i.e., product of the total number of clock cycles and the clock period. As shown in Table 8, except for the CMOS ASIC design of AES [39], MagCiM has less execution time than the other architectures.

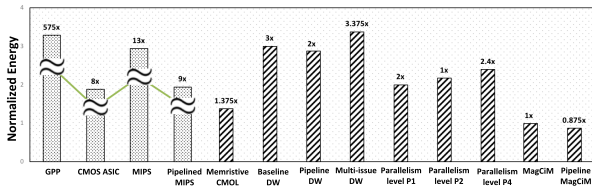


FIGURE 23. Normalized energy consumption in different architectures using AES encryption algorithm.

E. A DISCUSSION ON EFFICIENCY AND APPLICATIONS OF MagCiM

It should be noted that, MagCiM can be used as a tiny embedded microcontroller. In most of tiny embedded microcontrollers, there is no hardware multiplier or divider (e.g. ATmega128, Nios II Processor, iCE40HX8K FPGA). In such cases, the missing logic is usually implemented as an assembly language routine and available as part of the standard library. Therefore, such embedded processor including MagCiM takes much more time to perform scientific computations as compared to processors designed and optimized for high performance computation. However, in embedded system applications such as edge devices where power consumption and cost are serious concerns, the employment of such processors is inevitable. Firstly, in embedded applications, as opposed to high performance applications, the computations are not complex. Therefore, there are very limited number of complex operations with large operands in such applications. Secondly, most of the embedded systems are battery operated systems which use a harvesting approach to charge the battery. Therefore, the power consumption and being robust against power shortage are very crucial factors. In MagCiM processor, we have addressed these two important factors by using pure magnetic cells. MagCiM non-volatility feature (originating from the non-volatility of PC, Instruction Memory, and Data Memory), protects data against temporary power outages due to some noises or the battery shortage because of unavailability of harvesting resources. Thus, after energy becomes available, the reading and execution of instructions continues from the outage time. While in volatile processors such as MIPS, in the event of a temporary power outage, the algorithm must be restarted from the beginning; Therefore, the non-volatile feature plays a major role in saving power and reducing the execution time. The higher the number of outages, the more the difference between MagCiM and other processors performance is magnified.

Main advantages of the proposed MagCiM are:

1) The MagCiM processor uses pure magnetic memories. In other words, for the implementation of memory cells, only mCell devices have been used. This makes the implementation of MagCiM easier and less expensive compared to the previous work (in the previous work, a combination of MTJ and transistors has been used in the structure of memory cells).

2) As opposed to the previous works that employ MTJ devices. In MagCiM, Read Disturb does not occur due to the use of mCell in which the write and the read paths are completely separated.

3) Previous works have employed a sense amplifier to extract the output, while MagCiM uses a majority gate. This design prevents Decision Failure. In addition, fewer devices are used in its implementation as compared to sense amplifiers.

4) As MagCiM does not use Sense Amplifiers, It is significantly robust against Single Event Upsets (SEUs) and Decision failure. In addition, process variation has fewer effects on MagCiM.

X. CONCLUSION

In this paper, a novel Computing-In-Memory processor was proposed and evaluated. The MagCiM processor supports normally-off computing based using an all-magnetic mCell-based memory. The paper combines the advantages of both non-volatile memory and computation-in-memory architecture to offer a low-power and high-performance processor. The throughput of the MagCiM processor was also boosted using the proposed pipeline architecture. Circuit-level simulation results revealed that the MagCiM processor consumes significantly less leakage power and energy consumption and occupies less area as compared to the previously proposed methods while also offering the Normally-off computing capability which is a very viable computing paradigm for design of ultra-low power cyber-physical embedded systems.

REFERENCES

- [1] G. Dai, T. Huang, Y. Chi, J. Zhao, G. Sun, Y. Liu, Y. Wang, Y. Xie, and H. Yang, "GraphH: A processing-in-memory architecture for large-scale graph processing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 4, pp. 640–653, Apr. 2019.
- [2] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing data where it makes sense: Enabling in-memory computation," *Microprocessors Microsyst.*, vol. 67, pp. 28–41, Jun. 2019.
- [3] W. Banerjee, "Challenges and applications of emerging nonvolatile memory devices," *Electronics*, vol. 9, no. 6, p. 1029, Jun. 2020.
- [4] R. Zhang, T. Liu, K. Yang, C.-C. Chen, and L. Milor, "SRAM stability analysis and performance–reliability tradeoff for different cache configurations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 3, pp. 620–633, Mar. 2020.
- [5] X. Fong, Y. Kim, S. H. Choday, and K. Roy, "Failure mitigation techniques for 1T-1MTJ spin-transfer torque MRAM bit-cells," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 2, pp. 384–395, Feb. 2014.
- [6] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori, "Evaluation of hybrid memory technologies using SOT-MRAM for on-chip cache hierarchy," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 3, pp. 367–380, Mar. 2015.
- [7] D. M. Bromberg, H. E. Sumbul, J.-G. Zhu, and L. Pileggi, "All-magnetic magnetoresistive random access memory based on four terminal mCell device," *J. Appl. Phys.*, vol. 117, no. 17, p. 17B510, 2015.
- [8] V. Jamshidi and M. Fazeli, "Design of ultra low power current mode logic gates using magnetic cells," *AEU-Int. J. Electron. Commun.*, vol. 83, pp. 270–279, Jan. 2018.
- [9] V. Jamshidi, M. Fazeli, and A. Patooghy, "MGate: A universal magnetologic gate for design of energy efficient digital circuits," *IEEE Trans. Magn.*, vol. 53, no. 10, pp. 1–13, Oct. 2017.
- [10] R. Bishnoi, F. Oboril, and M. B. Tahoori, "Non-volatile non-shadow flip-flop using spin orbit torque for efficient normally-off computing," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 769–774.

- [11] D. Fujiki, S. Mahlke, and R. Das, "In-memory data parallel processor," in *Proc. 23rd Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, Mar. 2018, pp. 1–14, doi: [10.1145/3173162.3173171](https://doi.org/10.1145/3173162.3173171).
- [12] T. Li, Q. Wang, Y. Zhu, J. Jiang, G. He, J. Jin, Z. Mao, and N. Jing, "A novel resistive memory-based process-in-memory architecture for efficient logic and add operations," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 24, no. 2, pp. 25:1–25:22, Mar. 2019, doi: [10.1145/3306495](https://doi.org/10.1145/3306495).
- [13] L. Zhang, E. Deng, H. Cai, Y. Wang, L. Torres, A. Todri-Sanial, and Y. Zhang, "A high-reliability and low-power computing-in-memory implementation within STT-MRAM," *Microelectron. J.*, vol. 81, pp. 69–75, Nov. 2018.
- [14] X. Jiang, J. Bao, L. Zhang, and L. Bai, "A novel dual-reference sensing scheme for computing in memory within STT-MRAM," *Microelectron. J.*, vol. 121, Mar. 2022, Art. no. 105355.
- [15] S. Angizi, Z. He, and D. Fan, "PIMA-logic: A novel Processing-in-Memory architecture for highly flexible and energy-efficient logic computation," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, p. 162.
- [16] V. Nehra, S. Prajapati, T. N. Kumar, and B. K. Kaushik, "High-performance Computing-in-Memory architecture using STT/SOT-based series triple-level cell MRAM," *IEEE Trans. Magn.*, vol. 57, no. 8, pp. 1–12, Aug. 2021.
- [17] M. Zabihni, Z. I. Chowdhury, Z. Zhao, U. R. Karpuzcu, J.-P. Wang, and S. S. Sapatnekar, "In-memory processing on the spintronic CRAM: From hardware design to application mapping," *IEEE Trans. Comput.*, vol. 68, no. 8, pp. 1159–1173, Aug. 2019.
- [18] S. Angizi, Z. He, N. Bagherzadeh, and D. Fan, "Design and evaluation of a spintronic in-memory processing platform for nonvolatile data encryption," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 9, pp. 1788–1801, 2017.
- [19] F. Parveen, S. Angizi, and D. Fan, "Imflexcom: Energy efficient in-memory flexible computing using dual-mode sot-mram," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 3, p. 35, 2018.
- [20] H. Zhang, W. Kang, K. Cao, B. Wu, Y. Zhang, and W. Zhao, "Spintronic processing unit in spin transfer torque magnetic random access memory," *IEEE Trans. Electron Devices*, vol. 66, no. 4, pp. 2017–2022, Apr. 2019.
- [21] L. Xue, Y. Cheng, J. Yang, P. Wang, and Y. Xie, "Odyssey: A novel 3t-3mtj cell design with optimized area density, scalability and latency," in *Proc. 35th Int. Conf. Comput.-Aided Design*, 2016, p. 118.
- [22] D. Kobayashi, Y. Kakehashi, and K. Hirose, "Influence of heavy ion irradiation on perpendicular-anisotropy CoFeB-MgO magnetic tunnel junctions," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 4, pp. 1710–1716, Aug. 2014.
- [23] B. Wang, Z. Wang, K. Cao, Y. Zhang, Y. Zhao, and W. Zhao, "Radiation hardening design for spin-orbit torque magnetic random access memory," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–4.
- [24] C. Wang, Z. Wang, G. Wang, Y. Zhang, and W. Zhao, "Design of an area-efficient computing in memory platform based on STT-MRAM," *IEEE Trans. Magn.*, vol. 57, no. 2, pp. 1–4, Feb. 2021.
- [25] F. Parveen, S. Angizi, Z. He, and D. Fan, "IMCS2: Novel device-to-architecture co-design for low-power in-memory computing platform using coterminous spin switch," *IEEE Trans. Magn.*, vol. 54, no. 7, pp. 1–14, Jul. 2018.
- [26] S. Angizi, Z. He, A. Awad, and D. Fan, "MRIMA: An MRAM-based in-memory accelerator," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 5, pp. 1123–1136, May 2019.
- [27] S. M. Williams and M. Lin, "Architecture and circuit design of an all-spintronic FPGA," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2018, pp. 41–50.
- [28] M. Sarvaghad-Moghaddam, A. A. Orouji, and M. Houshmand, "A multi-objective synthesis methodology for majority/minority logic networks," *J. Comput. Electron.*, vol. 16, no. 1, pp. 162–179, Mar. 2017.
- [29] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vols. EC-8, no. 3, pp. 330–334, Sep. 1959.
- [30] M. Hayashikoshi, Y. Sato, H. Ueki, H. Kawai, and T. Shimizu, "Normally-off MCU architecture for low-power sensor node," in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2014, pp. 12–16.
- [31] D. Bromberg, "Current-driven magnetic devices for non-volatile logic and memory," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2014.
- [32] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSIM: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.
- [33] V. Jamshidi and M. Fazeli, "Pure magnetic logic circuits: A reliability analysis," *IEEE Trans. Magn.*, vol. 54, no. 10, pp. 1–10, Oct. 2018.
- [34] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design MIPS Edition: The Hardware/Software Interface*. Burlington, MA, USA: Morgan Kaufmann, 2020.
- [35] K. Malbrain. (2009). *Byte-Oriented-AES: A Public Domain Byteoriented Implementation of AES in C*. Accessed: Oct. 19, 2015. [Online]. Available: <https://code.google.com/p/byte-oriented-aes/>
- [36] S. Mathew, S. Satpathy, V. Suresh, M. Anders, H. Kaul, A. Agarwal, S. Hsu, G. Chen, and R. Krishnamurthy, "340 mv–1.1 v, 289 gbps/w, 2090-gate nanoaes hardware accelerator with area-optimized encrypt/decrypt gf(2⁴)² polynomials in 22 nm tri-gate CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 4, pp. 1048–1058, Apr. 2015.
- [37] S. Mathew, F. Sheikh, and A. Agarwal, "53 Gbps native GF(2⁴)² composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 767–776, May 2011.
- [38] Z. Abid, A. Alma'aitah, M. Barua, and W. Wang, "Efficient CMOL gate designs for cryptography applications," *IEEE Trans. Nanotechnol.*, vol. 8, no. 3, pp. 315–321, May 2009.
- [39] Y. Wang, L. Ni, C.-H. Chang, and H. Yu, "DW-AES: A domain-wall nanowire-based AES for high throughput and energy-efficient data encryption in non-volatile memory," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 11, pp. 2426–2440, Nov. 2016.



VAHID JAMSHIDI received the M.Sc. and Ph.D. degrees in computer engineering from the Iran University of Science and Technology, Tehran, Iran, in 2004 and 2017, respectively. He has been with the Department of Computer Engineering, Shahid Bahonar University of Kerman (SBUK), Kerman, Iran, since 2006, where he is currently an Assistant Professor. His current research interests include reliable issues in very large scale integration (VLSI) circuits and hybrid technologies, low power circuits and systems, dependable embedded systems, fault-tolerant computer architectures, and reliability modeling and analysis techniques.



AHMAD PATOGHY received the Ph.D. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2011. From 2011 to 2017, he was an Assistant Professor with the Iran University of Science and Technology, Tehran. From 2018 to 2021, he was with the University of Central Arkansas, AR, USA. Since August 2021, he has been with the Department of Computer Systems Technology, North Carolina Agricultural and Technical State University, where he is currently leading the Intelligent and Embedded Systems Laboratory. He has published more than 80 journal articles and conferences papers, and served as a reviewer and a PC Member for a wide variety of IEEE, ACM, Elsevier, and Springer journals and conferences. He has also served as a Panelist for the National Science Foundation, for reviewing grant proposals. His research interests include security and reliability of multi-processor systems-on-chips (MPSoC), hardware security in embedded and cyber-physical systems, and hardware design and acceleration for deep/spiking neural networks.



MAHDI FAZELI received the M.Sc. and Ph.D. degrees in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2005 and 2011, respectively. He is currently an Associate Professor with the School of Information Technology, Halmstad University, Sweden. His research interests include hardware security and trust, reliable VLSI circuits and systems, energy efficient computing, and dependable embedded systems.

...