IMCS2: Novel Device-to-Architecture Co-Design for Low-Power In-Memory Computing Platform Using Coterminous Spin Switch

Farhana Parveen, Shaahin Angizi, Zhezhi He, and Deliang Fan

Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816 USA

***Abstract*—Spin switch (SS) is a promising spintronic device which exhibits compactness, low power, non-volatility, input–output isolation leveraging giant spin Hall effect, spin transfer torque, and dipolar coupling. In this paper, we propose a novel device-to-architecture co-design for an in-memory computing platform using coterminous SS (IMCS2), which could simultaneously work as non-volatile memory and reconfigurable in-memory logic (AND/NAND, OR/NOR, and XOR/XNOR) without add-on logic circuits to memory chip. The computed logic output could be simply read out like a normal magnetic random access memory bit cell using the shared memory peripheral circuits. Such intrinsic in-memory logic could be used to process data within memory to greatly reduce power-hungry and long distance data communication in the conventional von Neumann computing system. The IMCS2-based in-memory bulk bitwise Boolean vector operation shows ∼9× energy saving and ∼3× speedup compared with that of DRAM-based in-memory computing platform. We further employ in-memory multiplication to evaluate the performance of the proposed in-memory computing platform for vector–vector multiplication with different vector sizes.**

***Index Terms*— Giant spin Hall effect (GSHE), in-memory computing, memory architecture, reconfigurable logic, spin switch (SS).**

I. Introduction

BIG DATA that inundates the modern information and computing sectors invigorates the emerging needs of re-designing the traditional computing platforms to support memory-oriented processing at exascale (10^{18} B/s) [1]. The separation of memory and computing units in von Neumann architecture demands long memory access latency, significant congestion at I/Os, limited memory bandwidth, and power-hungry data transfer. These give birth to the exasperating challenge entitled "memory wall" or "power wall" [2], especially for big data-driven applications [3]. To address these issues, various near-memory or in-memory computing platforms [4]–[20] are being investigated nowadays, which could pre-process data before being sent to the main processor, thus reducing data transfer power consumption and increasing the speed and bandwidth of the whole system. However, combining memory and logic is a challenging task since it needs simultaneous optimization of the "performance" for efficient logic operation and "density" for efficient memory operation.

Several dynamic random access memory (DRAM)-based memories have been proposed [15], [17], [21]–[23] to integrate logic in memory for bandwidth intensive computing within memory. However, there are several drawbacks of DRAM-based designs. First, DRAM is volatile, which requires repeated refresh operations to hold the data for a long time and only offers destructive read/compute operations. Second, these DRAM-based in-memory logic designs cannot perform all the Boolean logic functions within memory, for example, [15] can implement only AND and OR operations. Again, it requires four clock cycles to complete one compute operation, thus degrading the performance in terms of latency. Furthermore, for embedding logic functionality within memory chip, the complexity and cost of DRAM chip increase significantly.

Recently, emerging non-volatile memories are being exploited to implement energy-efficient logic computation within memory. Magnetic random access memory (MRAM) has been investigated as a promising candidate for designing in-memory processing platforms [14], [24]–[32] owing to its unique features, such as non-volatility, zero standby leakage, high write/read speed, compatibility with CMOS fabrication process, and scalability. However, the required high write current leads to high energy consumption (10× more than static random access memories (SRAM) in single cell) [27]. Again, MRAMs are prone to several failures [33]–[36], due to the shared read–write current path. Besides, in different MRAM-based in-memory logic designs [29], [30], [37], the sensing reliability degrades significantly since parallel magnetic tunnel junction (MTJ) resistance is considered. Pinatubo [16] has shown a general platform for in-memory computing with non-volatile memories. However, it also falls in the vulnerability to reliability concerns due to considering parallel compute current paths.

In this paper, we propose a novel device structure exploiting giant spin Hall effect (GSHE)-based spin switch (SS) [38], [39] which can be utilized to build a non-volatile memory cell. Furthermore, we have proposed the array architecture using our proposed device to implement in-memory computing platform, IMCS2. Our proposed design is capable of operating in two modes—memory mode and computing mode. In the memory mode, the proposed array could work

Manuscript received August 24, 2017; revised November 19, 2017 and February 16, 2018; accepted March 9, 2018. Corresponding author: D. Fan (e-mail: dfan@ucf.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMAG.2018.2819959

as a typical non-volatile memory. In the computing mode, two memory cells could be leveraged to realize bitwise AND/NAND, OR/NOR, and XOR/XNOR Boolean logic functions within memory without any add-on circuitry as in typical logic-in-memory platforms, thus keeping the memory density unaffected.

Our proposed design exploits GSHE-driven SS as the memory element [38], [39], which enables the complete isolation of read and write current paths, as opposed to the conventional spin transfer torque (STT)-MRAM [14], [24], [29], [30]-based designs. This improves the read reliability by reducing the vulnerability to unintentional bit flipping while reading. Again, IMCS2 is designed with two transistors per memory cell. Hence, it can successfully implement both memory and computing operations without sacrificing memory density compared with other spin-orbit torque (SOT)-MRAM designs [27], [40]–[42]. Moreover, our design can implement every Boolean logic function within one clock cycle, whereas other designs [30], [41] can implement only AND/OR logic operations. Pinatubo [16] implemented XOR function; however, they used capacitors and it requires multiple clock cycles to compute XOR. On the other hand, Li *et al.* [16] and Jain *et al.* [29] incorporated the parallel resistance between two or more adjacent memory cells, to implement in-memory logic computing. However, it is difficult to differentiate the parallel resistances for two logic states, unless the memory cells have very high tunneling magnetoresistance (TMR). Our design omits this difficulty by incorporating series resistance between two memory cells for logic operations. Above all, our design can implement logic operation between any two cells in any odd row and any even row within the memory array. This removes the restrictions on operands to be in the adjacent memory cells [29], [41]. In a nutshell, the major contributions of our proposed IMCS2 are as follows.

1) Novel Coterminous Spin-Switch structure which could be used as a memory cell exploiting the GSHE-driven MTJ concept with the complete isolation of read and write current paths.
2) Novel array circuit to incorporate in-memory computing platform activating two modes of operations—memory mode and computing mode.
3) Two transistors per memory cell, activating both memory and logic operations without area overhead or additional add-on circuitry to implement logic functionality.
4) Every two input Boolean logic function—AND/NAND, OR/NOR, and XOR/XNOR could be implemented within one sensing clock cycle, without prolonged latency.
5) Logic operation could be performed between any two cells in any odd row and any even row, thus introducing flexibility on the "operand locality" within the memory array.
6) Density optimized memory and performance optimized logic are ensured simultaneously for efficient memory and logic operations.

Device-circuit-architecture-level simulations were carried out to validate the functionality and performance of memory and logic operations in the proposed in-memory computing platform. Furthermore, bulk bitwise in-memory vector AND/OR operation and in-memory multiplier were implemented to elucidate the in-memory computing performance of our proposed IMCS2. Device-to-architecture co-simulation results show that bitwise vector operation can offer $\sim 9\times$ energy saving and $\sim 3\times$ speedup compared with that using DRAM-based in-memory computing platform.

The rest of this paper is organized as follows. Section II describes typical GSHE-based SS device structure. The proposed device structure along with its parameter specifications for modeling is presented in Section III. The proposed in-memory computing architecture is introduced in Section IV. Device-, circuit-, and system-level simulation results are analyzed in Sections V and VI. Reliability of the proposed design is investigated, and the analysis is shown in Section VII. Section VIII presents significant applications where our proposed in-memory computing architecture could be efficiently leveraged. Finally, Section IX concludes this paper.

II. Giant Spin Hall Effect-Driven Spin Switch

SS [38], [39] is a composite spintronic device structure exploiting three major spin transport phenomena, e.g., GSHE, STT, and dipolar coupling. Fig. 1(a) shows an SS with single MTJ that was first proposed in 2012 [38]. This SS consists of a spin Hall metal (SHM), two free ferromagnetic layers (FLs) coupled by a coupling insulator, an MTJ formed by a pinned ferromagnetic layer (PL), and a FL sandwitching a tunnel barrier. Fig. 1(b) shows the terminals for directing write or read currents through the device. Terminals Vin+ and Vin- are used for flowing current through the SHM, whereas VDD and Vout are used for flowing current through the MTJ.

Below the basic concepts underlying behind the SS along with its modeling approach [43] is described.

A. Basic Concepts

1) Magnetic Tunnel Junction: An MTJ is formed with a PL and a FL sandwiching a tunneling barrier. MTJ works as a storage element for the memory cell. The resistance states—high and low—are used to encode data "1" and "0," respectively. Due to the TMR effect [44], the resistance of MTJ is high, R_{AP} (/low, R_P) when the magnetization of two ferromagnetic (FM) layers are anti-parallel, AP (/parallel, P), as shown in Fig. 1(c). The FL magnetization could be manipulated by applying an external magnetic field, or through current-induced STT [26], or through SOT produced by GSHE [42]. This resistance could be measured by injecting sense current through MTJ, as shown in Fig. 1(c). In this paper, the MTJ model has used in-plane magnetic anisotropy (IMA), whose magnetic domain is along with the FM layer's transverse section.

2) Giant Spin Hall Effect: In comparison with the conventional method of programming the FL's magnetization of MTJ by external magnetic field or by current-induced STT, it is more efficient to utilize spin current generated by GSHE [37]. Here, the flow of charge current ($\pm Y$) through the SHM will cause the accumulation of oppositely directed electron spin on both surfaces of SHM due to the spin Hall effect [45]. Thus, a spin current in $\pm Z$ is generated and further SOT is produced on the adjacent free magnetic layer, causing switch of magnetization [42]. For example, as shown in Fig. 1(d),

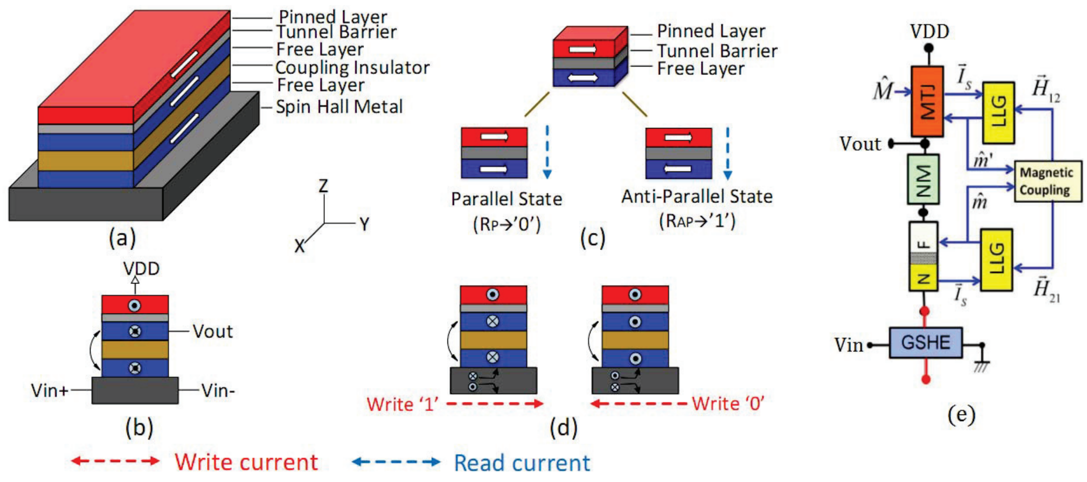


Fig. 1. (a) SS (single MTJ model). (b) SS with terminals. (c) MTJ structure. (d) Spin Hall effect. (e) Modular representation of the SS [43].

spin current produced in the $+Z$-direction by flowing charge current through the SHM in the $+Y$ (/$-Y$)-direction will align the magnetization of the FL adjacent to the SHM in the $-X$ (/$+X$)-direction, which is anti-parallel (/parallel) to that of the PL.

3) *Magnetic Coupling:* Magnetic coupling is represented by the magnetic interaction between a pair of magnets caused by both exchange- and dipolar-type interactions [43]. As in Fig. 1(a), a coupling insulator performs the task of magnetic coupling between the FM layers on the either side of it. Due to this coupling layer, the magnetization of the FL adjacent to the SHM is copied to the FL adjacent to the tunneling barrier [46]. As shown in Fig. 1(d), spin current produced in the $+Z$-direction by flowing charge current through the SHM in the $+Y$ (/$-Y$)-direction will align the magnetization of the FL adjacent to the SHM in the $-X$ (/$+X$)-direction, which will be copied to FL adjacent to the tunneling barrier. Hence, this will write data "1" (/"0") encoded as R_{AP} (/R_P), respectively, in the MTJ.

B. Modeling Approach

Datta *et al.* [38], Ganguly *et al.* [39], Camsari *et al.* [43], and Penumatcha *et al.* [47] established a set of elemental modules that represent diverse materials and phenomena through careful benchmarking against available theory and experiment. Such elemental modules can be integrated seamlessly to model composite devices involving both spintronic and nanomagnetic phenomena. In the developed spintronic modular approach library, the modules mainly consist of transport blocks based on the physics of transport and magnetic blocks based on the physics of magnetism. In the modeling of our conterminous SS devices, such spintronic modular approach is strictly followed in our work, as shown in Fig. 1(e). Here, our proposed coterminous SS device model includes spin current (I_s) generation through GSHE, magnetization dynamics (m) modeled through Landau–Lifshitz–Gilbert (LLG) equation, non-magnet (NM) module modeled as a reciprocal π-network containing a series and shunt conductance matrix, FM module modeled as a reciprocal π-network containing a series and shunt conductance matrix obtaining from spin-diffusion equation, FM–NM interface module consisting of the tunneling effect and ohmic contact, and magnetic coupling module to represent magnetic interaction between a pair of magnets using both exchange and dipolar interactions. Below the modules used in our design are briefly introduced.

1) *LLG Solver Module:* This module solves the LLG equation to obtain the time-dependent magnetization dynamics in the presence of magnetic fields and spin currents [43]. LLG equation for magnetization switching dynamics [48] is given by

$$(1+\alpha^2)\frac{\partial\vec{m}}{\partial t} = -|\gamma|(\vec{m}\times\vec{H}_{\text{eff}}) - \alpha|\gamma|\vec{m}\times(\vec{m}\times\vec{H}_{\text{eff}}) - \vec{m}\times\vec{m}\times\frac{I_s}{qN_{sx}} - \alpha\vec{m}\times\frac{I_s}{qN_{sx}}. \quad (1)$$

Here, α is Gilbert damping factor, γ is gyromagnetic ratio, H_{eff} is the effective field including dipolar coupling field, demagnetization field, thermal noise field, and anisotropy field. $N_s = M_s V/\mu_B$ is the number of spins, μ_B is Bohr magneton, and M_s and V are saturation magnetization and volume of FM, respectively. I_s is the spin polarized current.

2) *Giant Spin Hall Effect (GSHE) Module:* The GSHE module is used to model the functionality of spin current generation using charge current flow through the SHM. The model is derived from modified spin-diffusion equation using spin Hall angle as input and inverse and direct spin Hall effects as the results

$$G_{\text{sh}} = \sigma\frac{LW}{\lambda}\tanh\left(\frac{t}{2\lambda}\right) \quad (2)$$

$$G_{\text{se}} = \sigma\frac{LW}{\lambda}\text{csch}\left(\frac{t}{\lambda}\right) \quad (3)$$

$$I_{\text{spin}} = \beta G_0 \Delta V_{\text{charge}} \quad (4)$$

$$I_{\text{charge}} = \beta G_0 \Delta V_{\text{spin}} \quad (5)$$

$$G_0 = \sigma\frac{tW}{L}, \quad \beta = \theta_{\text{SH}}\frac{L}{t}s \quad (6)$$

where G_{sh} and G_{se} are the shunt and series conductances of the SHM model [43], σ is the conductivity of the SHM, L, W, and t are the length, width, and thickness of the SHM, θ_{SH} is

TABLE I
DEVICE PARAMETERS USED FOR MODELING

Module	Parameters	Value
MTJ	Conductance	0.1×10^{-3} S
	Polarization	0.7
	In-Plane Coefficient	1
	Out of Plane Coefficient	0
GSHE	Spin Flip Length	2×10^{-9} m
	Resistivity	$200 \times 10^{-8}\ \Omega m$
	Spin Hall Angle	0.3
FM-NM	Conductance	5×10^{15} S
	Polarization	0.7
	In-Plane Coefficient	1
	Out of Plane Coefficient	0
LLG	Damping Coefficient	0.01
	Saturation Magnetization	$800\ emu \times cc$
	Anisotropic Field Strength	$130\ Oe$
Dipolar Coupling	Saturation Magnetization	$800emu \times cc$
	Dipolar Coefficients	0.0256, -0.0128, -0.0128

the spin Hall angle, λ is the spin flip length, and ΔV_{charge} and ΔV_{spin} are the voltage difference across the SHM in the direction of charge current and spin current flow, respectively. Please note that G_0 is the conductance of the SHM across the direction of spin current flow and β is the spin Hall angle including the geometric factor (L/t).

3) Magnetic Tunnel Junction Module: MTJ is composed of a PL-tunneling barrier-FL stack, and hence has been modeled using two FM–NM interfaces in series [43], where the conductance of the MTJ is the product of two FM–NM interfaces.

4) FM–NM Interface Module: FM–NM module represents the interface between the FM (FL) and the NM (SHM), for modeling the spin current through the FM and NM layers, which uses the coherent transport theory. It is also modeled using series and shunt conductances G_{sh} and G_{se} of the FM–NM interface [43].

5) Magnetic Coupling Module: This module models the coupling mechanism between the two FLs. This module takes magnetizations of the two magnets as inputs and provides the magnetic fields that they exert on each other as the output [43].

Table I shows the device parameters that we have used for modeling the SS.

III. PROPOSED DEVICE STRUCTURE AND MODELING: COTERMINOUS SPIN SWITCH

In this paper, we propose a coterminous device structure consisting of two back-to-back SSs having a common PL. The proposed device structure is shown in Fig. 2(a).

In the proposed device structure, SHM 1 (2), FLs 1a, 1b (2a, 2b), coupling layer 1 (2), tunnel barrier 1 (2), and the PL form the first (second) SS. Note that the first and second SSs share their common PL. Thus, these two back-to-back SSs could be interpreted as forming a coterminous device structure. Due to the presence of the coupling layer, FL 1b (FL 2b) has the same magnetization as FL 1a (FL 2a). Here, FL 1b (FL 2b), tunnel barrier 1 (2) and the PL altogether form MTJ 1 (MTJ 2). The dimensions and materials that we considered for each of the abovementioned layers are provided in Table II.

Fig. 2(b) shows the terminals of this device. Where Vin1+ and Vin1− (Vin2+ and Vin2−) are the terminals for flowing

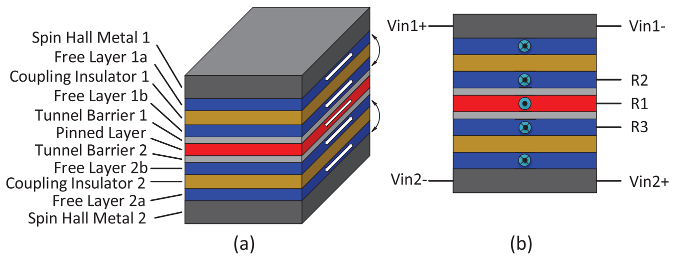


Fig. 2. (a) Proposed coterminous SS device structure. (b) Proposed device with terminals.

TABLE II
DIMENSIONS AND MATERIALS FOR THE PROPOSED DEVICE

Layer	Material	Dimension (L, W, t)
Spin Hall Metal	W [27]	(40nm, 80nm, 2.8nm)
Free Layer	CoFeB [27], [49]	(40nm, 80nm, 1.5nm)
Coupling Layer	FeCo-Oxide [46]	(40nm, 80nm, 2nm)
Tunnel Barrier	MgO [27], [49]	(40nm, 80nm, 1.3nm)
Pinned Layer	CoFeB [27], [49]	(40nm, 80nm, 1.5nm)

current through the SHM 1 (SHM 2) for programming FL 1a (FL 2a). Terminals R1 and R2 (R3) can be used to flow sense current through the MTJ 1 (MTJ 2).

Data can be written to FL 1a (2a) by flowing current through SHM 1 (SHM 2). As described in Section II, by flowing current through SHM 2 from Vin2+ to Vin2− (from Vin2− to Vin2+), the magnetization of FL 2a can be set into the parallel (anti-parallel) direction of the magnetization of PL. The magnetization of FL 2b will also be parallel (anti-parallel) to that of PL, since the coupling layer 2 couples the magnetizations of FL 2a and FL 2b. This will program MTJ2 resistance to be low, R_P (high, R_{AP}), which can be considered as data "0" (data "1"). Note that FL 1a is connected to the bottom surface of the SHM 1, which requires the reversed direction of current flow through SHM 1 for setting the magnetization of FL 1a in the same direction as that of FL 2a, as described above. It is why Vin1+ and Vin1− terminals are connected in the reversed direction as that of Vin2+ and Vin2−, respectively. With such connection, by flowing current through SHM 1 from Vin1+ to Vin1− (from Vin1− to Vin1+), the magnetization of FL 1a can be set into the parallel (anti-parallel) direction of the magnetization of PL. The magnetization of FL 1b will also be parallel (anti-parallel) to that of PL, since the coupling layer 1 couples the magnetizations of FL 1a and FL 1b. It will program MTJ1 resistance to be low, R_P (high, R_{AP}), which can be considered as data "0" (data "1"). MTJ 1 (MTJ 2) resistance can be read by flowing small sense current from R2 (R3) to R1.

A. Device Modeling and Validation

The modeling and simulation of the proposed device are based on the framework of modular approach [43], where device materials and physical phenomena are modeled by experimentally benchmarked modules [39]. The modules that are used are: GSHE module for modeling the functionality of the SHM, FM–NM interface module for modeling the behavior of the SHM and FL interfaces, LLG module for modeling the FL magnetization switching phenomenon, dipolar coupling

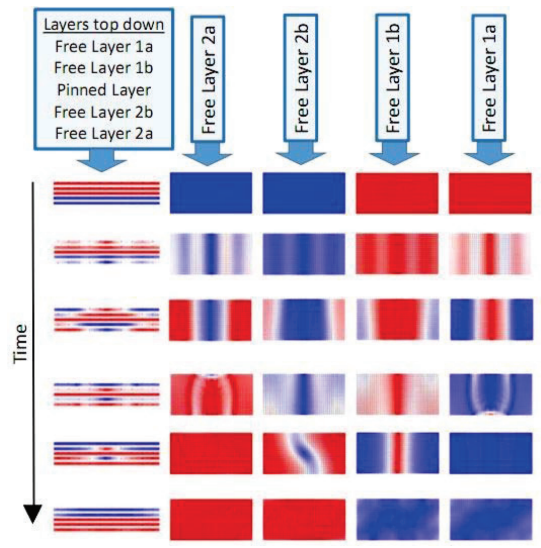


Fig. 3. Micromagnetic simulation results for the FLs.

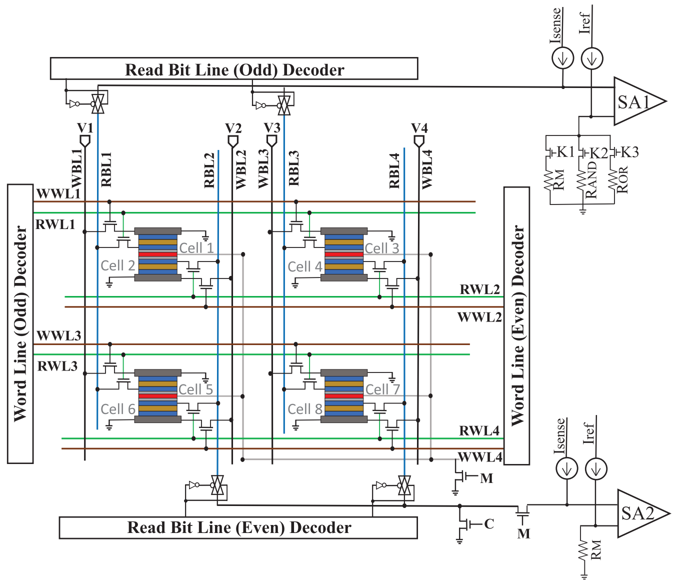


Fig. 4. Array architecture of the proposed in-memory computing architecture using coterminous SS.

module for modeling the magnetic field/magnetization coupling between the two FLs, and MTJ module for modeling the composite structure of FL and PL sandwiching a tunneling barrier. The baseline parameters [39] used for each module of the proposed device are listed in Table I.

We conducted micromagnetic simulation based on widely used object-oriented micromagnetic framework (OOMMF) built by NIST [50]. The same device parameters are used in OOMMF simulator, and the simulation results shown in Fig. 3 matches with the spintronic modular approach we used. In such micromagnetic simulation, all of the above discussed physical phenomena are included and 2 nm $\times$ 2 nm $\times$ 0.5 nm spin cube is used in simulation. The SHM layer is not shown for simplicity. It can be seen that when charge current is applied to both SHMs, FL 2a and 1a switched their magnetization due to spin current generated through GSHE, while layer 1b and 2b also switched their magnetization due to magnetic coupling discussed above. Please note that the impact of process variation is investigated in [51].

IV. Proposed In-Memory Computing Architecture

In this section, we introduce our proposed memory architecture using the proposed coterminous SS, which is capable of performing both memory operations and in-memory Boolean logic computing operations.

A. Array Architecture With Proposed Coterminous Spin Switch

Fig. 4 shows the architecture of 4 $\times$ 2 memory array. Here, each coterminous SS constitutes two memory cells. Each cell is associated with the write word line (WWL), read word line (RWL), write bitline (WBL), and read bitline (RBL). All word lines and RBLs are controlled by four peripheral decoders. The odd (even) word lines can be activated by the left (right) word line decoder. WBLs are connected to voltage drivers. Again, the odd (even) RBLs can be connected to SA1 (SA2) using the top (bottom) RBL decoder. Current-mode sense amplifiers (SA) [52] are connected to the RBLs for sensing the total MTJ resistance in the selected current path during read or compute operation.

TABLE III
Control Signals for Memory and Computing Mode Operations

	Write MTJ1	Write MTJ2	Read MTJ1	Read MTJ2	Logic between MTJ1 and MTJ2	
					AND/OR	XOR
M	1	1	1	1	0	1
C	0	0	0	0	1	0
WWL1	1	0	0	0	0	0
WWL2	0	1	0	0	0	0
WWL3	0	0	0	0	0	0
WWL4	0	0	0	0	0	0
RWL1	0	0	1	0	1	1
RWL2	0	0	0	1	1	1
RWL3	0	0	0	0	0	0
RWL4	0	0	0	0	0	0
RBL1	F*	F	SA1	F	SA1	SA1
RBL2	F	F	F	SA2	0	SA2
RBL3	F	F	F	F	F	F
RBL4	F	F	F	F	F	F
WBL1	V+/V-**	V+/V-	0	0	0	0
WBL2	0	0	0	0	0	0
WBL3	0	0	0	0	0	0
WBL4	0	0	0	0	0	0

*F = Floating, **V+ = 0.18V, V- = -0.18V

As shown in Fig. 4, each coterminous SS is connected with four transistors. Since each coterminous SS constitutes two memory cells, it can be said that each memory cell contains two transistors. One transistor is connected to the SHM to control the flow of write current through the SHM, whereas the other transistor is connected to the FL adjacent to the PL to control the flow of read current through the MTJ formed by the FL and PL.

All PLs in the entire array are shorted together and connected to the ground through a control transistor. Note that this transistor is not associated with each cell, rather is a part of the peripheral circuitry of the entire array. By turning this transistor ON, all PLs are connected to the ground. On the contrary, by turning this OFF, all PLs are floating while still being connected altogether. Such arrangement of PLs is

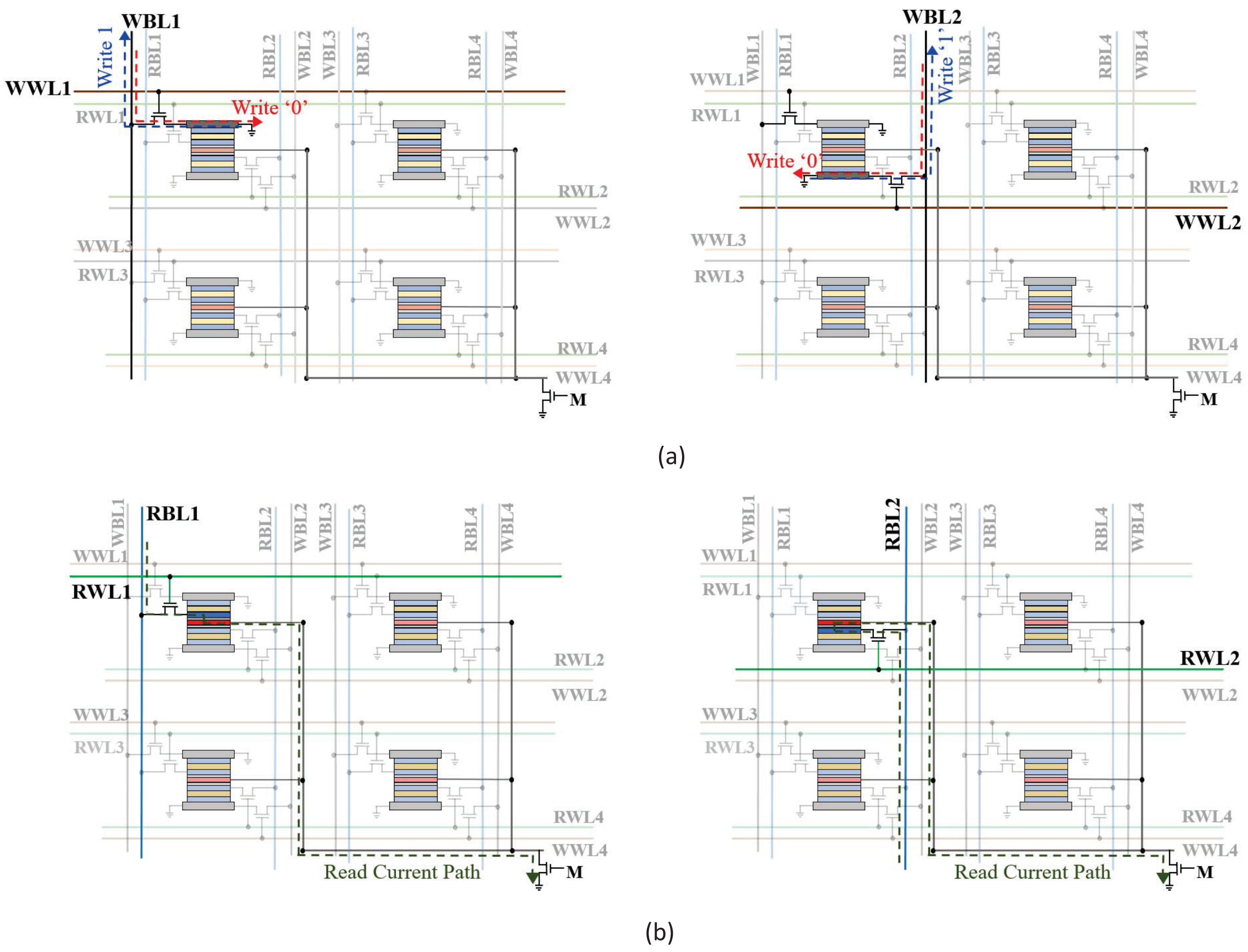


Fig. 5. (a) Write current paths for writing into cell1 (left) and cell2 (right). (b) Read current paths for reading MTJ1 (left) and MTJ2 (right).

made to activate the dual-mode operation—memory mode and computing mode, in the same array without any add-on circuit. Here, two control signals—M and C are used to switch the functionality between memory mode and computing mode. The control signals for memory and control mode operations are shown in Table III. The operations of the two modes are described in detail in Sections IV-B and IV-C. Also, memory write, read, and compute operations for an entire word are explained in detail.

B. Modes of Operation

1) Memory Mode: In memory mode, control signal M is set at high voltage (VDD) and C is set at low voltage (GND). So, all PLs are connected to the ground. Now, data can be written or read from memory cells by flowing write or read current through the appropriate path. The activation of the appropriate current path is implemented using control signals—WWL1-4, RWL1-4, WBL1-4, and RBL1-4. Here, "memory write" and "memory read" operations are explained in detail.

a) Memory write: To write a bit in any cell of the memory array shown in Fig. 4, write current should be flowed through the corresponding SHM. For example, to write into the cell of row-1 and column-1 (cell 1 with MTJ1), write current should be flowed through the SHM of that cell. In order to activate this write current path [as shown in Fig. 5(a)], WWL1 is activated using write line (odd) decoder. All other word lines are kept deactivated. Now, to write data 1 (0), the voltage driver V1 connected with WBL1 is set to the positive (negative) write voltage. (All other write voltage drivers are kept at zero voltage.) This allows charge current to flow from V1 to ground (ground to V1) through SHM. As described in Figs. 1(d) and 2(b), this will set MTJ1 resistance to be high, R_{AP} (low, R_P) which is encoded as data "1" ("0").

Similar to write into cell 2 with MTJ2, WWL2 is activated using write line (even) decoder and voltage driver; V2 connected with WBL2 is set to the positive (negative) write voltage. This activates write current path through the SHM of that cell.

b) Memory read: To read the data stored in a memory cell, read current should be flowed through the corresponding MTJ. For example, to read data stored in cell 1, read current should be flowed through MTJ1. To activate this read current path [as shown in Fig. 5(b)], RWL1 is activated using word line (odd) decoder and RBL1 is connected to SA1 by activating RBL (odd) decoder. All other word lines and bitlines are kept deactivated. Reference MTJ, R_M of SA1 is to be selected by setting key values K1-K3 = 100. Hence, a read current is flowed from R_M through MTJ1 to the ground (since $M = 1$), generating a sense voltage at the input of SA1. SA1 then compares MTJ1 resistance with reference MTJ, R_M whose resistance is set in between R_P and R_{AP}. So, if memory resistance, MTJ1 is higher (lower) than R_M, i.e., R_{AP} (R_P), the output of SA1 gives high (low) voltage—which indicates data "1" ("0") is read from cell 1.

Similarly for reading data stored in cell 2, RWL2 should be activated and RBL2 should be connected to SA2. Read current is flowed from R_M through MTJ2 to the ground [as shown in Fig. 5(b)].

2) Computing Mode: In the computing mode, logic operations between two cells can be computed by activating the appropriate compute current path and selecting the appropriate SA. In order to compute AND/OR logic between the data stored in two memory cells belonging to the same SS, control signal C is set at high voltage (VDD) and M is set at low voltage (GND). So, all PLs are now floating while being connected altogether. Here, compute current should be flowed through the two corresponding MTJs in series. Thus, summation of the MTJ resistances of such two cells is connected to the SA. Again, in order to compute XOR logic between two cells in an SS, both cells are needed to read separately and simultaneously by setting $M = 1$ and $C = 0$. Here, both read current paths and both SAs are needed to be activated. Now, by adding a simple peripheral circuitry at the output of SA2, logic XOR result can be obtained.

A noteworthy applicability of this proposed architecture is that logic operations can be performed between any cells in any odd row with any cells in any even row, more precisely—between a cell on any of the upper SS and a cell on any of the lower SS. To paraphrase, AND/OR/XOR logic can be computed between any cell#odd with any cell#even, e.g., between cell 1 (or cell 3 or cell 5 or cell 7) and cell 2 (or cell 4 or cell 6 or cell 8). Below implementation of all logic functions is described separately.

a) AND/NAND: To compute logic AND/NAND function, array should be activated in the computing mode (C = 1 and M = 0), so the PLs are **not** connected to the ground. For example, to compute AND/NAND between data stored in MTJ1 and MTJ2, series resistance of MTJ1 and MTJ2 is to be measured. To do so, compute current (as shown in Fig. 6) is flowed through both MTJ1 and MTJ2 by activating RWL1 and RWL2 simultaneously using word line (odd) and (even) decoders, respectively. RBL1 is connected to SA1 by activating RBL (odd), and RBL2 is connected to the ground (since $C = 1$) by activating RBL (even). All other word lines and bitlines are kept deactivated. Reference MTJ, R_{AND} of SA1 is selected by setting key values K1-K3 = 010. Hence, compute current is flowed from R_{AND} through MTJ1 and MTJ2 to the ground, generating a sense voltage at the input of SA1. SA1 then compares the resistance $R_{\text{TOT}} = R_{\text{MTJ1}} + R_{\text{MTJ2}}$ with reference MTJ, R_{AND} whose resistance is set in between $2R_{\text{AP}}$ and $R_P + R_{\text{AP}}$. So, if R_{TOT} is higher (lower) than R_{AND}, then the output of SA1 gives high (low) voltage—which indicates logic "1" ("0") is the result of AND operation. Here, NAND operation result is obtained from the differential output of SA1 [52]. Table IV explains the computation of AND/NAND operation in detail.

TABLE IV

IMPLEMENTATION OF COMPUTING MODE (AND/OR)

Data stored in the two MTJs	0 0	0 1	1 0	1 1
Summation of MTJ resistances, R_{TOT}	$R_P + R_P$	$R_P + R_{AP}$	$R_{AP} + R_P$	$R_{AP} + R_{AP}$
Compare R_{TOT} to R^1_{AND}	$<R_{AND}$	$<R_{AND}$	$<R_{AND}$	$>R_{AND}$
SA1 output (AND)	0	0	0	1
Compare R_{TOT} to R^2_{OR}	$<R_{OR}$	$>R_{OR}$	$>R_{OR}$	$>R_{OR}$
SA1 output (OR)	0	1	1	1

[1] R_{AND} = Between $2R_{AP}$ and $R_P + R_{AP}$
[2] R_{OR} = Between $2R_P$ and $R_P + R_{AP}$

In general, to compute logic AND/NAND between data stored in MTJ1 (or MTJ 3 or MTJ5 or MTJ7) and MTJ2 (or MTJ 4 or MTJ 6 or MTJ 8), RWL1 (or RWL1 or RWL3 or RWL3) and RWL2 (or RWL2 or RWL4 or RWL4) should be activated, while RBL1 (or RBL3 or RBL1 or RBL3) should be connected to SA1 and RBL2 (or RBL4 or RBL2 or RBL4) should be connected to the ground (as $C = 1$). For example, compute current path for AND/NAND operation between MTJ 1 and MTJ 6 is shown in Fig. 6(b).

b) OR/NOR: Logic OR/NOR can be implemented in a similar way as AND/NAND. The only exception is that reference MTJ, R_{OR}, of SA1 is selected by setting key values K1-K3 = 010 for this case. Thus, by flowing compute current from R_{OR} through MTJ1 and MTJ2 to the ground ($C = 1$), a sense voltage is generated at the input of SA1. SA1 then compares the resistance $R_{\text{TOT}} = R_{\text{MTJ1}} + R_{\text{MTJ2}}$ with reference MTJ, R_{OR} whose resistance is set in between $2R_P$ and $R_P + R_{\text{AP}}$. Hence, the result of OR (NOR) function could be obtained from SA1 output (differential output). Table IV explains the computation of OR/NOR operation in detail.

c) XOR/XNOR: To compute logic XOR/XNOR function, the array should be activated in the memory mode ($M = 1$ and $C = 0$), so PLs are connected to the ground. To compute XOR/XNOR between data stored in MTJ1 and MTJ2, both data stored in MTJ1 and MTJ2 are needed to be read separately but simultaneously. Both RWL1 and RWL2 are activated at the same time using word line (odd) and (even) decoders, respectively. Both RBL1 and RBL2 are connected to SA1 and SA2, respectively, using RBL (odd) and (even) decoders, respectively. Hence, two separate compute currents will flow through two MTJs, one from SA1 through MTJ1 to the ground and the other from SA2 though MTJ2 to the ground at the same time. Thus, SA1 and SA2 will generate outputs at the same time. Note that these two outputs are simply the results of memory read operations for MTJ1 and MTJ2, and hence reference MTJ for SA1 is selected to be R_M by setting K1-K3= 100. Now, a simple peripheral circuitry (described in this section) added to the end of SA2 can generate the result of XOR/XNOR logic between MTJ1 and MTJ2.

The XOR/XNOR operation between any odd cell and any even cell can be performed in a similar fashion as that mentioned for AND/NAND operation. The only difference is that even RBLs are connected to SA2 rather than to the ground.

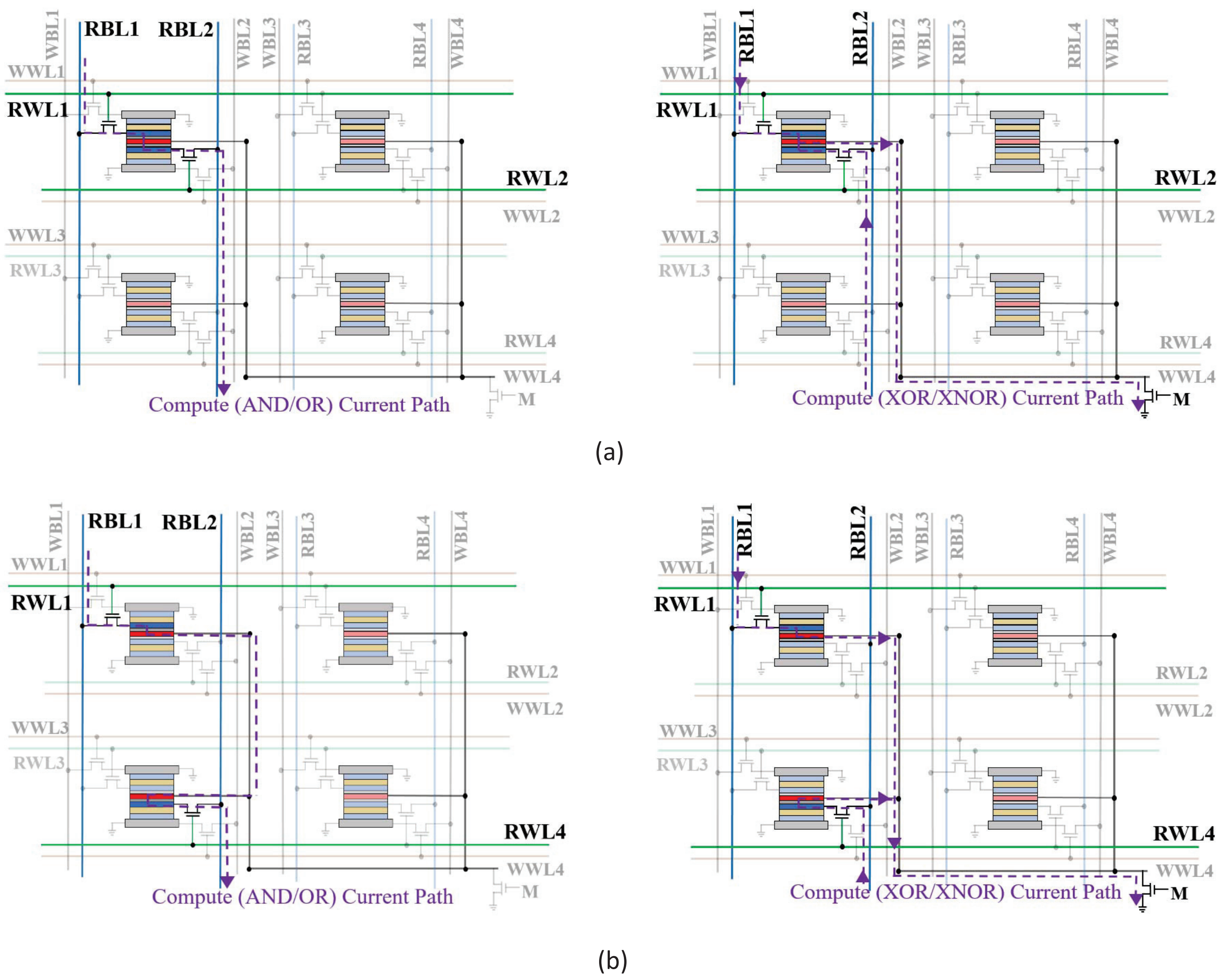


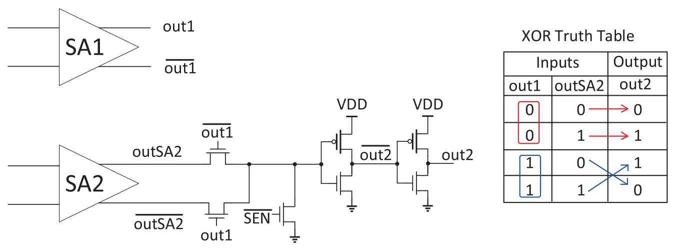
Fig. 6. (a) Compute current path for AND/OR operations between MTJ1 and MTJ2 (left) and compute current path for XOR operations between MTJ1 and MTJ2 (right). (b) Compute current path for AND/OR operations between MTJ1 and MTJ6 (left) and compute current path for XOR operations between MTJ1 and MTJ6 (right).

For example, current path for computing XOR/XNOR logic between MTJ1 and MTJ6 is shown in Fig. 6(b).

C. Peripheral Circuit Extension of Sense Amplifier for Implementing XOR/XNOR Logic Functions

If activated at the same time, SA1 and SA2 give the data stored in the selected even and odd cells, respectively, at their outputs simultaneously. Fig. 7 shows the peripheral circuitry added to the output of SA2 to compute logic XOR/XNOR function. It can be seen from the truth table of XOR gate that logic XOR output (out2) is the same as (/inverse of) SA2 output (outSA2), when SA1 output (out1) is 0 (/1). Hence, the peripheral circuitry is designed to propagate SA2 output—outSA2 when out1 is 0, and inverse of outSA2 when out1 is 1. Thus, this modification can implement XOR logic at out2 terminal and XNOR logic at $\overline{\text{out2}}$ terminal.

Note that our circuit-level simulation shows that it takes ~0.3 ns for SA1 or SA2 to generate outputs of read or



Inputs		Output
out1	outSA2	out2
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 7. Peripheral extension of SA2 for implementation of XOR/XNOR logic functions.

compute operations. Hence, XOR/XNOR result could be obtained within 1 clock cycle that is provided for sensing.

D. Complete Word Operation

For complete word operation in a more realistic array size other than the simple 4 × 2 array, our proposed architecture can perform memory write and memory read operations in parallel, while compute operations should be carried out bit by bit or serially.

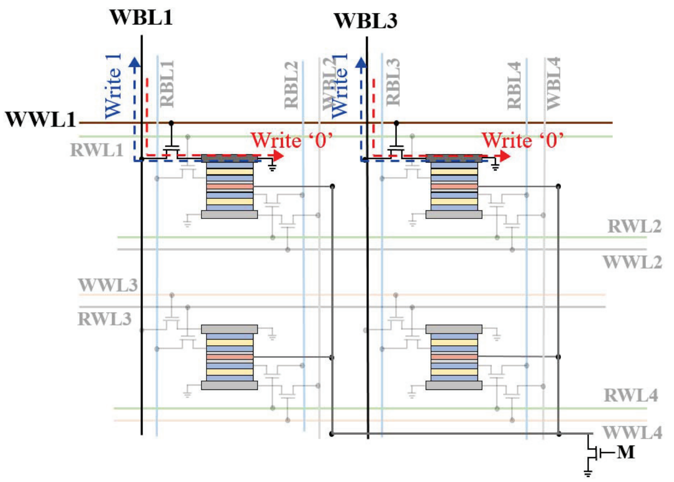


Fig. 8. Memory write operation for a complete word.

To clarify, an entire word, e.g., a 128 bit word, can be written into all odd (or even) cells of the same row. To do this, WWL corresponding to that particular row should be activated and appropriate write voltage should be provided through odd (even) write voltage drivers, as shown in Fig. 8. This will flow the write current in the appropriate direction through all SHMs corresponding to that particular row, and hence an entire word will be written to the memory. A notable fact is that our proposed architecture can allow writing two complete words, e.g., two 128 bit words, in parallel at the same time. However, this can be done only if one word is written in any of the odd (or even) rows, then the other word could be written in any of the even (or odd) rows, otherwise sneak path will arise corrupting actual data to be written.

Memory read operation can be carried out in parallel, i.e., entire word can be read out in the same cycle. To do so, one SA is needed for each bitline [53], as shown in Fig. 9. By simultaneously activating corresponding RWL and connecting all RBLs, it will read all the bits stored in the same selected word line. Fig. 9 shows complete word read operation for row1 with RWL1.

Please note that compute operation cannot be done in parallel for a complete word. Since all PLs are connected together in the logic mode, multiple even (odd) rows cannot be activated if any of the odd (even) rows are activated. Otherwise, compute current path will have multiple sneak paths, thus leading to erroneous logic results.

V. Device- and Circuit-Level Performance Evaluation

The device-level simulation is conducted by using the modular approach [43] for SS [38], which is a common framework connecting from basic device materials and physical phenomena all the way to circuits and systems [39]. The modules used for modeling the proposed device are described in Section III, and the materials and parameters used for each module are reported in Table II. For circuit-level simulation, 45 nm North Carolina State University (NCSU) Product Development Kit (PDK) Library [54] is used in SPICE to verify the proposed design and evaluate the performance.

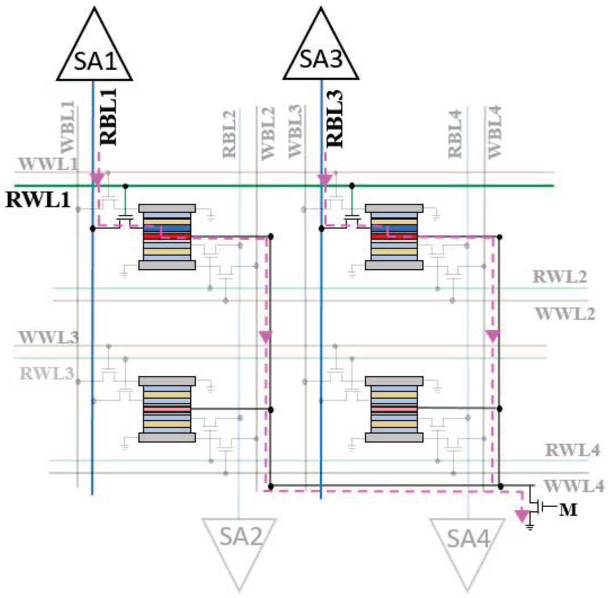


Fig. 9. Memory read operation for a complete word.

TABLE V
Comparison of Memory Mode Operation of Single Cell

	Standard STT MRAM [55]	1R/1W SOT MRAM [27]	IMCS2
Write access transistor width	500nm	180nm	240nm
Read access transistor width		180nm	90nm
Write Voltage	1	0.3/-0.2	0.18V
Avg. Write Power (μW)	266.44	17.80	20.61
Avg. Read Power (μW)	30.71	10.86	15.8
Compute (AND/NAND) Power (μW)	N/A	N/A	14.61
Compute (OR/NOR) Power (μW)	N/A	N/A	15.24
Compute (XOR/XNOR) Power (μW)	N/A	N/A	31.93
Write Delay (ns)	>10	10	9.8

Current requirements for memory write and read operations for the proposed coterminous SS device structure are determined by simulating the device framework built by interconnecting the modules [39], [43] to co-simulate with interface CMOS circuits in SPICE. Current required to be flowed through SHM layers to program (or write, in other words) the FLs varies with the variation of the width of the write access transistor (transistor with its gate connected with the corresponding WWL, as shown in Fig. 4) and the variation of voltage applied through the write voltage driver. Write current, delay, and power consumption for different transistor widths and voltages are extensively investigated, as shown in Fig. 10.

Current required for reading MTJ resistance is much lower compared to write current. Here, $\sim$5.6 μA sense current is flowed for 1 ns through the IMA MTJ during read. Again, due to the existence of the insulating coupling layer, risk of read disturbance is almost negligible. Here, computation operation is done in the same way as read operation by flowing $\sim$5 μA current for 1 ns.

Transistor widths for both write and read access transistors are provided in Table V, where memory mode performances are evaluated and compared with standard STT-MRAM [55] and contemporary 1R/1W SOT-MRAM [27] designs. To make the comparison fair, write delay is kept to be $\sim$10 ns with $\sim$114.5 μA write current through SHM. This requires write

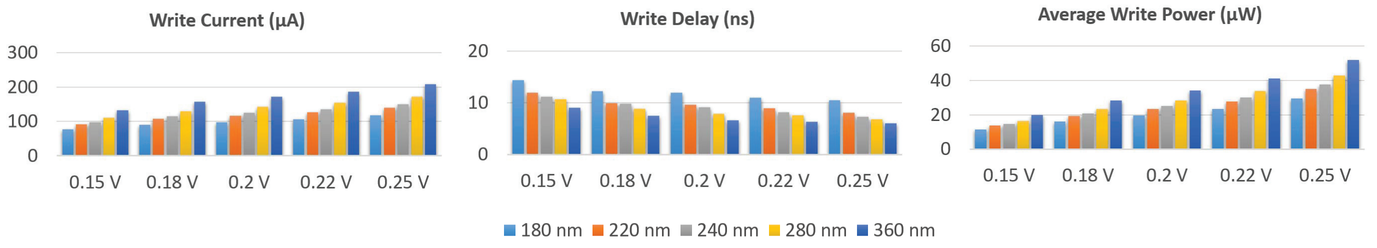


Fig. 10. Variation of write current, delay, and power with the variation of write access transistor size and write voltage.

access transistor width to be ∼240 nm for ±0.18 V write voltage.

Note that logic AND/OR output is almost similar to reading data from memory array through SA1. Hence, equivalent average logic (AND/OR) power is almost similar to the average read power. Again, logic XOR operation is similar to reading out two memory cells simultaneously using both SA1 and SA2. Hence, average logic (XOR) power should be almost twice as the average read power of single cell.

From Table V, it can be seen that our proposed design (IMCS2) offers significant improvement both in terms of read and write power than standard STT MRAM. It requires 92.26% less average write power and 47.57% less average read power than standard STT MRAM [55]. However, it requires 13.63% more average write power than contemporary 1R/1W SOT-MRAM design [27]. This is because of the fact that coupling mechanism between two FLs requires more power to enforce torque on the FL to change its magnetization. Again, average read power requirement is 31.27% more than contemporary 1R/1W SOT-MRAM design [27], which comes from different SA designs.

Please note that our proposed IMCS2 architecture is capable of performing in-memory computation of all Boolean logic functions (AND/NAND, OR/NOR, and XOR/XNOR), which other memory architectures [27], [55] cannot perform, with very small power and area overhead than contemporary 1R/1W SOT-MRAM design [27].

VI. System-Level Performance Evaluation

In this section, system-level memory mode performance of the proposed SS-based memory architecture is evaluated. For the simulation, modified self-consistent NVSim [56] along with an in-house developed C++ code is employed to verify system-level performance and to report accurate latency, energy, and area. Here, memory chip organization is configured by dividing it into multiple banks (bank organization: total 4×4 and active 1×1) consisting of multiple mats (mat organization: total 2×2 and active 1×1). Each mat includes multiple sub-arrays (sub-array size: 1024×512) organized in an H-tree routing manner. Table VI tabulates and compares the performance of our design with different memory arrays employed for in-memory processing in previous works (i.e., SRAM, DRAM, and standard STT-MRAM) for a sample memory capacity of 4 MB in 45 nm process node.

TABLE VI
SRAM, DRAM, STT-MRAM, and Proposed Design's Memory Model Validation and Comparison for a Sample 4 MB Memory

	SRAM		DRAM		STT-MRAM		IMCS2	
Metrics	Write	Read	Write	Read	Write	Read	Write	Read
Latency (ns)	1.53		2.7		11.13	2.62	10.91	2.94
Dynamic Energy (pJ)	297.46	312.56	967	1483	1219	791.13	815.2	821.65
Leakage Power (mW)	5258		185.5		830.3		810.01	
Area (mm^2)	10.544		4.702		4.611		6.12	

According to Table VI, our proposed IMCS2 memory model shows lower dynamic energy in comparison with DRAM. Besides, IMCS2 greatly reduces total leakage power compared with SRAM. Although the proposed design requires almost the same write latency as standard STT-MRAM, both designs have shown longer write latency compared with SRAM due to longer write latency of magnetic memory storage devices. Moreover, area overhead for IMCS2 is ∼23% more than DRAM and standard STT-MRAM design but still ∼42% less than SRAM design. It is worth noting that the first and foremost benefit of spintronic memories, compared with SRAM and DRAM, is their non-volatility with almost 10 years' retention time [27].

VII. Reliability Analysis

A. Read Disturbance Failure

Read disturbance arises from the risk of flipping a bit stored in memory while reading. Since sense current flows through MTJ, bits can accidentally get flipped during sensing due to current-induced STT [26]. However, here during memory read and computing operations, sense current flowed through the MTJ is ∼5.6μA, which is almost 10× lower than the critical current required for flipping the MTJ magnetization within 1 ns. Hence, the risk of bit flip during sensing is negligible. Moreover, due to the insulating coupling layer, read and write current paths are isolated. Again, coupling layer couples the magnetization of two FLs, making it too sturdy to get flipped by small sense current.

B. Read Decision Failure

Read decision failure may arise from ambiguity or failure in determining the correct state of MTJ while reading due to stochastic variation of the device's characteristic parameters (e.g., TMR) by fluctuations in temperature or process variation. One significant measure for determining the risk of read decision failure is voltage margin, i.e., the higher (lower) the voltage margin, the more (less) robust the proposed design is against variations. Variation tolerance of sense circuit is investigated by performing Monte Carlo simulation with 10000 iteration cycles and 5% stochastic variation in both MTJ resistance-area product (RAP) and TMR. Simulation result of sense voltage distribution (Fig. 11) shows ∼42.5 mV voltage

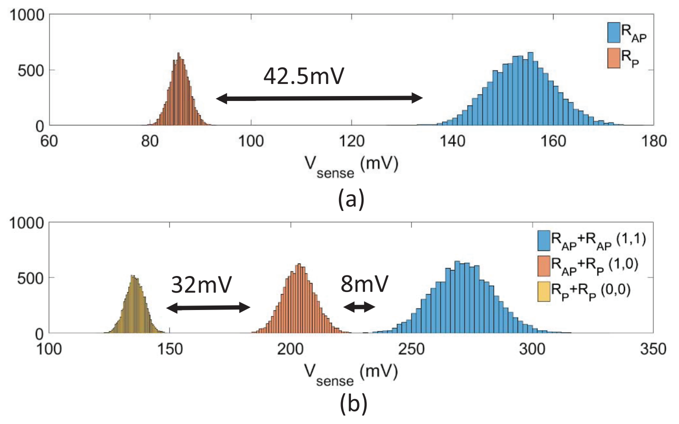


Fig. 11. Monte Carlo simulation result of sense voltage distribution for (a) memory read operation and (b) computing operation (with sense current = 5.6 μA).

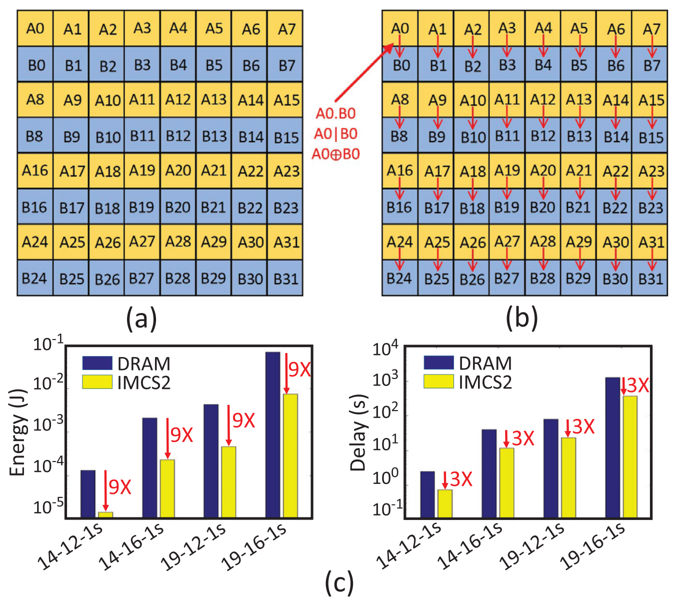


Fig. 12. Bitwise AND/OR operation for different vector data sets compared with DRAM. (a) Data Mapping for an 8×8 array. (b) Computing cycles. (c) Energy saving and speedup.

margin for memory read operation and ~8 mV voltage margin for computing operation in the worst case (between $R_P + R_P$ and $R_P + R_{\mathrm{AP}}$). This indicates that voltage margin is much lower for the case of computing operation compared with memory read operation. This is because computing operation incorporates two MTJs. The impact of MTJ RAP and TMR variation is more pronounced for computing operation than that for memory read operation. This could be improved by providing larger sense current with a tradeoff of larger sensing power consumption. Hence, a current driver could be used along with the SA to provide lower sense current (~5 μA) during memory read and higher sense current (>5 μA) during computing operation. This can lead to more reliable and variation tolerant operation without any significant sacrifice in power efficiency.

VIII. Case Studies

A. Bulk Bitwise Boolean Operation

The proposed in-memory computing architecture could be leveraged to implement bulk bitwise Boolean logic operations between two vectors stored in the same memory sub-array. This can lead to efficient re-use of the internal memory bandwidth without any add-on circuit as in typical logic-in-memory designs. Data mapping and performance evaluation for bitwise vector operations are given below.

1) Data Mapping: To perform bitwise logic (AND/OR/XOR) operation between two vectors, data from these two vectors are needed to be stored in the same memory sub-array. For example, for computing bitwise AND (/OR/XOR) operation between two vectors A and B, each being 32 bits long, odd rows (the top SS of each proposed coterminous device) of the memory array are written with data from vector A and even rows (the bottom SS of each proposed coterminous device) of the memory array are written with data from vector B, as shown in Fig. 12(a).

As explained in Section IV-D, data can be written in an entire row in a single clock cycle. Again, due to isolated current paths, odd rows and even rows can be written simultaneously. Here, for two 32 bit vectors, one 8×8 memory array can be used, where vector A (vector B) is written in our odd (even) rows. Hence, all the data can be written in four clock cycles (1 ns each), where in each clock cycle, one pair of rows (odd and even) can be written simultaneously. After data are written into the array, bitwise AND/OR/XOR operation can be performed between two corresponding bits of A and B. Here, computation operation cannot be done in parallel as mentioned in Section IV-D. Hence, for 32 bit vectors, it will take 32 cycles to complete the bitwise logic operation between two entire words. Hence, in total (write + sense) for 32 bitwise logic operation in 8×8 memory array with the given data mapping, it should take $4 + 32 = 36$ clock cycles.

2) Performance Evaluation: Performance of bulk bitwise vector operation using our proposed architecture is evaluated using similar simulation framework as described in Section V. Performance for bulk bitwise operations leveraging in-memory computing platform using DRAM [15] is also evaluated. We developed an in-house simulation platform incorporating $6F^2$ DRAM cell structure with 16 fF cell capacitance [57] using 45 nm technology node. Complete compute operation is completed within three consecutive operation cycles—precharge, access, and sense [15], [58], assuming data are already written in the memory array. Computation of AND/OR logic function is implemented within DRAM using three input majority gate design [15], with two data bits and one dummy bit as the three inputs. Energy and latency for bitwise AND/OR operations for different vector data set [16] are calculated. Please note that 19-16-1 s refers to a vector data set with vector length 2^{19} and number of vectors 2^{16}, and AND/OR operation is done between 2^1 rows, where "s" means sequentially. Fig. 12 shows the energy saving and speedup of in-memory bulk bitwise vector operation using our proposed IMCS2 array over that using DRAM array. Here, in-memory computing platform using our proposed memory array architecture offers ~9× energy saving and ~3× speedup compared with that using DRAM-based in-memory

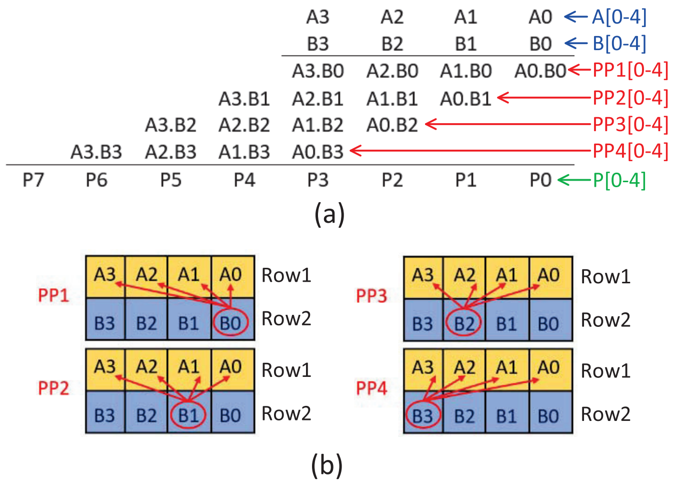

A3 A2 A1 A0 ← A[0-4]
B3 B2 B1 B0 ← B[0-4]
A3.B0 A2.B0 A1.B0 A0.B0 ← PP1[0-4]
A3.B1 A2.B1 A1.B1 A0.B1 ← PP2[0-4]
A3.B2 A2.B2 A1.B2 A0.B2 ← PP3[0-4]
A3.B3 A2.B3 A1.B3 A0.B3 ← PP4[0-4]
P7 P6 P5 P4 P3 P2 P1 P0 ← P[0-4]
(a)
PP1
PP2
PP3
PP4
Row1
Row2
(b)

Fig. 13. (a) Vector multiplication for two four-bit vectors. (b) Data mapping for computation of PPs using the proposed architecture.

computing platform [15]. Again, computation in DRAM is destructive, i.e., data stored in three DRAM cells associated with the computation are overwritten with the result of the logic operation, whereas stored data are retained even after computing in our SOT-MRAM-based design.

B. In-Memory Vector Multiplication

Aforementioned in Section IV-B2, logic AND/OR/XOR operations can be performed between any two bits stored in any odd row and in any even row [clarified in Fig. 6(b)]. In other words, there is no restriction that the two operands are needed to be written in adjacent cells. This inherent flexibility of performing logic operations could be exploited to implement an in-memory multiplier using the proposed architecture. Fig. 13 explains the multiplication operation between two vectors A and B along with data mapping for interpretation of implementation of a multiplier in the proposed memory array.

1) Data Mapping: A vector multiplier can be implemented using two sub-arrays: 1) partial product (PP) generation sub-array and 2) addition sub-array (adder). Data mapping and steps for calculating the PPs from PP sub-array are shown in Fig. 13(b), which depicts each PP vector is following an algorithm like a "nested for loop." The algorithm is given in Algorithm 1, which shows, calculation of PPs can be done by first determining all PPs incorporating B0, then doing the same for B1, B2, B3, and so on, using the same functionality [as portrayed in Fig. 13(b)]. This nested loop could be easily implemented using our proposed architecture by exploiting the inherent flexible computing operation between any two bits of A and B without any add-on circuit. Please note that since computation operation cannot be done in parallel, the PPs are calculated serially in bitwise fashion.

PPs obtained from PP sub-array are then written in adder sub-array. Now, in the adder sub-array, calculation of final product terms P[0-7] can be carried out by adding corresponding PP terms using the algorithm given in Algorithm 1.

Interestingly, this in-memory vector–vector multiplication technique can also be extended to implement in-memory vector–matrix and matrix–matrix multiplication using the proposed architecture.

Algorithm 1 Vector Multiplication

```
\\ assume, vector and matrix indices start from 0

\\ Algorithm for PP sub-array
Inputs: A ← m bit vector;
        B ← n bit vector;
Output: PP ← n by [(n+m)-1] matrix;

    Initialization:
1.  PP ← initialize with zeros;
2.  for i ← 0; i<n; i ← i+1 do
3.      for j ← 0; j<m; j ← j+1 do
4.          PP[i][j] ← B[i] and A[j];
5.      end for
6.  end for
7.  return PP;

\\ Algorithm for Adder sub-array
Input:      PP ← n by [(n+m)−1] matrix;
Output:     P ← n+m bit vector;
Variables:  sum ← m bit vector;
            carry ← 1 bit;
            temp ← 1 bit;

    Initialization:
1.  sum ← initialize with zeros;
2.
3.  \\ Half Adder Module
4.  Module HA (x, y)
5.      sum[0] ← x xor y;
6.      carry ← x and y;
7.  End module
8.
9.  \\ Full Adder Module
10. Module FA (x, y, carry)
11.     (sum[0], temp) ← HA (x, y);
12.     (sum[0], carry) ← HA(sum[0], carry);
13.     carry ← carry or temp;
14. End Module
15.
16. P[0] ← PP[0][0];
17. sum[0] to sum[m-2] ← PP[0][1] to PP[0][m−1];
18. for i ← 1; i<n; i ← i+1 do
19.     (P[i], temp) ← HA (PP[i][0], sum[0]);
20.     for j ← 1; j<m-1; j ← j+1
21.         (sum[j−1], temp) ← FA(PP[i][j],
            sum[j], temp);
22.     end for
23.     (sum[j−1], sum[j]) ← FA(PP[i][j], sum[j], temp);
24. end for
25.
26. P[i] to P[(n+m)-1] ← sum[0] to sum[m−1];
27.
28. Return P;
```

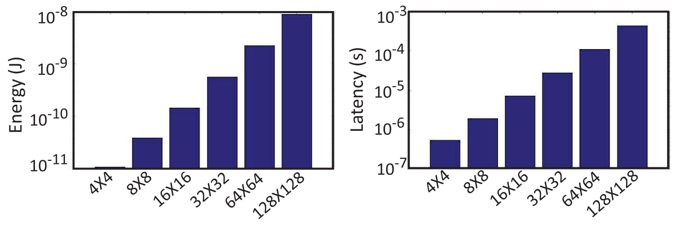


Fig. 14. Energy and latency of implementation of in-memory multiplier for $N \times N$ multiplier with $N = 4$, 8, 16, 32, 64, and 128.

2) Performance Evaluation: Here, multiplication energy and latency for $N \times N$ multiplier implemented using our proposed in-memory computing platform is demonstrated in Fig. 14, for different values of N. While calculating the energy and latency, internal read and write back energy and latency are taken into account. Please note that performance of the in-memory multiplier operation using our proposed architecture is evaluated using similar simulation framework as described in Section V.

IX. Conclusion

In this paper, we proposed a novel dual-mode in-memory computing architecture using a novel coterminous SS structure IMCS2. Extensive device-, circuit-, and system-level simulation results reveal significant performance improvement over standard STT-MRAM- and SRAM-based memory architectures. Our proposed design can be exploited for both memory and logic operations with reasonably comparative speed and power consumption with respect to the other contemporary SOT-MRAM designs. Furthermore, we considered two case studies including bulk bitwise in-memory vector AND/OR operation and in-memory multiplier to elucidate its in-memory computing performance. Device-to-architecture co-simulation results show that bitwise vector operation can offer ~9× energy saving and ~3× speedup compared with that using DRAM-based in-memory computing platform. To summarize, by adding several significant features as non-volatility, in-memory logic operation, zero leakage power, low dynamic power consumption, high packing density etc.; our proposed design can thrive a new paradigm for future power efficient in-memory computing platform.

Acknowledgment

This work was supported in part by the National Science Foundation under Grant 1740126 and in part by the Semiconductor Research Corporation nCORE. The authors would like to thank Dr. S. Ganguli for helpful co-operations regarding the usage of the SS model.

References

[1] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. ISCA*, vol. 43. Jun. 2016, pp. 27–39.
[2] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, Mar. 1995.
[3] Y. Wang *et al.*, "An energy-efficient nonvolatile in-memory computing architecture for extreme learning machine by domain-wall nanowire devices," *IEEE Trans. Nanotechnol.*, vol. 14, no. 6, pp. 998–1012, Nov. 2015.
[4] D. Patterson *et al.*, "Intelligent RAM (IRAM): Chips that remember and compute," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 1997, pp. 224–225.
[5] M. Oskin, F. T. Chong, and T. Sherwood, "Active pages: A computation model for intelligent memory," *ACM SIGARCH Comput. Archit. News*, vol. 26, no. 3, pp. 192–203, 1998.
[6] E. Riedel, C. Faloutsos, G. A. Gibson, and D. Nagle, "Active disks for large-scale data processing," *Computer*, vol. 34, no. 6, pp. 68–74, Jun. 2001.
[7] R. Nair *et al.*, "Active memory cube: A processing-in-memory architecture for exascale systems," *IBM J. Res. Develop.*, vol. 59, nos. 2–3, pp. 17:1–17:14, Mar./May 2015.
[8] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim, "NDA: Near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2015, pp. 283–295.
[9] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski, "TOP-PIM: Throughput-oriented programmable processing in memory," in *Proc. 23rd Int. Symp. High-Perform. Parallel Distrib. Comput.*, 2014, pp. 85–98.
[10] J. Ahn, S. Yoo, O. Mutlu, and K. Choi, "PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture," in *Proc. ACM/IEEE 42nd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2015, pp. 336–348.
[11] J. T. Pawlowski, "Hybrid memory cube (HMC)," in *Proc. IEEE Hot Chips 23 Symp. (HCS)*, Aug. 2011, pp. 1–24.
[12] Q. Zhu, K. Vaidyanathan, O. Shacham, M. Horowitz, L. Pileggi, and F. Franchetti, "Design automation framework for application-specific logic-in-memory blocks," in *Proc. IEEE 23rd Int. Conf. Appl.-Specific Syst., Archit. Process. (ASAP)*, Jul. 2012, pp. 125–132.
[13] J.-P. Wang and J. D. Harms, "General structure for computational random access memory (CRAM)," U.S. Patent 9 224 447B2, Dec. 29, 2015.
[14] Q. Guo, X. Guo, R. Patel, E. Ipek, and E. G. Friedman, "AC-DIMM: Associative computing with STT-MRAM," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 189–200, 2013.
[15] V. Seshadri *et al.*, "Fast bulk bitwise AND and OR in DRAM," *IEEE Comput. Archit. Lett.*, vol. 14, no. 2, pp. 127–131, Jul./Dec. 2015.
[16] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.
[17] J. Draper *et al.*, "The architecture of the DIVA processing-in-memory chip," in *Proc. 16th Int. Conf. Super Comput.*, 2002, pp. 14–25.
[18] Z. He, S. Angizi, F. Parveen, and D. Fan, "Leveraging dual-mode magnetic crossbar for ultra-low energy in-memory data encryption," in *Proc. Great Lakes Symp. VLSI*, 2017, pp. 83–88.
[19] S. Angizi, Z. He, and D. Fan, "Energy efficient in-memory computing platform based on 4-terminal spin Hall effect-driven domain wall motion devices," in *Proc. Great Lakes Symp. VLSI*, 2017, pp. 77–82.
[20] D. Fan, "Low power in-memory computing platform with four terminal magnetic domain wall motion devices," in *Proc. IEEE/ACM Int. Symp. Nanosc. Archit. (NANOARCH)*, Jul. 2016, pp. 153–158.
[21] D. G. Elliott, W. M. Snelgrove, and M. Stumm, "Computational RAM: A memory-SIMD hybrid and its application to DSP," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1992, pp. 30.6.1–30.6.4.
[22] P. M. Kogge, "EXECUBE—A new architecture for scaleable MPPs," in *Proc. Int. Conf. Parallel Process. (ICPP)*, vol. 1. Aug. 1994, pp. 77–84.
[23] D. Patterson *et al.*, "A case for intelligent RAM," *IEEE Micro*, vol. 17, no. 2, pp. 34–44, Mar. 1997.
[24] Y. Kim, S. K. Gupta, S. P. Park, G. Panagopoulos, and K. Roy, "Write-optimized reliable design of STT MRAM," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, Aug. 2012, pp. 3–8.
[25] X. Chen *et al.*, "Energy-aware adaptive restore schemes for MLC STT-RAM cache," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 786–798, May 2017.
[26] X. Fong *et al.*, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 1, pp. 1–22, Jan. 2016.
[27] Y. Seo, K.-W. Kwon, X. Fong, and K. Roy, "High performance and energy-efficient on-chip cache using dual port (1R/1W) spin-orbit torque MRAM," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 3, pp. 293–304, Sep. 2016.
[28] X. Guo, E. Ipek, and T. Soyata, "Resistive computation: Avoiding the power wall with low-leakage, STT-MRAM based computing," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 371–382, 2010.

[29] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan. (2017). “Computing in memory with spin-transfer torque magnetic RAM.” [Online]. Available: https://arxiv.org/abs/1703.02118

[30] W. Kang, H. Wang, Z. Wang, Y. Zhang, and W. Zhao, “In-memory processing paradigm for bitwise logic operations in STT–MRAM,” *IEEE Trans. Magn.*, vol. 53, no. 11, Nov. 2017, Art. no. 6202404.

[31] W. Kang, Z. Wang, Y. Zhang, J. O. Klein, W. Lv, and W. Zhao, “Spintronic logic design methodology based on spin Hall effect–driven magnetic tunnel junctions,” *J. Phys. D, Appl. Phys.*, vol. 49, p. 065008, Feb. 2016.

[32] H. Zhang, W. Kang, L. Wang, K. L. Wang, and W. Zhao, “Stateful reconfigurable logic via a single-voltage-gated spin Hall-effect driven magnetic tunnel junction in a spintronic memory,” *IEEE Trans. Electron Devices*, vol. 64, no. 10, pp. 4295–4301, Oct. 2017.

[33] K.-W. Kwon, X. Fong, P. Wijesinghe, P. Panda, and K. Roy, “High-density and robust STT-MRAM array through device/circuit/architecture interactions,” *IEEE Trans. Nanotechnol.*, vol. 14, no. 6, pp. 1024–1034, Nov. 2015.

[34] B. Del Bel, J. Kim, C. H. Kim, and S. S. Sapatnekar, “Improving STT-MRAM density through multibit error correction,” in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6.

[35] W. Kang *et al.*, “Yield and reliability improvement techniques for emerging nonvolatile STT-MRAM,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 1, pp. 28–39, Mar. 2015.

[36] 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.

[37] G. Prenat *et al.*, “Ultra-fast and high-reliability SOT-MRAM: From cache replacement to normally-off computing,” *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 1, pp. 49–60, Mar. 2016.

[38] S. Datta, S. Salahuddin, and B. Behin-Aein, “Non-volatile spin switch for Boolean and non-Boolean logic,” *Appl. Phys. Lett.*, vol. 101, no. 25, p. 252411, 2012.

[39] S. Ganguly, K. Y. Camsari, and S. Datta. (2016). “Evaluating spintronic devices using the modular approach.” [Online]. Available: https://arxiv.org/abs/1608.03681

[40] Y. Kim, X. Fong, K.-W. Kwon, M.-C. Chen, and K. Roy, “Multilevel spin-orbit torque MRAMs,” *IEEE Trans. Electron Devices*, vol. 62, no. 2, pp. 561–568, Feb. 2015.

[41] F. Parveen, S. Angizi, Z. He, and D. Fan, “Low power in-memory computing based on dual-mode SOT-MRAM,” in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2017, pp. 1–6.

[42] G. Prenat, K. Jabeur, G. Di Pendina, O. Boulle, and G. Gaudin, “Beyond STT-MRAM, spin orbit torque RAM SOT-MRAM for high speed and high reliability applications,” in *Proc. Spintronics-Based Comput.*, 2015, pp. 145–157.

[43] K. Y. Camsari, S. Ganguly, and S. Datta, “Modular approach to spintronics,” *Sci. Rep.*, vol. 5, 2015, Art. no. 10571.

[44] G. Autès, J. Mathon, and A. Umerski, “Strong enhancement of the tunneling magnetoresistance by electron filtering in an Fe/MgO/Fe/GaAs(001) junction,” *Phys. Rev. Lett.*, vol. 104, no. 21, p. 217202, 2010.

[45] C.-F. Pai, L. Liu, Y. Li, H. W. Tseng, D. C. Ralph, and R. A. Buhrman, “Spin transfer torque devices utilizing the giant spin Hall effect of tungsten,” *Appl. Phys. Lett.*, vol. 101, no. 12, p. 122404, 2012.

[46] V. Sokalski *et al.*, “Naturally oxidized FeCo as a magnetic coupling layer for electrically isolated read/write paths in mLogic,” *IEEE Trans. Magn.*, vol. 49, no. 7, pp. 4351–4354, Jul. 2013.

[47] A. V. Penumatcha, C.-C. Lin, V. Q. Diep, S. Datta, J. Appenzeller, and Z. Chen, “Impact of scaling on the dipolar coupling in magnet–insulator–magnet structures,” *IEEE Trans. Magn.*, vol. 52, no. 1, pp. 1–7, Jan. 2016.

[48] K. Jabeur, G. Di Pendina, G. Prenat, L. D. Buda-Prejbeanu, and B. Dieny, “Compact modeling of a magnetic tunnel junction based on spin orbit torque,” *IEEE Trans. Magn.*, vol. 50, no. 7, pp. 1–8, Jul. 2014.

[49] J. Z. Sun *et al.*, “Spin-torque switching efficiency in CoFeB-MgO based tunnel junctions,” *Phys. Rev. B, Condens. Matter*, vol. 88, no. 10, p. 104426, Sep. 2013.

[50] (2016). *The Object Oriented Micromagnetic Framework (OOMMF) Project at ITL/NIST*. [Online]. Available: http://math.nist.gov/oommf/

[51] W. Kang, L. Zhang, J.-O. Klein, Y. Zhang, D. Ravelosona, and W. Zhao, “Reconfigurable codesign of STT-MRAM under process variations in deeply scaled technology,” *IEEE Trans. Electron Devices*, vol. 62, no. 6, pp. 1769–1777, Jun. 2015.

[52] Z. He and D. Fan, “A low power current-mode flash ADC with spin hall effect based multi-threshold comparator,” in *Proc. Int. Symp. Low Power Electron. Design*, 2016, pp. 314–319.

[53] R. Bishnoi, M. Ebrahimi, F. Oboril, and M. B. Tahoori, “Architectural aspects in design and analysis of SOT-based memories,” in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2014, pp. 700–707.

[54] (2011). *FreePDK45:Contents*. [Online]. Available:http://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[55] X. Bi, M. A. Weldon, and H. Li, “STT-RAM designs supporting dual-port accesses,” in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2013, pp. 853–858.

[56] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, “NVSim: A circuit-level performance, energy, and area model for emerging non-volatile memory,” in *Proc. Emerg. Memory Technol.*, 2014, pp. 15–50.

[57] Y. Yanagawa, T. Sekiguchi, A. Kotabe, K. Ono, and R. Takemura, “In-substrate-bitline sense amplifier with array-noise-gating scheme for low-noise 4F^2 DRAM array operable at 10-ff cell capacitance,” in *Proc. VLSIC*, Jun. 2011, pp. 230–231.

[58] D. T. Wang and B. L. Jacob, “Modern DRAM memory systems: Performance analysis and scheduling algorithm,” Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Maryland, College Park, MD, USA, 2005.

Farhana Parveen (S’12) received the B.Sc. and M.Sc. degrees in electrical and electronics engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2014 and 2016, respectively. She is currently pursuing the Ph.D. degree in electrical engineering with the University of Central Florida, Orlando, FL, USA.

Her current research interests include emerging technology-based non-volatile memory design, in-memory computing platform design, and re-configurable computing platform design.

Shaahin Angizi (S’15) received the B.Sc. degree in computer engineering, hardware from the South Tehran Branch, Islamic Azad University (IAU), Tehran, Iran, in 2012, and the M.Sc. degree in computer engineering, computer systems architecture from the Science and Research Branch, IAU, Tabriz, Iran, in 2014. He is currently pursuing the Ph.D. degree in computer engineering with the University of Central Florida, Orlando, FL, USA.

His current research interests include low-power very large scale integration designs, spin-based computing, and quantum-dot cellular automata.

Zhezhi He (S’16) received the B.S. degree in information science and engineering from Southeast University, Nanjing, China, in 2012, and the M.E. degree in electrical and computer engineering from Oregon State University, Corvallis, OR, USA, in 2015. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Central Florida, Orlando, FL, USA.

His current research interests include neuromorphic computing, analog and mixed signal circuit design, and emerging technology.

Deliang Fan (M’15) received the B.S. degree in electronic information engineering from Zhejiang University, Hangzhou, China, in 2010, and the M.S. and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2012 and 2015, respectively.

In 2015, he joined the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, USA, as an Assistant Professor. His current research interests include ultralow-power brain-inspired (neuromorphic), non-Boolean, and Boolean computing using emerging nanoscale devices, such as spin transfer torque devices and memristors, nanoscale physics-based spintronic device modeling and simulation, and low-power digital and mixed-signal CMOS circuit design.