Hybrid Spin-CMOS Polymorphic Logic Gate With Application in In-Memory Computing

Shaahin Angizi[1], Zhezhi He[2], An Chen[3], and Deliang Fan[2]

[1]Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816 USA
[2]School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA
[3]Semiconductor Research Corporation (SRC), Durham, NC 27703 USA

In this article, we initially present a hybrid spin-CMOS polymorphic logic gate (HPLG) using a novel 5-terminal magnetic domain wall motion device. The proposed HPLG is able to perform a full set of 1- and 2-input Boolean logic functions (i.e., NOT, AND/NAND, OR/NOR, and XOR/XNOR) by configuring the applied keys. We further show that our proposed HPLG could become a promising hardware security primitive to address IC counterfeiting or reverse engineering by logic locking and polymorphic transformation. The experimental results on a set of ISCAS-89, ITC-99, and École Polytechnique Fédérale de Lausanne (EPFL) benchmarks show that HPLG obtains up to 51.4% and 10% average performance improvements on the power-delay product (PDP) compared with recent non-volatile logic and CMOS-based designs, respectively. We then leverage this gate to realize a novel processing-in-memory architecture (HPLG-PIM) for highly flexible, efficient, and secure logic computation. Instead of integrating complex logic units in cost-sensitive memory, this architecture exploits a hardware-friendly approach to implement the complex logic functions between multiple operands combining a reconfigurable sense amplifier and an HPLG unit to reduce the latency and the power-hungry data movement further. The device-to-architecture co-simulation results for widely used graph processing tasks running on three social network data sets indicate roughly 3.6× higher energy efficiency and 5.3× speedup over recent resistive RAM (ReRAM) accelerators. In addition, an HPLG-PIM achieves ∼4× higher energy efficiency and 5.1× speedup over recent processing-in-DRAM acceleration methods.

Index Terms— **Domain wall motion (DWM), in-memory computing, polymorphic gate, spintronics.**

I. Introduction

RECENTLY, polymorphic logic has become a promising hardware security primitive in logic locking or polymorphic transformation to address security issues, such as integrated circuit counterfeiting and reverse engineering [1]–[4]. Polymorphic logic was first introduced in 2004 [5], by embedding multiple functionalities within the same cell using some controllable factors to determine the input–output correlation. For instance, voltage or temperature can be used as the controlling factor to configure the cell's functionality in a variety of applications [6].

Typically, polymorphic logic is implemented through reconfigurable hardware designs with conventional CMOS-based circuits [7], [8]. However, the need for reconfiguration gives rise to issues like large-area overhead and power consumption, increased complexity, and so on. To counter these issues, exploitation of emerging devices for implementing polymorphic functions is being investigated, such as polymorphic designs with SiNW FET and FinFETs [6], [9]. However, these polymorphic gates could only morph between two preset logic functions (not all functions), leading to high relative complexity (i.e., ratio of the number of transistors with respect to the number of implementable logic functions) and limited efficiency in logic obfuscation/encryption applications. With this in mind, a power-efficient and compact polymorphic logic gate that could morph between all logic functions is yet to come.

From a different perspective, recently, processing-in-memory (PIM) architectures, as a potentially viable way to solve the memory wall challenge in the conventional Von Neumann computer system (e.g., long memory access latency, significant congestion at I/Os, limited memory bandwidth, and huge leakage power), have been well explored [10]–[12]. The key concept behind the PIM is to embed logic units (not necessarily polymorphic) within memory to exploit the external and internal memory bandwidths better. This can lead to remarkable savings in data communication energy and latency besides providing an inherent in-memory parallelism for data processing. The proposals for exploiting DRAM- [11] and SRAM-based [12] PIM architectures can be found in the recent literatures. However, they encounter inevitable drawbacks, such as high leakage power or initial data overwritten, that hinder their further processing. This topic has become even more intriguing by emerging non-volatile memory (NVM) technologies, such as phase-change memory (PCM) [13], resistive RAM (ReRAM) [10], and magnetic RAM (MRAM) [14]. In particular, MRAM technology has recently become a promising high-performance NVM candidate for both last level cache and main memory, owing to its fast read/write characteristics. Hence, PIM in the context of different NVMs, specially using MRAM [15], [16], without sacrificing memory capacity can open a new way to efficient computing paradigm.

Representative works of such PIM architectures include Pinatubo [17] as a general architecture capable of doing

Manuscript received December 17, 2018; revised September 24, 2019; accepted November 19, 2019. Date of current version January 20, 2020. Corresponding author: S. Angizi (e-mail: angizi@knights.ucf.edu).

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

Digital Object Identifier 10.1109/TMAG.2019.2955626

bulk bitwise operations, multi-purpose in-memory processing (MPIM) [18] as a multi-purpose ReRAM-based PIM, PRIME [10], and ISAAC [19] as dot product engines for neural network acceleration based on ReRAM, RIMPA [20] as a threshold logic PIM architecture based on domain wall-RAM (DW-RAM), and spin-transfer torque (STT)-MRAM-based PIM in [15] and [16]. All these architectures can be reconfigured to memory and computing modes where the modified sense amplifiers (SAs) typically play a major role in performing rowwise or columnwise in-memory logic.

Nonetheless, recent PIM designs have several limitations. Although they are well optimized for two-operand bulk bitwise operations, they still unavoidably rely on the external processing unit in order to perform more complex in-memory logics; otherwise, the performance degradation of the PIM could be significant due to multi-cycle operations and unnecessary write-backs. For example, performing AND2 function between A and B (row vectors) can be made in a single cycle in most NVM- and SRAM-based PIMs [12], [17], [20]–[22]; however, when it comes to $(A \cdot B) \oplus$ (C+D), such platforms require at least six cycles (if support in-memory XOR2) to make it. As a result, current platforms cannot properly use their in-memory computation capacity in such operations. To compensate this issue, two solutions have been put forward, either sacrificing the PIM speed by keeping data inside memory and handling multi-cycle overloads [17], [21] or transmitting data to a digital processing unit [23], [24] and imposing power hungry long-distance data communication.

For clarification, this article is an extension of our previously published work in [25], where we proposed a hybrid spin-CMOS polymorphic logic gate (HPLG) based on a new composite 5-terminal multi-layer magnetic DW motion (DWM) device. HPLG is able to implement a full set of 1- and 2-input Boolean logic functions (i.e., NOT, AND/NAND, OR/NOR, and XOR/XNOR) by configuring the applied keys without any structural alteration. In this article, we show that HPLG can be a prospective candidate in hardware security applications, in addition to offering low power consumption, low relative complexity, high compactness, and polymorphism in logic circuits. In addition, we leverage this design to realize an efficient PIM platform. Our main contributions in this article can be briefly listed as follows.

1) We further investigate and show the performance improvement of HPLG utilization in three different logic benchmarks compared with CMOS and recent spin-CMOS non-volatile logic designs.
2) We design a reconfigurable PIM architecture based on spin-orbit torque (SOT)-MRAM, called *HPLG-PIM*, based on a set of novel micro-architectural and circuit-level schemes. It positions the *HPLG-PIM* as a massive data-parallel computational unit, which owns three distinct operating modes (i.e., memory, computing, and boosted computing).
3) We pave a new way to push the boundaries of PIM using HPLG further, so that complex bitwise computations can be performed between the operands within memory. This further reduces the latency and the energy-concerning state-of-the-art PIM hardware.

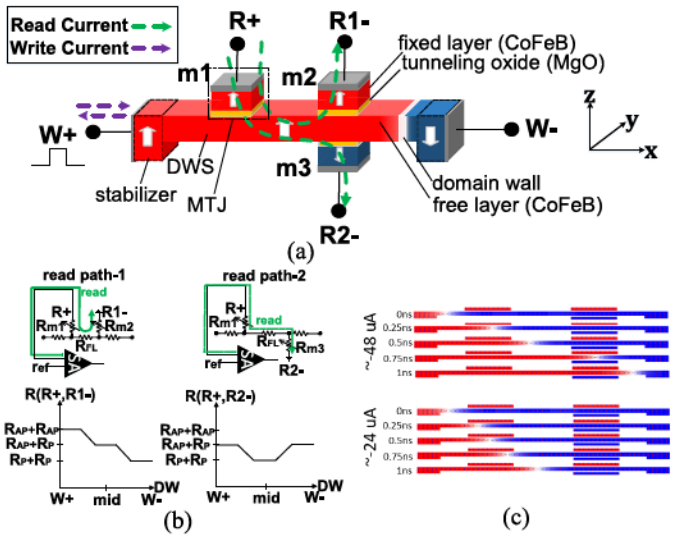


Fig. 1. (a) Proposed 5T-DWM device structure. (b) Resistive model of two read paths and total magnetoresistances of each path for different DW positions. (c) Micromagnetic simulation for writing current from W+ to W−.

4) We demonstrate how *HPLG-PIM* can handle and accelerate graph processing workloads by developing new mapping and partitioning methods. Moreover, the proposed scheme is evaluated using a variety of real-world social network graph data compared with other state-of-the-art acceleration solutions, i.e., DRAM, ReRAM, and STT-MRAM.

The rest of this article is organized as follows. We present the novel spintronic device structure and HPLG design in Sections II and III, respectively. Section IV focuses on designing the *HPLG-PIM* architecture. In Section V, the systematic performance evaluation of the HPLG and the proposed PIM platform is provided. Section VI presents and evaluates the *HPLG-PIM*'s acceleration method for a widely used graph-processing task. Finally, Section VII concludes this article.

II. Composite Spintronic Device Structure

The proposed five-terminal DWM (5T-DWM) device, as shown in Fig. 1(a), is a composite multi-layer structure consisting of a DW strip (DWS) and three sensing magnetic tunnel junctions (MTJ). Note that a typical MTJ consists of two ferromagnetic (FM) layers (e.g., CoFeB [26]) with a tunnel barrier (e.g., MgO [27]) sandwiched between them. One of the magnetic layers is a fixed layer, while the other one is a free layer. Due to the tunneling magnetoresistance (TMR) effect [27], the resistance of the MTJ is high (or low) when the magnetization of two FM layers are anti-parallel (AP) [or parallel (P)]. The free-layer magnetization could be manipulated by applying an external magnetic field or through the current-induced STT [14]. A DWS is typically a wire-like magnetic nano-strip (e.g., CoFeB, Co, and Ni [28]), in which multiple magnetic domains can be formed [29]. The transition region between two opposite magnetic domains is called DW. When electrons flow through the DWS from one end to the opposite end, they continuously track the local magnetization

profile through exchanging their angular momentum with the local magnetic moment. As a result, the local magnetic moment in the DWS experiences a spin-transfer torque if the neighboring moments are not aligned in the same direction. Hence, the spin-transfer torque is exerted on the magnetic moments in and around the DW region when current flows through the DWS. Such a current-induced spin-transfer torque can displace the DW along the direction of electron flow, which is then observed as DWM along the magnetic nano-strip. Fukami *et al.* [30] experimentally demonstrate steady DWM with a current density above 6×10^{11} A/m^2 and 60 m/s DW velocity for 20 nm-wide magnetic nano-strips.

Fig. 1(a) shows the 5T-DWM device structure with the five terminals as Write terminals, W+ and W−, and Read (or Sense) terminals, R+, R1−, and R2−. Our proposed device structure consists of a thin (90 nm × 20 nm × 2 nm) magnetic DWS and three MTJs ($m1$, $m2$, and $m3$). The MTJs are formed by placing a small fixed nano-magnet (20 nm × 20 nm) as the sensing head on the top and bottom of the DWS, with a thin MgO layer (1.5 nm) between the nano-magnet and the DWS. Here, the nano-magnets work as the "fixed" FM layers and the DWS works as the "free" FM layer while forming the sense MTJs. The free magnetic layer DWS is laterally connected to two anti-parallel fixed magnetic domains of larger thickness so-called stabilizer. This larger thickness at the edges of the free layer (DWS) is used to stabilize the DW at an intermediate position within the free layer, which has been experimentally proven in [30]. In order to have a better controllability and thermal stability of DWS, three artificial trapping sites located in the W+ end, middle, and W− end could be manufactured [31].

In the proposed design, the three sensing terminals (i.e., R+, R1−, and R2−) lead to two read paths [see Fig. 1(b)], and thus to two different resistance behaviors corresponding to DW locations: 1) resistance from R+ to R1−, which is the series resistance of $m1$ and $m2$, represented as R(R+, R1−) and 2) resistance from R+ to R2−, which is the series resistance of $m1$ and $m3$, represented as R(R+, R2−). Such a design leads to unique sensing resistance response based on the DW positions within the nano-strip, as plotted in Fig. 1(b). The fixed magnetic layers of $m1$ and $m2$ have the same magnetization direction as the stabilizer at the W+ terminal, while the fixed magnetic layer of $m3$ has the same magnetization direction as the stabilizer at the W− terminal. Due to the tunneling TMR effect, the resistance of the MTJ depends on the magnetization directions of the two FM layers on both sides of the insulator layer. The resistances of the sensing MTJs depend on the magnetization of DWS underneath, i.e., the DW positions. In this design, the resistance of $m1$ and $m2$ will be high, R_{AP} (or low, R_P), if the DW is at the W+ (or W−) side of the DWS due to anti-parallel (or parallel) magnetization. The resistance of $m1$ and $m2$ will be different if the DW is in the middle. In addition, due to different fixed magnetic layers of $m2$ and $m3$, they have differential resistance states with the same DW positions. In summary, the three MTJ resistances are determined by the DW positions within the DWS, which will be leveraged to design our polymorphic logic gate circuit in Section III.

TABLE I
PARAMETERS USED IN MICRO-MAGNETIC DEVICE SIMULATION

Symbol	Quantity	Values
α	Damping coefficient	0.02
K_u	Uniaxial anisotropy constant	$3.5 \times 10^5 J/m^3$
M_s	Saturation magnetization	$6.8 \times 10^5 A/m$
A_{ex}	Exchange stiffness	$1.1 \times 10^{-11} J/m$
P	Polarization	0.6
t_{MgO}	MgO thickness	1.5 nm
$(L.W)_{MTJ}$	MTJ area	$20 \times 20nm^2$
$(L.W.t)_{DWS}$	DWS dimension	$90 \times 20 \times 2nm^3$

In order to simulate the proposed 5T-DWM device precisely with the CMOS interface circuits in SPICE, the device is modeled as two MTJs with variable resistance depending on DW positions in each read path. The resistive models for both read paths are shown in Fig. 1(b). Considering the resistivity of the DWS, an equivalent resistive network for read path-1 can be written as

$$R = R_{\mathrm{m1}} + R_{\mathrm{FL}} + R_{\mathrm{m2}} = \begin{cases} 2RA_{\mathrm{AP}}/A_{\mathrm{MTJ}} + R_{\mathrm{FL}}, & \text{DW at left end} \\ 2RA_{\mathrm{P}}/A_{\mathrm{MTJ}} + R_{\mathrm{FL}}, & \text{DW at right end} \\ (RA_{\mathrm{AP}} + RA_{\mathrm{P}})/A_{\mathrm{MTJ}} + R_{\mathrm{FL}}, & \text{DW at middle} \end{cases} \quad (1)$$

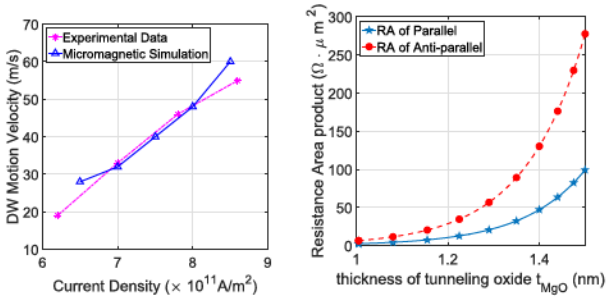
where R_{m1} and R_{m2} represent the resistances of two read MTJs, which are shown by $m1$ and $m2$, respectively, R_{FL} is the lateral free-layer resistance between the two read MTJs, $\mathrm{RA_{AP}}$ and $\mathrm{RA_P}$ denote the MTJ resistance–area (RA) product for anti-parallel and parallel configurations, respectively, obtained in non-equilibrium Green's function (NEGF)-based MTJ model [32], and A_{MTJ} is the read MTJ area. Therefore, the output resistance of read path-1 can have three different values based on the DW positions.

The output resistance (from R+ to R2−) of the 5T-DWM device can be expressed as follows:

$$R = R_{\mathrm{m1}} + R_{\mathrm{FL}} + R_{\mathrm{m3}} = \begin{cases} (RA_{\mathrm{P}} + RA_{\mathrm{AP}})/A_{\mathrm{MTJ}} + R_{\mathrm{FL}}, & \text{DW at left/right end} \\ 2RA_{\mathrm{P}}/A_{\mathrm{MTJ}} + R_{\mathrm{FL}}, & \text{DW at middle} \end{cases} . \quad (2)$$

Thus, the read path-2 has only two distinct resistance levels based on its DW positions and the output resistances are identical when the DW is positioned in the left or right end.

The magnetic device dynamics in write path is simulated in the object-oriented micro-magnetic framework (OOMMF [33]) based on the well-known Landau–Lifshitz–Gilbert (LLG) [14] equation with the device parameters listed in Table I. Fig. 1(c) shows the transient micromagnetic simulation of the DWS with the lateral currents of ∼48 and ∼24 μA from W− to W+ (electron flow is from W+ to W−). It can be seen that the DW is moved from W+ to W− (or middle) of the DWS within ∼1 ns for ∼48 μA (or ∼24 μA) current. We benchmarked the micro-magnetic simulation with the experimental data in [30] (the same nano-stripe width of 20 nm is fabricated) and it shows a good match as shown in Fig. 2(a). Note that the simulated magnetic DWS has perpendicular magnetic anisotropy (PMA), which is the same as the fabricated device in [30]. The MTJ is modeled in



(a) (b)

Fig. 2. (a) Simulated DWM velocity versus lateral current density, showing a good match with experimental data reported in [30]. (b) RA product versus the thickness of tunneling oxide in AP and P state (with 50 mV constant voltage).

Verilog-A, using the NEGF-LLG solution for spin-to-charge interface [32]. The MTJ RA product versus tunneling oxide thickness in anti-parallel (AP) and parallel (P) states is plotted in Fig. 2(b). Note that the presented 5T-DWM device supports two pivotal operations: 1) DWM can be precisely controlled by the magnitude and direction of the laterally applied current, with the assistance of the notches on the DWS and 2) the MTJs mounted on the DWS can have configurable resistances within the sensing path. The aforementioned design concepts have been experimentally demonstrated in [34] and [35].

III. Hybrid Polymorphic Logic Gate

A. Circuit Design

As shown in Fig. 3, the proposed HPLG circuit is designed with three keys (K1, K2, and K3), which consists of a 5T-DWM device and 13 transistors. Among these 13 transistors, six transistors are used to control the direction of current flow through the 5T-DWM device and the other seven transistors are used to form a differential latch to sense the logic output. The differential latch compares the 5T-DWM device resistance in the read path with a reference MTJ (resistance value shown and explained later).

The HPLG is able to perform the NOT, AND/NAND, OR/NOR, and XOR/XNOR Boolean logic functions based on different key configurations. Note that all the input transistors are designed to work in the deep triode region by applying ΔV (100 mV) across the drain and source (VDS$\sim$100 mV) terminals. Hence, it will lead to ultra-small voltage drop and, thus, ultra-small power consumption. For a complete Boolean operation, the HPLG requires three subsequent stages, i.e., Reset, Compute, and Sense. Based on our micromagnetic simulation described in Section II, the threshold current to move the DW from W$-$ to W$+$ is $\sim$48 μA within 1 ns. Each input transistor (i.e., A, B, K1, and R_{st}), if turned on, is designed to provide $\sim$24 μA (or $\sim-$24 μA) to DWS by proper sizing. In the Reset stage ($R_{st} = K1 = 1$), the reset and K1 transistors are turned on for 1 ns. Then, a current of $\sim$48 μA flows from W$-$ to W$+$ terminal to initialize the DW position at the W$-$ side. In the compute stage, as shown in Fig. 3, operands (A, B) and one particular key (K1) are

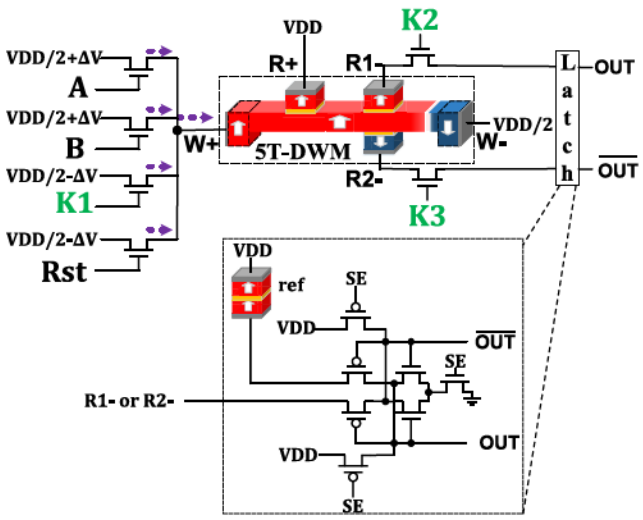


Fig. 3. Proposed hybrid polymorphic logic gate (HPLG) circuit design based on 5T-DWM device.

applied to control the input transistors and further determine the DW position within DWS for different logic functions.

Table II lists the detailed HPLG operations with respect to the combination of different inputs and logic function configured by specific keys. Note that I_{DWM} is the total current flowing from W$+$ to W$-$. We define that the positive current is from W$+$ to W$-$ and the negative current is from W$-$ to W$+$. For AND/NAND, OR/NOR, and NOT Boolean logic functions, the summation of $m1$ and $m2$ resistances [i.e., R(R$+$, R1$-$) in Fig. 1(b)] represents the output state. Now, it can be seen that, the output resistance (summation of $m1$ and $m2$ resistances) is lower than the reference MTJ (located in the Latch unit shown in Fig. 3), only when both $m1$ and $m2$ have parallel configurations ($R_P + R_P$, DW at W$-$ side). Note that the reference MTJ resistance is $0.5\times[(R_P+R_P)+(R_P+R_{AP})]$, where R_P and R_{AP} are the parallel and anti-parallel MTJ resistances, respectively. For all the other DW positions, the output resistance will be higher than the reference MTJ, since the summation of $m1$ and $m2$ resistances is either $R_P + R_{AP}$ (when DW is in the middle) or $R_{AP} + R_{AP}$ (when DW is at the W$+$ side). This can be leveraged for implementing AND/NAND, OR/NOR, and NOT gate by configuring different biasing conditions.

1) *Implementation of* AND/NAND *Gate:* For implementing the AND/NAND gate, we set K1=1 (corresponding transistor source voltage is VDD/2-ΔV). This causes $\sim-$24 μA current flowing from W$-$ to W$+$. Note that a $\sim$24 μA will be injected into the DWS if the corresponding input (A or B) is high (corresponding transistor drain voltage is VDD/2+ΔV). Therefore, the net current flowing into the DWS (i.e., W$+$ terminal) is the summation current from A, B, and K1 transistors. If K1=1 ($\sim-$24 μA), the net current flowing into the DWS is either $\sim-$24 μA or around zero when no input is high or only one input is high, respectively. Thus, the DW will remain at the initial W$-$ side, which will consequently make the output low. For the other case

TABLE II
IMPLEMENTATION OF DIFFERENT LOGIC FUNCTIONS USING HPLG

Inputs	A B	0 0	0 1	1 0	1 1
AND/ NAND	I_{DWM}	$-26.2\mu A$	$2.14\mu A$	$1.24\mu A$	$26.05\mu A$
	DW	W-	W-	W-	middle
	$R_{m1}+R_{m2}$	R_P+R_P	R_P+R_P	R_P+R_P	$R_{AP}+R_{AP}$
Keys: K1=1, K2=1, K3=0	$\overline{OUT}$ (AND)	0	0	0	1
	OUT (NAND)	1	1	1	0
OR/ NOR	I_{DWM}	3.82nA	$27.57\mu A$	$27.57\mu A$	$50.62\mu A$
	DW	W-	middle	middle	W+
	$R_{m1}+R_{m2}$	R_P+R_P	R_P+R_{AP}	R_P+R_{AP}	$R_{AP}+R_{AP}$
Keys: K1=0, K2=1, K3=0	$\overline{OUT}$ (OR)	0	1	1	1
	OUT (NOR)	1	0	0	0
NOT(A) B=0	I_{DWM}	$3.82nA$		$27.57\mu A$	
	DW	W-		middle	
Keys: K1=0, K2=1, K3=0	$R_{m1}+R_{m2}$	R_P+R_P		R_P+R_{AP}	
	$\overline{OUT}$ (NOT)	1		0	
XOR /XNOR	I_{DWM}	3.82nA	$27.57\mu A$	$27.57\mu A$	$50.62\mu A$
	DW	W-	middle	middle	W+
	$R_{m1}+R_{m3}$	R_P+R_{AP}	R_P+R_P	R_P+R_P	R_P+R_{AP}
Keys: K1=0, K2=0, K3=1	$\overline{OUT}$ (XNOR)	1	0	0	1
	OUT (XOR)	0	1	1	0

Note: I_{DWM}= Total current flowing from W+ to W-, DW= DW position in DWS, $R_{m1}+R_{m2}$ = Summation of resistances of m1 and m2, i.e. R(R+, R1-), $R_{m1}+R_{m3}$ = Summation of resistances of m1 and m3, i.e. R(R+, R2-), R_P is the resistance of MTJ with parallel configuration, R_{AP} is the resistance of MTJ with anti-parallel configuration.

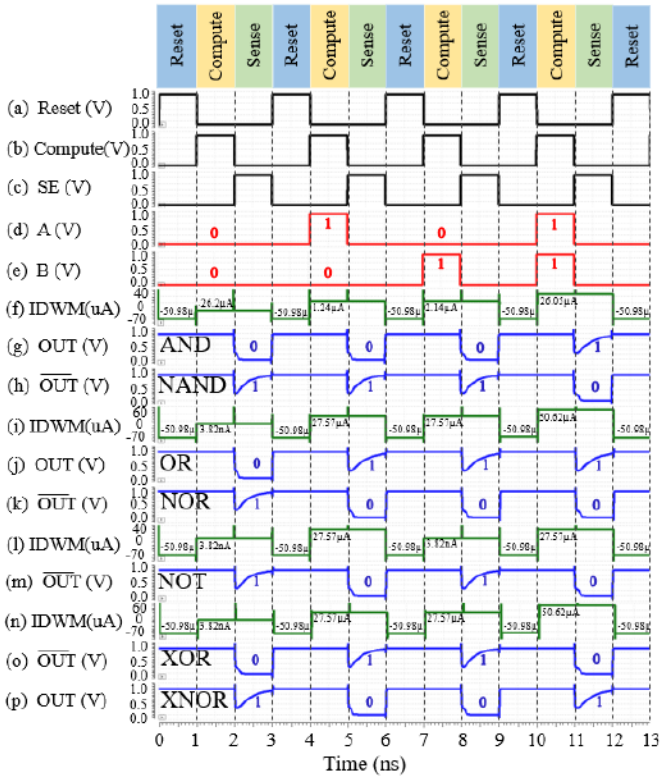


Fig. 4. Transient simulation results for different logic gates implemented by HPLG. (a)–(c) Control signals. (d) and (e) Input signals. (f)–(h) AND/NAND gate implementation with [K1,K2,K3] = [1,1,0]. (i)–(k) OR/NOR gate implementation with [K1,K2,K3]=[0,1,0]. (l) and (m) NOT gate implementation with [K1,K2,K3]=[0,1,0] and B=0. (n)–(p) XOR/XNOR gate implementation with [K1,K2,K3]=[0,0,1].

(both A and B are high), the net current is around 24 μA, which will move DW to the middle notch, resulting in high output. Note that the actual simulated net current under different logic inputs is shown in Table II, which might be slightly different from the ideal case. However, the logic functionality will not be disturbed. When the computation is done, the differential latch working as a sensing circuit (SE=1) is used to compare the read path resistance (i.e., R(R+, R1−) by setting K2=1 and K3=0 with a reference MTJ. Hence, after computation, we can get the result of AND (or NAND) operation from the OUT (or $\overline{OUT}$) port of the differential latch.

2) *Implementation of* OR/NOR *Gate:* For implementing the OR/NOR gate, we set K1=0. Now the DW will remain at the W− side only if both A and B are low, which will consequently make the output low. For the other cases, when only one or both A and B are high, the DW will be moved to the middle or to the W+ side, respectively, thus leading to high output. Hence, after computation, we can get the result of OR (or NOR) operation from the OUT (or $\overline{OUT}$) port of the differential latch.

3) *Implementation of* NOT *Gate:* For implementing the NOT gate, we set K1=0, B=0, and A is the input operand. Thus, the DW position will depend only on A, i.e., if A is high (or low), then DW will be in the middle (or at W− side); thus, the output will be high (or low). Hence, after computation, we can get the result of NOT operation from the $\overline{OUT}$ port of the differential latch. The detailed circuit operations could be seen in Table II. Obviously, OUT port in this case results buffer output.

4) *Implementation of* XOR/XNOR *Gate:* For the XOR/XNOR logic function, the read path-2 [i.e., R(R+, R2−)] in Fig. 1(b) will be used to represent the output state. It can be seen from Table II that the output (summation of $m1$ and $m3$ resistances) is low, only when both $m1$ and $m3$ have parallel configurations ($R_P + R_P$, DW in the middle notch). For other DW positions (either at W+ or W− side), the output is high, since the summation of $m1$ and $m3$ resistances will be $R_P + R_{AP}$. This can be leveraged for implementing the XOR/XNOR gate by configuring the biasing conditions. If we set K1=0, based on the simulated net current flowing into DWS shown in Table II, the DW will be in the middle notch when only one input A or B is high. Then, the sensing resistance ($m1$ and $m3$) is $R_P + R_P$ and consequently generates low output. For the other cases, when both A and B are high or low, the DW will be at the W+ or W− side, respectively. Then, the sensing resistance is $R_P + R_{AP}$, and thus generates high output. When the computation phase is done, a differential latch working as a sensing circuit (SE=1) is used to compare the summation of m1 and m3 resistances (K2=0, K3=1) with the reference MTJ. Hence, we can get the result of XNOR (/XOR) operation from the OUT (or $\overline{OUT}$) port of the differential latch. The detailed circuit operation could be seen in Table II.

B. Device-/Circuit-Level Simulation

For device-level simulation, we benchmarked DWM dynamics with experimental data [30] using OOMMF [33]. The MTJ (consisted of DWS, tunneling oxide layer, and fixed FM layer) is modeled in the Verilog-A using NEGF-LLG (non-equilibrium Green's function and LLG equations) solution for the spin-to-charge interface and calibrated with data in [30] and [32]. For the circuit-level simulation, a Verilog-A model of the 5T-DWM device was developed to co-simulate with the interface CMOS circuits in Cadence Specter and SPICE. The 45 nm North Carolina State University (NCSU) product development kit (PDK) library [36] is used in SPICE to verify the proposed design and acquire the performance of designs. The circuit is simulated with three consecutive cycles—Reset, Compute, and Sense, where 1 ns pulsewidth is used for each cycle. The transient simulation plots for different logic functions implemented by HPLG are shown in Fig. 4. It can be seen that a complete set of Boolean logic could be achieved by setting different key values.

IV. HPLG-Based PIM Platform

In this section, we leverage the proposed HPLG to realize a configurable PIM platform called the *HPLG-PIM*. Fig. 5 depicts the *HPLG-PIM*'s sub-array architecture based on SOT-MRAM and detailed modified peripheral circuitry. The basic sub-array architecture mainly consists of a memory row decoder (MRD), a memory column decoder (MCD), a voltage driver (VD), an SA, and an HPLG. This architecture can be adjusted by the Ctrl unit Ⓐ to work in tri-mode that performs memory read-write, in-memory computing, and boosted computing operations.

Each SOT-MRAM cell [with the composite structure of spin Hall metal (SHM) and MTJ [14], [37]] is associated with the write word line (WWL), read word line (RWL), write bitline (WBL), read bitline (RBL), and source line (SL) to perform the memory operations. Here, in each memory cell, the resistance of the MTJ with parallel magnetization in both magnetic layers (data-"0") is lower than that of MTJ with anti-parallel magnetization (data-"1"). To program the free-layer magnetization, the flow of charge current through SHM (Tungsten, $\beta - W$ [38]) will cause the accumulation of opposite-directed spin on both the surfaces of SHM due to the spin Hall effect [14]. Thus, a spin current is generated and further produces an SOT on the adjacent free magnetic layer, causing switch of magnetization. In this article, the magnetization dynamics of the free FM layer is modeled by the LLG equation with the STT term and the SHE term, as used in [39]. Note that the ferromagnets in MTJ have in-plane magnetic anisotropy (IMA) [14]. With the given thickness (1.2 nm) of the tunneling layer (MgO), the TMR of the MTJ is ∼168.5% [40].

The VD component Ⓑ is modified such that can select between data input coming from different sources (i.e., the inter- and intra-sub-arrays, SA, and HPLG). The peripheral decoders (active-high output) control the activation of the current path through the array. MRD Ⓒ is modified based on the buffering method such that two WLs can be simultaneously selected. A reconfigurable voltage mode SA Ⓓ [41]

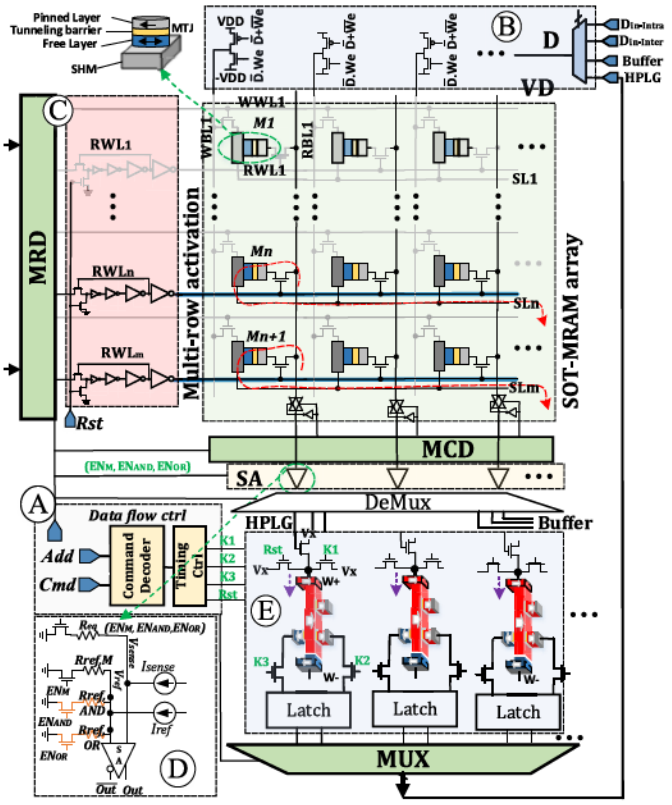


Fig. 5. Proposed triple-mode *HPLG-PIM* sub-array architecture.

is connected to the RBL for sensing the total resistance in the selected current path during the read or computing mode. Here, we elaborate the functionality of the proposed sub-array architecture.

1) Memory Mode: To *write* a data bit in any of the SOT-MRAM cells, write current should be injected through the heavy metal substrate of the SOT-MRAM. To activate this write current path (e.g., for *M1*), WWL1 should be activated by MRD and SL1 is grounded, while all the other word lines and SLs are kept deactivated (floating). Data (D) can come from different sources, and accordingly when write enable (We) is activated, write circuit can assign proper voltage (positive/negative) on WBL in order to write "1" (or "0"). This allows sufficient charge current (∼120 μA) flows from VD to ground (or ground to V1), leading to MTJ resistance in High-R_{AP} (or Low-R_P) encoded as data "1" (or "0"). For a typical memory *read* operation, a single memory cell is selected to compare its sense voltage (V_{sense}) with a reference voltage (V_{ref}) by injecting a small sense current (I_{sense}) through the selected SOT-MRAM cell. To activate this read current path, for example, for *M1*, RWL1 is activated, while SL1 is grounded and all the other word lines and SLs are kept deactivated. MCD activates the RBL1 line to be connected to the SA. Hence, a read current flows from the selected SOT-MRAM cell to ground, generating a sense voltage at the input of SA Ⓓ, which is compared with the memory mode reference voltage ($V_{sense,P} < V_{ref} < V_{sense,AP}$). This reference voltage generation branch is selected by setting the Enable

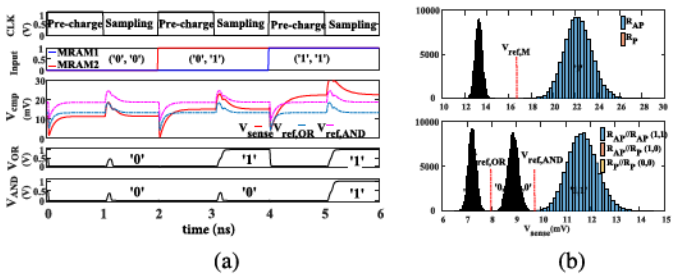


Fig. 6. (a) Transient simulation of AND/OR operations in computing mode. (b) Monte Carlo simulation result of V_{sense} distribution.

values $(EN_{\text{M}}, EN_{\text{AND}}, EN_{\text{OR}}) = (1, 0, 0)$. This selection is performed by the command issued by the Ctrl unit.

2) Computing Mode: In this read-like computation, every two bits stored in the identical column (i.e., *local operand*) can be selected and sensed simultaneously. To activate the computing current path (as shown in Fig. 5), two RWLs (here RWLn and RWLm) are activated by MRD Ⓒ, while SLn and SLm are grounded and all the other word/SLs are kept floating. Then, the equivalent resistance of such parallel-connected SOT-MRAM cells (Mn and Mm) and their cascaded access transistors are compared with a specific reference by SA Ⓓ. Through selecting different reference resistances (EN_{M}, EN_{AND}, and EN_{OR}), SA can perform basic in-memory Boolean functions (i.e., AND/NAND and OR/NOR). For the AND operation, R_{ref} is set at the midpoint of $R_{\text{AP}}//R_{\text{P}}$ ("1,""0") and $R_{\text{AP}}//R_{\text{AP}}$ ("1," "1"), and for OR operation, R_{ref} is set at the midpoint of $R_{\text{P}}//R_{\text{P}}$ and $R_{\text{P}}//R_{\text{AP}}$.

Fig. 6(a) shows the transient simulation results of the sense circuit under the 2 ns period clock signal (CLK). In the simulation, we take the data stored in MRAM1 and MRAM2 as inputs. When CLK is high, SA is in the pre-charging phase and the output is reset to "0." When CLK is low, the SA is in the sampling phase and generates logic computation result depending on the reference voltage configuration. Furthermore, to validate the variation tolerance of the sense circuit, we have performed Monte Carlo simulation with 100 000 trials. A $\sigma = 5\%$ variation is added on the RA product (RA_{P}), and a $\sigma = 10\%$ process variation is added on the TMR (typical MTJ conductance variation [42]). The simulation result of the sense voltage (V_{sense}) distributions in Fig. 6(b) shows the sense margin of such PIM architecture. It will be reduced by increasing the logic fan-in (i.e., number of parallel memory cells). Thus, to avoid the read failure (overlapping of V_{sense} distribution), only two fan-in rowwise in-memory logics are used. Note that parallel computing/read within a sub-array is implemented by using one SA per bitline with the exactly same mechanism as Fig. 5.

3) Boosted Computing Mode: In-memory computing can be even further boosted, leveraging HPLG as a low overhead and highly efficient solution in the *HPLG-PIM* platform. The key idea is to incorporate a reconfigurable HPLG per bitline in the memory sub-array after SAs, as shown in Fig. 5 Ⓔ. Combining the reconfigurability of SA and HPLG not only boosts memory sub-array performance in a sense that it can facile implementing complex in-memory logic functions but also can increase the protection against adversarial attacks on memory outputs. The output of each SA is connected to a Demultiplexer (DeMux). According to the mode selector, output data can be routed to either output buffers or HPLGs. To do the computation, a small modification is applied to HPLG's inputs. Rather than the one-time input injection shown in Fig. 3, the inputs coming from SA are consecutively written to the HPLG. The input transistor size of the HPLG is adjusted in a way that, if the SA output is "1," ∼24 μA current will flow from W+ to W− of the HPLG. As per the micromagnetic simulations shown in Fig. 1(c), this will move DW to the middle pinning site. On the other hand, if the SA output is "0," then no current will be injected to HPLG, and hence, the DW position will not change. The Ctrl Ⓐ only needs to generate four controlling signals to handle the HPLG unit based on different functions tabulated in Table II. We show the boosted computing mode can massively reduce power-hungry long-distance data communication between the processor and the memory by processing complex raw data, though it intrinsically imposes two memory cycles to implement in-memory operations.

Fig. 7 shows how the *HPLG-PIM*'s boosted computing mode can accelerate complex bulk logic functions as opposed to the recent bitline computing platforms. Here, we chose Pinatubo [17] as a promising PIM architecture supporting a wide range of in-memory functions. We take a simple function $f = x \cdot y + z \cdot w$ as an example. We observe that Pinatubo [see Fig. 7(b)] needs at least five cycles to realize such bitwise operation between x, y, z, and w rows. It first calculates the in-memory $x \cdot y$ function in a single cycle (1); however, it requires to write back the result to a temporary row of the sub-array (t_1) (2). Then, the same procedure needs to be done to implement the $z \cdot w$ function by saving the result in t_2 (3) and (4). Now, f can be implemented by performing the rowwise OR2 function on the temporary rows (5). However, the HPLG unit in *HPLG-PIM* can be used as a computational buffer to save the intermediate results of $x \cdot y$ and $z \cdot w$ in two consecutive cycles [see Fig. 7(a)]. Now, OR2 operation is already performed by setting the keys as (0,1,0) in the HPLG unit. The *HPLG-PIM*'s boosted computing mode also offers a *multi-row computation* method. In this method, exploiting the intrinsic feature of the HPLG unit, multiple operands can be individually read out and consecutively sent for computation. This one-cycle/one-operand approach may lead to a higher latency while working with a large number of operands though it offers an alternative way to overcome the small $R_{\text{ON}}/R_{\text{OFF}}$ ratio in most NVMs hindering multi-operand implementation.

V. Performance Evaluation

A. HPLG

In this section, we assess the performance of the proposed HPLG from different perspectives to further demonstrate its efficiency as a highly secured and standalone logic gate for replacing conventional designs.

1) Relative Complexity Analysis: The comparison among different polymorphic gate realizations is presented in Table III in terms of number of transistors and number of imple-

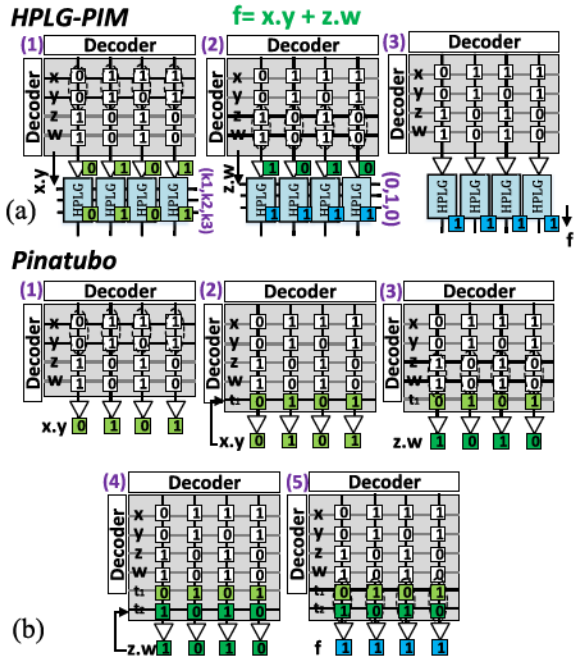


Fig. 7. Parallel computation with (a) *HPLG-PIM*'s boosted computing mode versus (b) recent bitline computing platforms such as Pinatubo [17].

TABLE III
COMPARISON OF DIFFERENT POLYMORPHIC GATES

Design	Morph method	#T†	#F‡	Rel. Comp. (#T/#F)
[5]	27/125°C Temperature	6	2	3
[6]	3.3/0.0/1.5V External Signal	6	2	3
[6]	3.3/0.0 V External Signal	10	3	3.33
[6]	0.0/0.9/1.1/1.8V External Signal	6	2	3
[6]	[6]1.2/3.3V Vdd	11	4	2.75
[6]	3.3/1.8V Vdd	8	2	4
[1]	0.0/3.3V External signal	9	2	4.5
[7]	Vdd and Gnd interchange	4	2	2
HPLG	3 Keys	13	7	1.86

(†) Number of transistors, (‡) Number of implementable functions

mentable functions. As can be seen, the HPLG is capable of implementing seven Boolean logic functions (including six 2-input functions as well as a NOT function) using only 13 transistors. Whereas other polymorphic logic gate designs could only morph between two to four functions [1], [5]–[7]. In addition, the comparison is made in terms of relative complexity as the ratio of number of transistors (or #T) with respect to the number of implementable functions (or #F). Visibly, the relative complexity of the HPLG is greatly reduced due to the fact that a single 5T-DWM device is used to perform all the logic functions without any structural alteration. Our proposed design shows ∼38% lower relative complexity than the polymorphic gate design that uses temperature [5] as the controlling factor. Again, our proposed design shows up to ∼58% lower relative complexity than the polymorphic gate design that uses voltage [1] as the controlling factor.

2) Security Analysis: The proposed HPLG can be utilized as a promising hardware security primitive. For example, several proposed key gates can be "inserted" into a combinational or sequential circuit to hide the functionality of the IC from the adversary, known as *"Logic Locking"* [2]. Another effective solution to the threat of reverse engineering is *"Polymorphic Transformation"* [1], where some of the logic gates of the IC can be "replaced" by HPLG, so-called partial replacement. Whereas a full replacement can provide the strongest protection against reverse engineering. In both cases, the keys for the proposed polymorphic logic gates can be placed within an IC in a tamper proof memory for activating IC after returning from the foundry [2].

Here, the un-trusted adversaries in the IC manufacturing and fabrication chain may prompt for IC counterfeiting or overproducing. However, since keys are not provided to the foundry due to the non-operational purpose, they cannot sell the overproduced ICs in the market unless they know the keys for activating the ICs. In case if they place an adversary as the endpoint user, who has access to the activated IC for operational use, a possible attack can be invoked based on the following attack model. It is assumed that the adversary has the following two things: 1) complete gate-level netlist and 2) activated IC (can be purchased from the open market). With this attack model, the adversary can know the correct outputs for all the input combinations. For example, for an M inputs IC, the adversary might know the outputs for all the 2^M input combinations. Now by applying *"brute force search"* or *"ATPG algorithm"* [2], they can try to find out keysets that can produce the appropriate input–output patterns. However, Subramanyan *et al.* [2] show that brute force and ATPG algorithms cannot expose all the keys if key-gates are inserted properly at certain critical positions within the circuit. Again, McDonald *et al.* [1] indicate that full replacement of logic gates by polymorphic gates can ensure strong protection against reverse engineering. Specially, the vulnerability decreases with the increase in the number of inputs and the length of the key-set. For our case, each of our logic gates requires three keys to produce the desired output.

Now, considering a full replacement for an M input and N gate circuit, the reverse engineer will need to test 2^{M+3N} input-key combinations for getting the appropriate output. For example, the simplest ISCAS-85 benchmark circuit C17 with five inputs and six gates can have $2^{5+3\times6} = 2^{23} = 83\,88\,608$ input-key combinations, whereas, C7552 with 207 inputs and 3512 gates can have $2^{10\,743} \approx \infty$. Hence, if a circuit has very large N and M, the adversary will need to test almost an infinite number of input-key combinations, making the reverse engineering process exhaustively difficult. Thus, the proposed HPLG is promising to implement a secured computing platform by providing logic locking and polymorphic transformation either in a full or partial replacement.

3) FA Design (A Case Study): To explore the efficiency of the proposed HPLG in real combinational circuit designs further, we consider it to realize a full adder (FA) cell. The schematic representation of an FA based on XOR and majority gate (MG) and an equivalent HPLG network is illustrated in Fig. 8(a). As shown, a hybrid spin-CMOS FA is designed employing three HPLGs with no additional circuit overhead by integrating HPLGs into main FA blocks. The *Sum* logic can be expressed as the three-input XOR of A, B, and C_{in} as input operands (A⊕ B ⊕ C_{in}). It can be seen that two cascaded HPLGs with proper configuration readily generate *Sum* output.

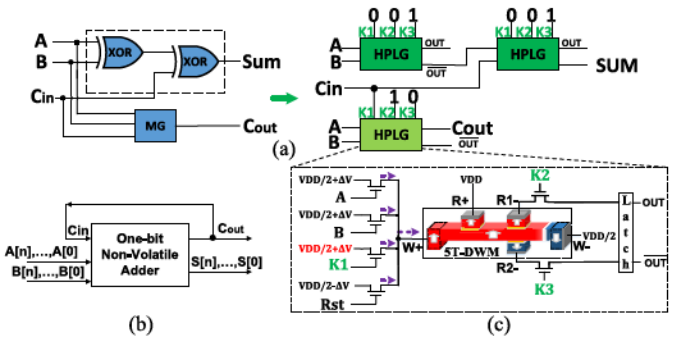


Fig. 8. (a) Schematic of an FA based on XOR and MG and equivalent HPLG network. (b) *N* bit serial adder structure based on the proposed one-bit adder. (c) Proposed modified HPLG for the implementation of MG logic.

TABLE IV
COMPARISON OF DIFFERENT FA DESIGNS

FA Design	Dynamic Power	Static Power	#Transistors
CMOS [43]	49.4μW	1.5nW	42
GSHE [43]	15.6μW	0.3nW	-
MTJ [44]	16.3μW	0.0nW	34
Proposed Design	12.73μW	0.0nW	39

TABLE V
BENCHMARK CHARACTERISTICS AND HPLG UTILIZATION IN THE FULL REPLACEMENT SCENARIO

ISCAS-89	Circuit Function	Inputs	Outputs	$\#FF^{\dagger}$	$\#HPLG^{\ddagger}$
s27	Logic	4	1	3	10
s298	Traffic light controller	3	6	14	119
s349	4-bit add-shift multiplier	9	11	15	161
s400	TLC	3	6	21	162
s420	Fractional multiplier	19	2	16	196
s526	TLC	3	6	21	193
s820	PLD	18	19	5	289
s838	Fractional multiplier	35	2	32	390
s1196	Logic	14	14	18	529
s1423	Logic	17	5	74	657
s15850	Logic	14	87	597	9772
s38584	Logic	12	278	1452	19253
ITC-99					
b02	FSM that recognizes BCD	1	1	4	22
b04	Compute min and max	11	8	66	597
b05	Elaborate the Mem	1	36	34	935
b10	Voting system	11	6	17	189
b11	Scramble string	7	6	31	481
b12	Guess a sequence	5	6	121	1036
b13	Interface to meteo sensors	10	10	53	339
b14	Viper processor (subset)	32	54	245	4775
EPFL					
Ctrl. I	ALU Control Unit	7	26	-	174
Ctrl. II	Mem. Controller	1204	1231	-	46836
Arith. I	Log2	32	32	-	32060
Arith. II	Square-root	64	128	-	18484

(†) Number of CMOS flip-flops used in different benchmark circuits, (‡) Number of utilized HPLGs in HPLG-100% implementation, where all the logic gates are replaced with HPLG. Note that the number of HPLGs used in HPLG-50% implementation is obtained by $\lfloor \frac{\#HPLG}{2} \rfloor$

The *Cout* logic can be simply implemented using a 3-input MG [20] as MG (A, B, and C_{in}) based on HPLG. For implementing the MG functionality, we need to switch the K1 transistor's source voltage from VDD/2-ΔV to VDD/2+ΔV [see Fig. 8(c)]. This will reverse the current direction through the transistor with K1. Hence, this K1 transistor will work as the third input operand of a 3-input MG with inputs as A, B, and K1. Now if at least two of the three inputs are high (majority of A, B, and K1 is high), the total current flowing from W+ to W$-$ will be at least $\sim$48 μA. Such current will move DW from W$-$ side to W+ as simulated in the micromagnetic simulation shown in Fig. 1(c), and the corresponding R(R+, R1$-$) is $R_{\text{AP}}+R_{\text{AP}}$. For the other cases (majority of A, B, and K1 is low), the DW will be either in the middle or at the W$-$ side, resulting in lower R(R+, R1$-$). Hence, using a reference MTJ of $0.5 \times [(R_{\text{P}} + R_{\text{AP}}) + (R_{\text{AP}} + R_{\text{AP}})]$, the 3-input MG can be readily implemented and *Cout* logic can be generated.

It is worth pointing out that due to the non-volatility of the proposed 1 bit FA, an *N* bit serial adder connecting the carry-out to carry-in can be readily designed, as shown in Fig. 8(b) [43]. Such design does not sacrifice the operation latency due to the fact that the higher bit should wait the carry-out signal from low bit. Thus, an *N* bit adder can be implemented by employing only one single non-volatile FA without extra overheads, leading to greatly reduced area and power consumption, while maintaining almost the same throughput [43].

The power consumption (at 500 MHz) and transistor count of the FA implemented using HPLG is compared with previously published CMOS, giant spin Hall effect (GSHE), and MTJ-based FAs [43], [44]. The comparison is shown in Table IV. It is being seen that FA design using HPLG offers up to 74.23% lower power consumption with respect to the conventional CMOS-based design [43]. It also shows 18.39% and 21.9% lower power consumption than other non-volatile FA designs that use GSHE-based devices [43] and MTJ-based FA [44]. Again, FA could be implemented using 7.14% lower transistor count with HPLG than the conventional FA design [43]. Note that the DWM device will not increase the chip size due to the hybrid spin-CMOS back end of line fabrication technology [30].

4) Benchmark Analysis: In this section, we evaluate the performance of the proposed HPLG compared with recent 2-input spin-CMOS non-volatile logic gates referred to as NVLs and the CMOS counterpart in different logic benchmarks with respect to the power-delay product (PDP). Contrary to CMOS-based implementations, NVLs typically enjoy the lower power consumption; however, they impose an increased delay to process the data [20], [45]. Thus, the PDP can be impartially used to compare CMOS- and NVL-based implementations [46]. Note that NVL1 [47] is a DW racetrack-based logic and NVL2 [45] is an MTJ-based logic without polymorphism, but capable of performing the basic logic functions. Accordingly, we developed the HPLG library based on device-/circuit-level evaluations, which contains a functionally complete set of Boolean logic gates as discussed earlier to co-simulate with CMOS circuits. The generated library is then used in a commercial synthesis tool, i.e., Synopsys Design Compiler, to map the produced optimized HDL code with an HPLG-based design. It is noteworthy that in order to have a fair comparison between CMOS-, NVL-, and HPLG-based designs, the same CMOS sequential logic circuit (flip-flop) is used for all designs.

Table V lists a set of three different benchmark circuits [i.e., ISCAS-89, ITC-99, and École Polytechnique Fédérale de Lausanne (EPFL)] with the number of inputs and used

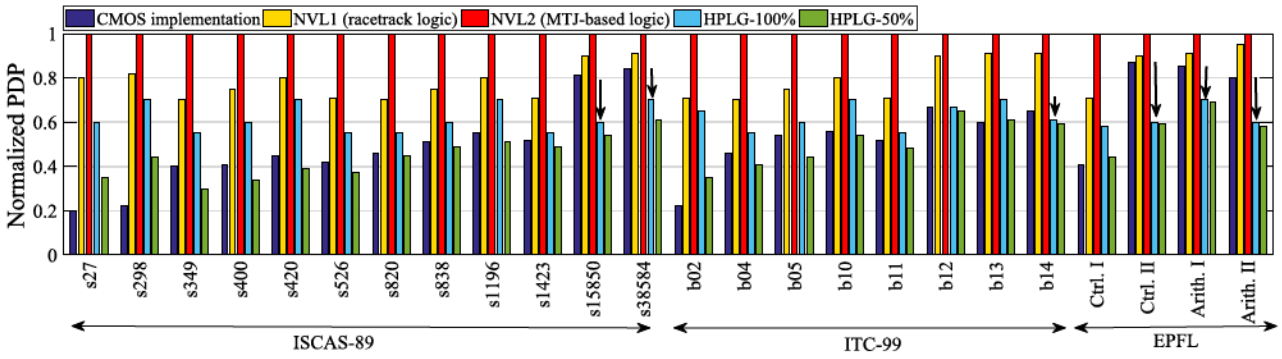


Fig. 9. Normalized PDP of different benchmark circuits for replacement with the proposed HPLG-, NVL1-, NVL2-, and CMOS-based circuit.

HPLGs. As discussed earlier, the larger these factors are, the more secure the computing platform is provided. Fig. 9 shows the normalized PDP analysis of CMOS- and NVL-based implementations compared with HPLG in different benchmark circuits. While we consider a full replacement of the benchmark circuits for CMOS- and NVL-based implementations, two distinct scenarios are considered to explore the performance of HPLG implementation: 1) a full replacement (100%) and 2) a partial replacement (50%), where half of the logic gates in each benchmark are randomly selected to be substituted with HPLG and the rest are CMOS-implemented. As can be seen, the proposed HPLG-100% implementation outperforms other NVLs in all the benchmark circuits. The obtained PDP results for CMOS implementation are also relatively lower than the other implementations owing to the high-speed switching of CMOS. However, by increasing the size of the benchmark, HPLG-100% circuits can show even superior performance as compared with CMOS (see s15850, s38584, b14, Ctrl. II, Arith. I, and Arith. II benchmark circuits). This PDP improvement compared with CMOS mainly comes from almost the zero static power of HPLG compared with the power-hungry CMOS, which compensates the long delay of the magnetic gates. On the other hand, the well-secured HPLG-50% implementation can show even more promising performance with up to 39.3%, 51.4%, and 10% average PDP improvements compared with NVL1, NVL2, and CMOS-based design, respectively. From the area overhead perspective, CMOS implementation shows the least footprint as compared with NVLs and HPLG-based designs. As a conclusion, the HPLG utilization rate can be precisely tailored by circuit designers according to specific constraints to provide a highly secured and efficient circuit. Note that the results provided herein are obtained at the gate level and physical design parameters are not considered within the document space available.

B. HPLG-PIM

In this section, we evaluate the performance of the proposed *HPLG-PIM* architecture from both memory and computation capability perspectives.

1) Memory Mode: To achieve the overall memory performance of the *HPLG-PIM* platform as an SOT-MRAM-based architecture with a modified peripheral circuitry, we extensively modified the system-level memory evaluation tool NVSim [48] to co-simulate with an in-house developed C++ code based on circuit-level results. Table VI tabulates and compares the memory performance [Write (W)/Read (R)] of the proposed design with three different memory candidates for a sample 4 MB memory chip in the 45 nm process node.

TABLE VI
MEMORY MODEL COMPARISON

	SRAM 4MB		DRAM 4MB		STT-MRAM 4MB		HPLG-PIM 4MB	
Metrics	W	R	W	R	W	R	W	R
Latency (ns)	1.07	0.9	2.7	2.4	10.2	1.08	1.29	1.15
Dynamic Energy (pJ)	297.4	312.5	967	1483	368.5	232.2	312.3	271.9
Leakage Power (mW)	5258		585.4		744.2		775.2	
Area (mm^2)	10.544		6.504		5.963		6.231	

In our simulations, we follow the iso-capacity constraint, where similar memory capacity is used for all the candidates. As expected, magnetic memories and DRAM show less area overhead compared to SRAM. The proposed memory model imposes ~40% less area compared with the SRAM with the same memory configuration. In addition, the magnetic memory models save a lot of leakage power compared with SRAM due to their non-volatility nature. Our design also outperforms other candidates in terms of dynamic energy owning to its low write voltage (~400 mV for "1" and ~−320 mV for "0"). Albeit *HPLG-PIM* improves the write energy and latency compared with the standard STT-MRAM and DRAM, all magnetic candidates have shown longer write latency than SRAM due to the longer write period of the magnetic memory storage devices.

2) Computing Modes: While the *HPLG-PIM* is especially designed to be a highly parallel and efficient PIM platform for bulk bitwise operations in structured applications, as will be discussed in Section VI, we first evaluate its performance in unstructured and bulk logic benchmarks. To assess the performance of *HPLG-PIM* as a new PIM platform, a comprehensive device-to-architecture evaluation framework along with two in-house simulators is developed. First, at the device level, we jointly use the NEGF and the LLG with the spin Hall effect equations to model the SOT-MRAM bit-cell [14], [49]. For the circuit-level simulation, a Verilog-A model of the 2T1R SOT-MRAM device is developed to co-simulate with the interface CMOS circuits in Cadence Specter and SPICE. The 45 nm North Carolina State University (NCSU) PDK library [36]

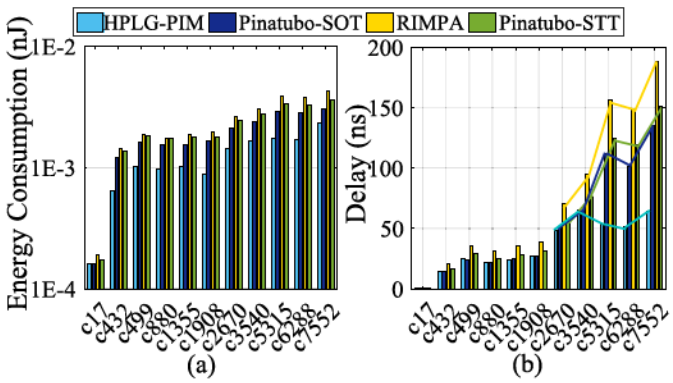


Fig. 10. (a) Energy consumption and (b) delay of ISCAS-85 benchmarks mapped to three different PIM architectures (Y-axis in energy plot: log scale).

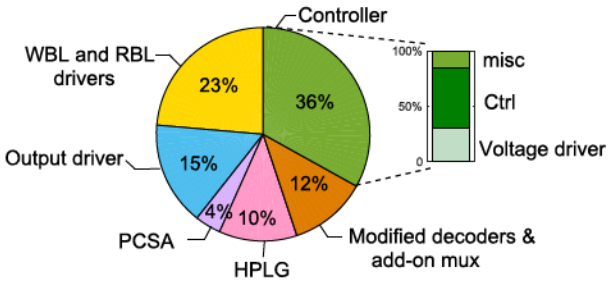


Fig. 11. Area overhead of the *HPLG-PIM* architecture within one Bank.

is used in SPICE to verify the proposed design and acquire the performance. Second, an architectural-level simulator is built based on NVSim [48]. Based on the device-/circuit-level results, our simulator can alter the configuration files (.cfg) corresponding to different array organization and report performance metrics for PIM operations. The controllers and add-on circuits are synthesized by the Design Compiler [50] with an industry library. Third, a behavioral-level simulator is developed in MATLAB, calculating the latency and energy that *HPLG-PIM* spends on different applications. In addition, it has a mapping optimization framework to maximize the performance according to the available resources. Here, to evaluate logic performance in computing and boosted computing modes, a logic netlist in the Berkeley Logic Interchange Format (.blif) is fed into ABC [51] to obtain synthesized logic networks. Meanwhile, parameters such as fan-in restriction are set up during the synthesis. The synthesized networks are then mapped to the proposed PIM architecture using the MATLAB code to assess the performance.

Fig. 10 gives the ISCAS-85 combinational circuit benchmarks implemented using the *HPLG-PIM*, RIMPA [20], and Pinatubo [17]. To have an impartial comparison, Pinatubo, as a general system architecture for NVMs, is implemented with both standard STT-MRAM and identical SOT-MRAM cell as used for our design. As shown, the proposed design offers the lowest energy and delay compared with the counterparts in different benchmarks. We observe that: 1) our proposed design reduces the energy consumption by ~55%, 67%, and 74.4% compared with Pinatubo-SOT, Pinatubo-STT, and RIMPA, respectively. This considerable improvement mainly comes from proposed logic efficiency and reduced-cycle operations resulted from HPLG add-on and 2) it outperforms mentioned PIM architectures with 31.3%, 40%, and 52% reduction in delay on different benchmark circuits. It is worth pointing out that for five more complex benchmark circuits (i.e., c2670, c3540, c5315, c6288, and c7552), as logic complexity increases, *HPLG-PIM* can show much better performance than the rest.

3) Area Overhead Estimation: Our experiments show that *HPLG-PIM* imposes 2.8% area overhead to the original memory die, and Pinatubo [17] and RIMPA [20] incur 0.9% and 17% area overhead, respectively. Fig. 11 shows the breakdown of area overhead resulted from add-on hardware to memory chip. It can be seen that the controller and modified drivers contribute more than 50% of this area overhead in a Bank where the HPLG unit takes up to 10%.

VI. Application: Graph Processing

The *HPLG-PIM*'s parallel operations can be readily used to accelerate a wide variety of real-world applications. Recently, in-memory acceleration of graph processing tasks has attracted much attention. From the graph processing algorithm perspective, network topology analysis can help us better understand the intricate connectivity of complex networks in practical problems [52], [53]. For instance, the matching index is a basic topology parameter that characterizes the similarity between two vertices in a network. It measures the ratio of common neighbors for a pair of vertices. Evaluation of these network properties plays an essential part in potential applications, such as social network analysis, traffic flow control, and so on. For the sake of limited space, we briefly explain one widely used task so-called *matching index*.

The matching index $M_{i,j}$ quantifies the "similarity" between two vertices based on the number of common neighbors shared by vertices as $((\sum \text{common neighbors})/(\sum \text{total number of neighbors}))$. The main task here is to find the common and total number of neighbors, which can be carried out and accelerated by *HPLG-PIM*. Fig. 12 provides a straightforward example to elucidate the mapping and acceleration method of *HPLG-PIM*. Initially, the sample the four-vertex network is converted into adjacency matrix and stored in four consecutive rows of a sub-array. To find the common neighbors of two particular vertices (e.g., V1 and V2), *HPLG-PIM* performs parallel `AND2` on the rows and SA's outputs determine the matches (here, V4). In addition, the total number of neighbors is found by performing the `OR2` operation on the same rows. Then, *HPLG-PIM*'s operations `XOR2` (2-cycle) operation can readily process the summation operation to generate corresponding index matrix.

A. Data Partitioning and Allocation

The real-world graph consists of millions of vertices and edges that need to be processed. To efficiently map such graphs into the *HPLG-PIM* architecture, graph-partitioning methods are needed. Here, we adopt the interval-block partitioning method to balance the workloads of each *HPLG-PIM*'s chip

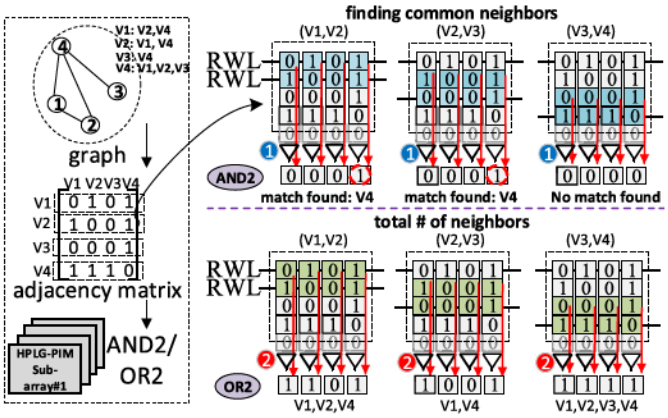


Fig. 12. *HPLG-PIM*'s mapping and acceleration for matching index task.

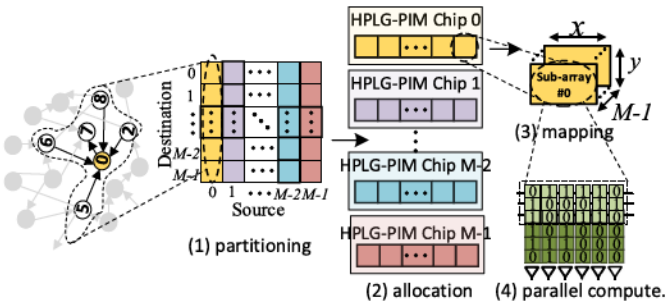


Fig. 13. Graph partitioning and allocation of *HPLG-PIM* accelerator.

TABLE VII
SOCIAL NETWORK DATA SETS

Dataset	Nodes	Edges	Graph Information
ego-Facebook	4,039	88,234	profiles & friends lists from Facebook [55]
dblp-2010	326,186	1,615,400	scientific collaboration network
amazon-2008	735,323	5,158,388	similarity among books reported by Amazon store

and maximize parallelism. We use the hash-based method [54] to split the vertices into M intervals and then divide the edges into M^2 blocks, as shown in Fig. 13. Now each block is allocated a specific chip and accordingly mapped to its sub-arrays. Considering an N-vertex sub-graph that needs to be mapped to a *HPLG-PIM* chip with N_s activated sub-arrays with the size of $x \times y$, each sub-array can process n vertices ($n \leq f | n \in N$, $f = min(x, y)$). Therefore, the number of sub-arrays for processing an N-vertex sub-graph can be formulated as $N_s = \lceil \frac{N}{f} \rceil$.

B. Experiment Setup

To evaluate the performance of the accelerators, we take three social network data sets as tabulated in Table VII. Then, we map and run two graph-processing tasks, i.e., degree centrality and matching index on them that seek most of the *HPLG-PIM*'s operations.

C. Accelerators' Setup

For impartial comparison, we reimplement other accelerator designs under our performance evaluation framework, where

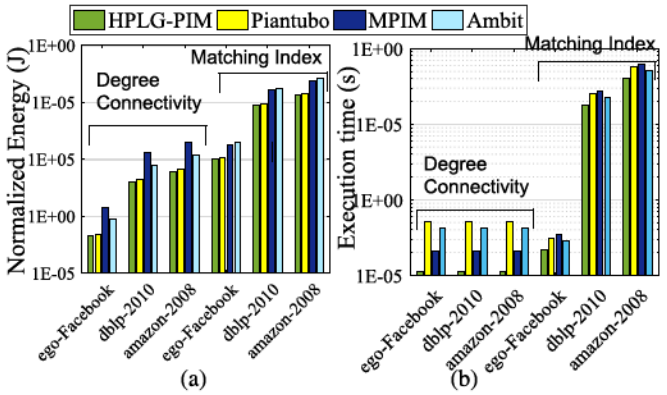


Fig. 14. Normalized log-scaled (a) energy consumption and (b) execution time of different PIM accelerators.

the detailed setup are introduced as follows. ***STT-MRAM:*** We developed a Pinatubo-like [17] accelerator by modifying memory SAs. Pinatubo is implemented by the standard STT-MRAM cell (.cell file in NVSim [48]). ***ReRAM:*** An MPIM-like [18] accelerator with 256 sub-arrays and one buffer sub-array per bank is considered for evaluation. In the computational sub-arrays, for each mat, there are 256 × 256 ReRAM cells. For evaluation, an NVSim simulator [48] was extensively modified to work with the Design Compiler [50] to emulate MPIM functionality. Note that the default NVSim's ReRAM cell file (.cell) was adopted for assessment. ***DRAM:*** We developed an Ambit-like [56] accelerator for graph processing. Ambit implements the logic function using capacitor-based majority functions. We accordingly modified CACTI-3DD [57] for the evaluation of DRAM's solution. The controllers were synthesized in the Design Compiler [50]. ***GPU:*** We used the NVIDIA GTX 1080Ti Pascal GPU. It has 3584 CUDA cores running at 1.5 GHz (11TFLOPs peak performance). The energy consumption was measured with the NVIDIA's system management interface. We scaled the achieved results by 50% to exclude the energy consumed by cooling, and so on.

D. Overall Performance Improvement

Fig. 14(a) shows the normalized energy consumption of the under-test PIM accelerators on two graph processing tasks (i.e., degree connectivity and matching index). As shown, *HPLG-PIM* offers the highest energy efficiency compared with others owing to its low-energy and fully parallel operations. We observe that *HPLG-PIM* achieves on average 4× higher energy efficiency than that of the Ambit accelerator. The main reason here is the energy efficiency of the basic operations in the *HPLG-PIM*; as discussed earlier, *HPLG-PIM* can finish the operations (such as AND2) in one single cycle; however, similar operation in Ambit imposes multi-cycle operations, avoiding destructive data-overwritten. Fig. 14(a) shows that *HPLG-PIM* solution saves 1.7× and 3.6× energy compared with that of Pinatubo and MPIM solutions. It is worth pointing out that Ambit is not capable of implementing parallel XOR in memory required for different tasks and, therefore, impose excessive delay and energy consumption to memory chip. To realize such operation in the Ambit platform, we con-

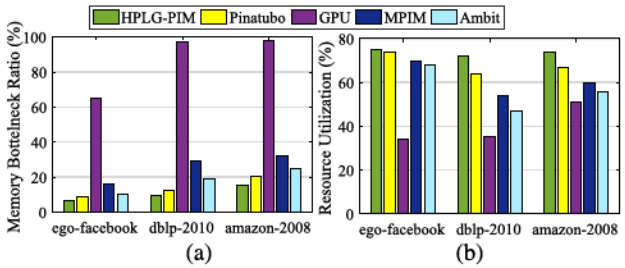


Fig. 15. (a) Memory bottleneck ratio. (b) Resource utilization ratio.

sider multi-cycle MG-based implementation [46]. Meanwhile, the Pinatubo and MPIM uses the multi-cycle XOR logic due to their XOR-friendly architecture. Furthermore, *HPLG-PIM* does not follow the conventional ReRAM-based crossbar designs to realize the PIM, which brings significant energy efficiency due to eliminating the digital-to-analog converter/analog-to-digital converter (DAC/ADC) units. Note that MPIM and Pinatubo-PCM (not implemented here) platforms support multi-row operations (up to 256 rows), which can be useful in specific vector processing tasks; however, *HPLG-PIM* can perform such operation in multi-cycles using the multi-row computation method.

Fig. 14(b) shows the *HPLG-PIM* and other PIMs' execution time on different tasks. It shows that *HPLG-PIM* solution is on average 5.1× faster than the DRAM solution (Ambit) and 2.5× faster than the STT-MRAM solution (Pinatubo). This is mainly because of the ultra-fast and parallel in-memory operations of the *HPLG-PIM* compared with the multi-cycle DRAM/STT-MRAM operations. In addition, we see that *HPLG-PIM* is 5.3× faster than the ReRAM solution. In addition, we compared the energy saving and speed-up of the matching index task running on *HPLG-PIM* and GPU on the amazon-2008 data set. We observe that *HPLG-PIM* can achieve about 19.1× higher energy efficiency and 11.2× speed-up compared with GPU.

E. Memory Bottleneck

Fig. 15(a) depicts the memory bottleneck ratio, i.e., the time fraction at which the computation has to wait for data and on-/off-chip data transfer obstructs its performance (memory wall happens) running matching index task on three data sets. The evaluation is performed according to the peak performance and experimentally extracted results for each platform considering the number of memory access in each data set. We observe that *HPLG-PIM* along with other PIM solutions (except MPIM) spends less than ∼25% time for memory access and data transfer. Due to unbalanced computation and data movement in MPIM and the limitation in the number of activated sub-arrays, it shows higher ratio than other PIMs. However, GPU spends more than 90% time waiting for the loading data. The less memory wall ratio can be interpreted as the higher resource utilization ratio for the accelerators, which is plotted in Fig. 15(b). We observe that *HPLG-PIM* can efficiently use up to 70% of its computation resources. Overall, PIM solutions can demonstrate high ratio (more than 45%), which reconfirms the results reported in Fig. 15(a).

VII. Conclusion

In this article, we initially proposed an HPLG using a new 5-terminal magnetic DWM device. The proposed HPLG is able to perform a full set of 1- and 2-input Boolean logic functions (i.e., NOT, AND/NAND, OR/NOR, and XOR/XNOR) by configuring the applied keys. The experimental results on a set of ISCAS-89, ITC-99, and EPFL benchmarks show that a well-secured HPLG implementation can obtain up to 51.4% and 10% average improvements on the PDP compared with recent non-volatile logic and CMOS-based designs, respectively. We then leveraged this gate to realize a novel PIM architecture for highly flexible and efficient logic computation. The simulation results for widely used graph processing tasks running on three social network data sets indicate roughly 3.6× higher energy efficiency and 5.3× speed-up over recent ReRAM accelerators. In addition, HPLG-PIM achieves ∼4× higher energy efficiency and 5.1× speed-up over recent processing-in-DRAM acceleration methods.

VIII. Acknowledgment

This work was supported in part by the National Science Foundation under Grant 1740126 and Grant 1908495 and in part by Semiconductor Research Corporation nCORE under Grant 1740126 and Grant 1908495.

References

[1] J. T. McDonald, Y. C. Kim, T. R. Andel, M. A. Forbes, and J. McVicar, "Functional polymorphism for intellectual property protection," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2016, pp. 61–66.

[2] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2015, pp. 137–143.

[3] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," in *Proc. 41st Annu. Design Automat. Conf.*, Jun. 2004, pp. 753–760.

[4] S. Patnaik, N. Rangarajan, J. Knechtel, O. Sinanoglu, and S. Rakheja, "Advancing hardware security using polymorphic and stochastic spin-Hall effect devices," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, Mar. 2018, pp. 97–102.

[5] A. Stoica, R. S. Zebulum, X. Guo, D. Keymeulen, M. I. Ferguson, and V. Duong, "Taking evolutionary circuit design from experimentation to implementation: Some useful techniques and a silicon demonstration," *IEE Proc.-Comput. Digit. Techn.*, vol. 151, no. 4, pp. 295–300, Jul. 2004.

[6] Y. Bi *et al.*, "Emerging technology-based design of primitives for hardware security," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 1, p. 3, Dec. 2016.

[7] A. Stoica, R. Zebulum, and D. Keymeulen, "Polymorphic electronics," in *Evolvable Systems: From Biology to Hardware*, 2001, pp. 291–302.

[8] R. Ruzicka, "New polymorphic NAND/XOR gate," in *Proc. 7th WSEAS Int. Conf. Appl. Comput. Sci.*, Venice, Italy, 2007, pp. 192–196.

[9] Y. Bi, P.-E. Gaillardon, X. S. Hu, M. Niemier, J.-S. Yuan, and Y. Jin, "Leveraging emerging technology for hardware security – Case study on silicon nanowire fets and graphene symfets," in *Proc. IEEE 23rd Asian Test Symp.*, Nov. 2014, pp. 342–347.

[10] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. 43rd Int. Symp. Comput. Archit.*, Jun. 2016, pp. 27–39.

[11] V. Seshadri and O. Mutlu, "Simple operations in memory to reduce data movement," in *Advances in Computers*, vol. 106. Amsterdam, The Netherlands: Elsevier, 2017, pp. 107–166.

[12] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Compute caches," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 481–492.

[13] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 2–13, 2009.

[14] X. Fong *et al.*, "Spin-transfer torque devices for logic and memory: Prospects and perspectives," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 1, pp. 1–22, Jan. 2016.

[15] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic RAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 470–483, Mar. 2018.

[16] 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.

[17] 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 Annu. Design Automat. Conf.*, Jun. 2016, p. 173.

[18] M. Imani, Y. Kim, and T. Rosing, "MPIM: Multi-purpose in-memory processing using configurable resistive memory," in *Proc. 22nd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2017, pp. 757–763.

[19] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.

[20] S. Angizi, Z. He, F. Parveen, and D. Fan, "RIMPA: A new reconfigurable dual-mode in-memory processing architecture with spin Hall effect-driven domain wall motion device," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2017, pp. 45–50.

[21] 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 Annu. Design Automat. Conf.*, Jun. 2018, p. 162.

[22] Y. Pan *et al.*, "A multilevel cell STT-MRAM-based computing in-memory accelerator for binary convolutional neural network," *IEEE Trans. Magn.*, vol. 54, no. 11, Nov. 2018, Art. no. 9401305.

[23] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on RRAM," in *Proc. 22nd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Jan. 2017, pp. 782–787.

[24] S. Angizi and D. Fan, "IMC: Energy-efficient in-memory convolver for accelerating binarized deep neural network," in *Proc. Neuromorphic Comput. Symp.*, 2017, p. 3.

[25] F. Parveen, Z. He, S. Angizi, and D. Fan, "Hybrid polymorphic logic gate with 5-terminal magnetic domain wall motion device," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2017, pp. 152–157.

[26] H. Honjo *et al.*, "10 nmf perpendicular-anisotropy CoFeB-MgO magnetic tunnel junction with over 400° C high thermal tolerance by boron diffusion control," in *Proc. Symp. VLSI Technol.*, Jun. 2015, pp. T160–T161.

[27] 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, 2010, Art. no. 217202.

[28] H. Cao, Z. Xu, H. Sang, D. Sheng, and C. Tie, "Template synthesis and magnetic behavior of an array of cobalt nanowires encapsulated in polyaniline nanotubules," *Adv. Mater.*, vol. 13, no. 2, pp. 121–123, 2001.

[29] R. D. McMichael and M. J. Donahue, "Head to head domain wall structures in thin magnetic strips," *IEEE Trans. Magn.*, vol. 33, no. 5, pp. 4167–4169, Sep. 1997.

[30] S. Fukami *et al.*, "20-nm magnetic domain wall motion memory with ultralow-power operation," in *IEDM Tech. Dig.*, Dec. 2013, pp. 3–5.

[31] A. Himeno, T. Okuno, T. Ono, K. Mibu, S. Nasu, and T. Shinjo, "Temperature dependence of depinning fields in submicron magnetic wires with an artificial neck," *J. Magn. Magn. Mater.*, vol. 286, pp. 167–170, Feb. 2005.

[32] X. Fong, S. K. Gupta, N. N. Mojumder, S. H. Choday, C. Augustine, and K. Roy, "KNACK: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque MRAM bit-cells," in *Proc. Int. Conf. Simulation Semiconductor Processes Devices*, Sep. 2011, pp. 51–54.

[33] M. J. Donahue and D. G. Porter. *OOMMF: Object Oriented Micromagnetic Framework*. Accessed: 2016. [Online]. Available: http://math.nist.gov/oommf/

[34] L. Chang, Y. Yao, P. Lin, and S. Lee, "Magnetic interaction in domain wall depinning at square notch and antinotch traps," *IEEE Trans. Magn.*, vol. 47, no. 10, pp. 2519–2521, Oct. 2011.

[35] D. Bromberg, M. Moneck, V. Sokalski, J. Zhu, L. Pileggi, and J.-G. Zhu, "Experimental demonstration of four-terminal magnetic logic device with separate read- and write-paths," in *IEDM Tech. Dig.*, Dec. 2014, pp. 1–33.

[36] (2011). *Ncsu eda Freepdk45*. [Online]. Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[37] B. Wang, Z. Wang, C. Hu, Y. Zhao, Y. Zhang, and W. Zhao, "Radiation-hardening techniques for spin orbit torque-MRAM peripheral circuitry," *IEEE Trans. Magn.*, vol. 54, no. 11, Nov. 2018, Art. no. 3401905.

[38] 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, 2012, Art. no. 122404.

[39] 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*, Aug. 2016, pp. 314–319.

[40] 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.

[41] S. Angizi, Z. He, A. S. Rakin, and D. Fan, "CMP-PIM: An energy-efficient comparator-based processing-in-memory neural network accelerator," in *Proc. 55th Annu. Design Automat. Conf.*, Jun. 2018, p. 105.

[42] H. Noguchi *et al.*, "Novel voltage controlled MRAM (VCM) with fast read/write circuits for ultra large last level cache," in *IEDM Tech. Dig.*, Dec. 2016, pp. 5–27.

[43] Y. Zhang, B. Yan, W. Wu, H. Li, and Y. Chen, "Giant spin Hall effect (GSHE) logic design for low power application," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, 2015, pp. 1000–1005.

[44] S. Matsunaga *et al.*, "MTJ-based nonvolatile logic-in-memory circuit, future prospects and issues," in *Proc. Conf. Design, Automat. Test Eur.*, Apr. 2009, pp. 433–435.

[45] K. Huang, R. Zhao, and Y. Lian, "STT-MRAM based low power synchronous non-volatile logic with timing demultiplexing," in *Proc. IEEE/ACM Int. Symp. Nanoscale Archit.*, Jul. 2014, pp. 31–36.

[46] 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.*, vol. 37, no. 9, pp. 1788–1801, Sep. 2018.

[47] K. Huang and R. Zhao, "Magnetic domain-wall racetrack memory-based nonvolatile logic for low-power computing and fast run-time-reconfiguration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 9, pp. 2861–2872, Sep. 2016.

[48] 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.

[49] S. Angizi, Z. He, F. Parveen, and D. Fan, "IMCE: Energy-efficient bit-wise in-memory convolution engine for deep neural network," in *Proc. 23rd Asia South Pacific Design Automat. Conf.*, Jan. 2018, pp. 111–116.

[50] *Synopsys Design Compiler. Product Version 14.9.2014*, Synopsys, Mountain View, CA, USA.

[51] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proc. Int. Conf. Comput. Aided Verification*. Berlin, Germany: Springer, 2010, pp. 24–40.

[52] S. Angizi, J. Sun, W. Zhang, and D. Fan, "GraphS: A graph processing accelerator leveraging SOT-MRAM," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, Mar. 2019, pp. 378–383.

[53] S. Angizi and D. Fan, "GraphiDe: A graph processing accelerator leveraging in-DRAM-computing," in *Proc. Great Lakes Symp. VLSI*, May 2019, pp. 45–50.

[54] G. Dai *et al.*, "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.

[55] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 539–547.

[56] V. Seshadri *et al.*, "Ambit: In-memory accelerator for bulk bit-wise operations using commodity DRAM technology," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2017, pp. 273–287.

[57] K. Chen *et al.*, "CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory," in *Proc. Conf. Design, Automat. Test Eur.*, Mar. 2012, pp. 33–38.

Shaahin Angizi (S'15) received the B.Sc. degree in computer engineering, hardware from South Tehran Branch, Islamic Azad University (IAU), Tehran, Iran, in 2012, and the M.Sc. degree in computer engineering, computer systems architecture from 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 research interests include in-memory computing, deep learning, low-power VLSI designs, and spin-based computing.

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 Arizona State University, Tempe, AZ, USA.

His research interests include machine learning, neuromorphic computing, emerging memory technology, and analog/mixed signal circuit design.

An Chen (SM'11) received the Ph.D. degree in electrical engineering from Yale University, New Haven, CT, USA, in 2004.

He is on assignment from IBM Corporation, San Jose, CA, USA, to serve as the Executive Director of the Nanoelectronics Research Initiative (NRI), Durham, NC, USA, and is based at Almaden Research Laboratories, San Jose. The NRI supports university-based research on future nanoscale logic devices to replace the CMOS transistor in the 2020 timeframe. He started working on emerging memory technologies at Spansion LLC, Sunnyvale, CA, USA. In 2007, he joined Advanced Micro Devices (AMD), Santa Clara, CA, USA, as a Full-Time Assignee to the Nanoelectronics Research Initiative (NRI) program with SRC. He continued working on beyond-CMOS devices with the NRI and STARnet Programs at GLOBALFOUNDRIES, Santa Clara, CA, USA, which separated from AMD in 2009. He is also a memory Tech Lead responsible for research collaborations with industry consortia and partners on emerging memories.

Dr. Chen has been the Chair of the Emerging Research Device (ERD) Group of the International Technology Roadmap for Semiconductors (ITRS), since 2011.

Deliang Fan (M'15) received the M.S. and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2012 and 2015, respectively, under the supervision of Prof. K. Roy.

He is currently an Assistant Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University (ASU), Tempe, AZ, USA. Before joining ASU in 2019, he was an Assistant Professor with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, USA. He has authored and coauthored over 90 peer-reviewed international journal articles/conference papers. His primary research interests include energy efficient and high performance big data processing-in-memory circuit, architecture and algorithm, with applications in deep neural network, data encryption, graph processing and bioinformatics acceleration-in-memory system; brain-inspired (Neuromorphic) computing, hardware-aware deep learning optimization and AI security.

He served as the Technical Program Committee Member of DAC, ICCAD, GLSVLSI, ISVLSI, ASP-DAC, and ISQED. He is the receipt of the Best Paper Award of GLSVLSI 2019, ISVLSI 2018, and ISVLSI 2017. He also served as a Technical Reviewer for over 30 international journals /conferences, such as nature electronics, IEEE TNNLS, TVLSI, TCAD, TNANO, TC, TCAS, ISCAS, and ISLPED. He is also the Technical Area Chair of GLSVLSI 2019/2020, ISQED 2019/2020, and the Financial Chair of ISVLSI 2019.