Digital Object Identifier

Domain-Specific STT-MRAM-based In-Memory Computing: A Survey

Alaba Yusuf [1], Tosiron Adegbija [2], (Senior Member, IEEE), and Dhruv Gajaria [34], (Member, IEEE)
[1]Department of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona, USA (email: alabayusuf@arizona.edu)
[2]Department of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona, USA (email: tosiron@arizona.edu)
[3]Pacific Northwest National Lab
Richland, WA, USA (email: dhruv.gajaria@pnnl.gov)
[4]Work done while at the University of Arizona.

Corresponding author: Alaba Yusuf (email: alabayusuf@arizona.edu).

This work was supported in part by the National Science Foundation (NSF) under grant CNS-1844952. Any opinions, findings, conclusions, or recommendations expressed are those of the authors and do not necessarily reflect the views of the NSF.

ABSTRACT In recent years, the rapid growth of big data and the increasing demand for high-performance computing have fueled the development of novel computing architectures. Among these, in-memory computing architectures that leverage the high-density and low-latency nature of modern memory technologies have emerged as promising solutions for domain-specific computing applications. STT-MRAM (Spin Transfer Torque Magnetic Random Access Memory) is one of such memory technology that holds great potential for in-memory computing due to numerous advantages such as non-volatility, high density, high endurance, and low power consumption. This survey paper aims to provide a comprehensive overview of the state-of-the-art in STT-MRAM-based domain-specific in-memory computing (DS-IMC) architectures. We examine the challenges, opportunities, and trade-offs associated with these architectures from the perspective of various application domains, like machine learning, image and signal processing, and data encryption. We explore different experimental research tools used in studying these architectures, guidelines for efficiently designing them, and gaps in the state-of-the-art that necessitate future research and development.

INDEX TERMS domain-specific architectures, in-memory computing, spin-transfer torque magnetic RAM.

I. INTRODUCTION AND MOTIVATION

Most modern computer systems are based on the von Neumann architecture, where memory units are separated from the processing units. During program execution, data must be transferred back and forth between the processing and memory units, leading to significant costs in latency and energy. The latency associated with accessing data from the memory units is the critical performance bottleneck for a broad range of applications, given the "memory wall" [1]—the significant disparity between the speed of the memory and processing units. In addition, the energy cost of moving data is another significant challenge, given the fact that the computing systems are power-limited. Current approaches such as graphics processing units (GPUs) [2] or application-specific processors [3], [4] are unlikely to overcome the problem of data movement because they are designed to perform specific tasks or functions and are not intended to handle the complexity of managing data movement between different components. Therefore, novel architectures must be explored to address this perennial problem in modern computer systems.

One promising and increasingly popular approach to mitigating the overheads of data transfer is to physically place computing units closer to the memory. This concept is known as near-memory computing (NMC) and has benefited significantly from the recent commercialization of advanced memory modules such as hybrid memory cube (HMC) [5], advances in die stacking technology, and high bandwidth memory (HBM) [6]. However, a physical separation between the memory and processing units in NMC architectures still exists. In-memory computing (IMC) [7], [8], illustrated in Fig. 1, enhances NMC by co-locating the memory and processing elements, such that computations are performed within the memory. This is achieved by exploiting the attributes of the memory devices, like their array-level organization, peripheral circuitry, and control logic.

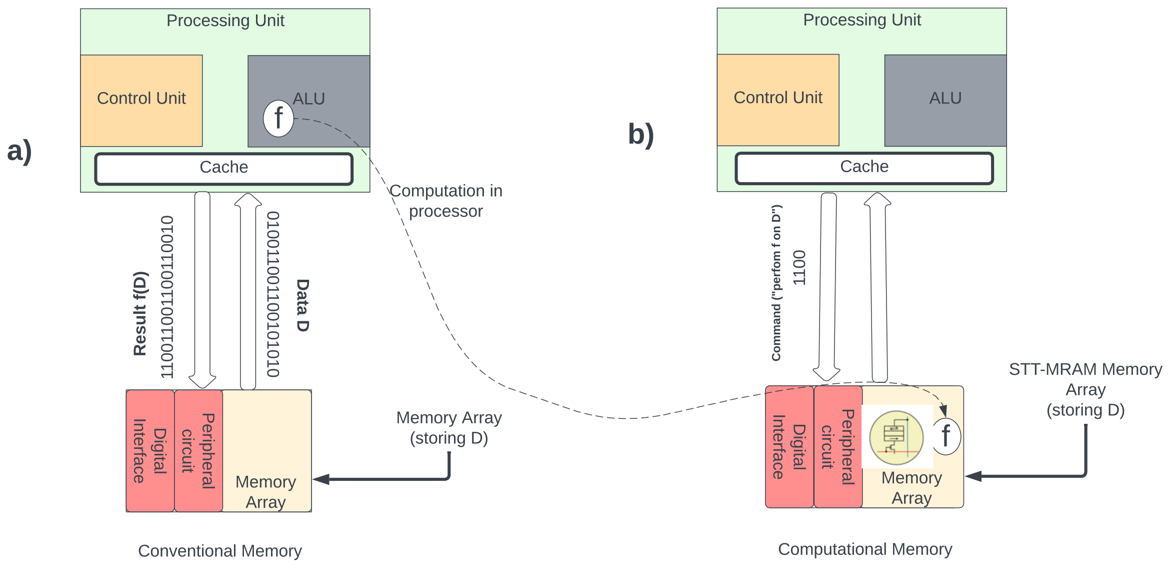

FIGURE 1: **Computing architectures a) Typical Von Neumann architecture b) in-memory computing architecture**

In general, IMC architectures can be designed as application-specific integrated circuits (ASICs), general-purpose architectures, or domain-specific architectures. ASICs are highly optimized for specific computing tasks and offer the highest performance, but they are limited in their versatility and can be expensive to develop and manufacture. Alternatively, general-purpose designs feature generalized arithmetic-logic units that are more versatile but may not provide the level of performance required for some applications. This paper is motivated by the benefits of domain-specific design over ASIC or general-purpose designs [9], [10]. Domain-specific designs focus on providing optimal performance for specific domains (i.e., a group of applications with similar characteristics and computational needs) and enable a compromise between ASICs and general-purpose designs for addressing the memory and power walls inherent in modern-day computing architectures.

A. OVERVIEW OF EMERGING MEMORY TECHNOLOGIES

Today's computing systems heavily rely on the memory subsystem, which plays a crucial role in the performance and energy of computer systems. In most traditional computers, the memory hierarchy includes caches implemented using static random-access memory (SRAM) and main memory implemented with dynamic random-access memory (DRAM). These memory technologies operate on a charge storage mechanism, meaning they store data by manipulating the electrical charge state of their storage elements. However, challenges arise as these charge-based memories encounter difficulties in scaling down to the 10-nm node and beyond. The nanoscale susceptibility to charge loss poses challenges in performance, reliability, and noise margin. To address this, non-charge-based emerging memory technologies, most commonly, *non-volatile memories (NVMs)*, are actively researched to revolutionize the memory hierarchy [11]. The ideal memory device features include fast read/write speed (<ns), low operation voltage (<1 V), low energy consumption (fJ/b for read/write), long data retention time (>10 years), long read/write cycling endurance (>10 17 cycles), and excellent scalability (<10 nm). While achieving all these ideals in a single universal memory device is nearly impossible, emerging resistance-based NVM technologies like pin-transfer torque magnetic Random Access Memory (STT-MRAM) , phase-change RAM (PCRAM), and resistive RAM (ReRAM), among others, are pursued to fulfill some of these characteristics [12], [13], [14]. These NVMs share common features as nonvolatile two-terminal devices, distinguishing states through high-resistance state (HRS) and low-resistance state (LRS) with differing switching mechanisms. STT-MRAM uses ferromagnetic layers, and relies on the manipulation of electron spins to switch between these resistance states, while PCRAM relies on phase change in materials to achieve the same objective, and ReRAM changes resistance by altering the conductivity of a material, typically metal oxides, through the migration of defects. These NVM technologies offer diverse solutions for nonvolatile data storage with various advantages and characteristics. For instance, STT-MRAM has smaller cell area compared to SRAM, maintaining low programming voltage, fast read/write speed, and long endurance, making it an attractive replacement for embedded memories in the last level-cache and main memories [15], [16], [17]. PCRAM and ReRAM offer low programming voltage and fast write/read speed, making them attractive as a replacement for existing technologies in resource-constrained memory systems.

B. SUITABILITY OF STT-MRAM FOR IMC

To fully exploit the energy, performance, and area benefits of domain-specific designs, the IMC architecture must be designed using technologies uniquely suited to domain-specific IMC designs. STT-MRAM —the focus of this paper—is one of the most promising emerging technologies for domain-specific IMC architectures due to its unique combination of properties, including non-volatility, high endurance, high density, and extremely low leakage. In addition, STT-MRAM's features include energy-efficient operation with non-constant refresh cycles, low read and write latencies compared to other NVMs, low power consumption [18], compact cell size that allows for denser memory placement near computational units, robust integration capabilities facilitating the co-location of memory and processing units on a single chip, seamless compatibility with CMOS logic for smooth interaction between memory and computation elements, support for parallelism and pipelining, and the capacity to modify individual bits for efficient data storage, manipulation, and processing within IMC scenarios [19], [20], [21].

As a result, a rapidly growing body of research focuses on exploring designs for STT-MRAM-based IMC. There have been a few prior works that have surveyed the state-of-the-art in STT-MRAM-based IMC, some of which emphasize STT-MRAM computing from the perspectives of the bit-cell, circuit, and system levels [22], [23], [24]. However, our survey approaches STT-MRAM-based IMC from the unique perspective of domain-specific designs. Since much prior work has shown the clear benefits of domain-specific designs over ASICs or general-purpose designs for emerging computer architectures [25], [26], [27], [9], we make a strong case for more research emphasis on "STT-MRAM-based domain-specific IMC (DS-IMC)". Given that the development of STT-MRAM-based IMC architectures is still in its early stages, and further research is required to fully realize the potential of this technology, this survey aims to explore in detail the potentials of STT-MRAM-based DS-IMC. We will examine the challenges, opportunities, and trade-offs associated with these architectures and discuss their potential for future research and development. The analysis covers aspects such as STT-MRAM cell design, type of MTJ, and parameters like CMOS size and state-of-the-art simulation tools used in the examined works. The survey categorizes articles based on real-world scenarios, including applications in Machine Learning (BNN, CNN, SNN), Image Processing, and Data Encryption. Additionally, driven by insights from the surveyed works, we enumerate some design guidelines for STT-MRAM-based DS-IMC and discuss challenges and opportunities associated with each design to offer a comprehensive overview of the current landscape in this research domain.

In summary, this survey makes the following main contributions:

- We review recent works proposed for implementing STT-MRAM IMC from the perspective of domain-specific architectures. Although complementary to other surveys that focus on circuit-level [22], [23], [24] and architectural-level [28], [29] designs, this is the first survey that explores the tradeoffs involved with domain-specific computing using STT-MRAM-based IMC. The survey is performed from the perspective of three important domains: machine learning, image and signal processing, and data encryption, and we draw important insights that can be applied to other application domains.
- To enable practical research on STT-MRAM-based DS-IMC, we explore different simulators and tools that can be used to implement STT-MRAM IMC for a variety of domains.
- We discuss some design guidelines, challenges, and gaps associated with STT-MRAM-based DS-IMC. There is a strong need to understand the factors that currently limit the design and deployment of DS-IMC using STT-MRAM. We provide some suggestions on research directions for addressing these gaps.

II. BACKGROUND

STT-MRAM is a non-volatile memory (NVM) technology that outperforms other technologies such as ferroelectric field effect transistor (Fe-FET), PCM, and RRAM, particularly in terms of read time and energy efficiency [28]. In addition, STT-MRAM, more so than many other emerging NVM technologies, has demonstrated commercial viability [30], making it one of the best alternatives for IMC studies. Choe et al. [31] emphasize the dynamic research and development of STT-MRAM technology by major industry players, including Everspin Technologies, GlobalFoundaries, Avalanche Technologies, Sony, Micron, IMEC, CEA-LETI, Applied Materials, Samsung, Fujitsu, IBM, TSMC, and Spin Transfer Technologies (STT). Notably, Everspin introduced a 3rd generation standalone 256Mb STT-MRAM (pMTJ) and a 1Gb STT-MRAM. Collaborative efforts by Samsung, Sony, and Avalanche resulted in the development of 28nm eSTT-MRAM (pMTJ) and eSTT-MRAM (pMTJ) with a 40nm node. Everspin offers a range of MRAM products, including Toggle-mode MRAM (1st generation) and STT-MRAM (2nd to 4th generation), with variations in MTJ structures.

Despite the appealing features of STT-MRAM, its adoption faces challenges. Chi et al. [18] point out that the write operation is generally slower and more energy-consuming than the read operation. As technology advances, the decreasing write current, coupled with the limited scalability of read current, leads to increased read disturbance errors [32]. Additionally, overstressing the MTJ with write voltage may hinder achieving the required endurance for on-chip caches, as highlighted by Yu et al. [33].

In this section, we provide a summary background on STT-MRAM and discuss its retention time, an important configurable design parameter in STT-MRAM with significant implications for its efficiency. Thereafter, we summarize some of the main challenges associated with using STT-MRAM in practice and explore some methods for addressing those challenges. In addition, we also provide a brief overview of

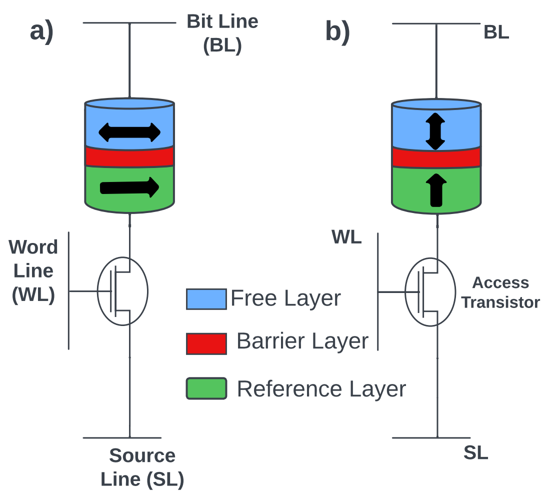

FIGURE 2: **Types of MTJs used in STT-MRAM IMC design a) In-Plane MTJ design. b) Interface-Perpendicular MTJ design.**

domain-specific architectures (DSA) and the components that make up a DSA.

A. BACKGROUND ON STT-MRAM

1) Overview of STT-MRAM

a: Basics of STT-MRAM

STT-MRAM is comprised of a magnetic tunnel junction (MTJ) cell, which is used to store data, and an NMOS transistor, as shown in Figure 2. A typical MTJ cell consists of two ferromagnetic layers separated by an oxide layer. The magnetic orientation of one layer, the free layer, can be freely rotated, while the magnetization of the other layer, the reference layer, is fixed. Thus, the magnetization of the free layer can be parallel or anti-parallel to the reference layer. As a result, the electric resistance of the MTJ cell changes to high for anti-parallel and low for parallel magnetization. These two states represent bits "0" and "1", respectively. The difference in the electric resistance values denoted as R_{AP} and R_{AP}, respectively, is what is known as "tunnel magneto-resistance" (TMR) ratio. TMR refers to the ratio of the difference in resistance between the low-resistance state (parallel magnetization) and its high-resistance state (anti-parallel magnetization) to the resistance in the parallel state. Increasing the TMR ratio increases the separation between states and improves the reliability of the cell [34]. This ratio is crucial for the reliable reading of data stored in the memory cell. The TMR ratio is given by the following equation [35]:

$$TMR = \frac{R_{AP} - R_P}{R_P} \tag{1}$$

Data stored in an STT-MRAM cell is read when a read current (I_r) flows through the MTJ cell to sense its resistance state. Likewise, data is written into an STT-MRAM cell when a write current (I_w) is much higher than the read and the critical current (I_c), which is the minimum current required to switch the magnetization of the MTJ for a given write pulse. A high write current or pulse results in high dynamic energy. This issue is further exacerbated due to the stochastic nature of the writing (switching) process as well as the high sensitivity to process variation, thereby leading to large timing margins [36], as described in Section II-A2d. The key parameter of the MTJ is the Thermal Stability Factor Δ, which specifies the stability of the magnetic orientation of the free layer against thermal noise [37], [38], [39]

Lots of research works address some of the issues discussed above as they relate to MTJs. For instance, Amirany et al. [40] develop a hardware model for a stochastic neuron utilizing the stochastic behavior of MTJs in the subcritical current regime. The model consists of four main components: Pre-Charge Sense Amplifier (PCSA), Fixed MTJs, reconfigurable MTJ, and write and control circuits. The PCSA senses resistance between two MTJs during pre-charge and evaluation phases, facilitating MTJ reconfiguration. Fixed MTJs serve as reference resistors, and reconfigurable MTJ introduces stochasticity. Write and control circuits manage the reconfigurable MTJ through deterministic restoring and stochastic switching. Image binary simulation results with over 10,000 images demonstrate approximately 0.25% PSNR and 0.02% SSIM variation compared to the software counterpart.

While performing sensing, a bit-line may exhibit varying current levels depending on the stored bit. Likewise, sensing the currents across multiple word-lines will yield distinct output current values. Three potential current values can be generated, contingent upon whether the bits are "1" or "0." These current values encompass I_{0-0}, $I_{1-0/0-1}$, and I_{1-1}. These reference currents are employable for direct computation of AND/NAND or NOR/OR operations.

Figure 3 illustrates a typical STT-MRAM in-memory computational architecture. This configuration encompasses several key components, including a row decoder, word line (WL) drivers, memory array, peripheral circuits comprising column decoders, sense amplifiers, and a digital interface.

The role of the row decoder is to select and activate a specific row of memory cells based on the provided address, enabling efficient data access. Meanwhile, the word line is responsible for connecting to the memory cells within a row, facilitating both data reading from and writing to these cells.

Furthermore, the column decoder plays a vital role by selecting and activating a particular column of memory cells for read or write operations. To enhance data accuracy during read processes, the sense amplifier amplifies and detects the subtle voltage differences that represent data within the memory cells.

The peripheral circuitry's function is to manage various aspects of data input and output, addressing and controlling functions, thus streamlining the data storage and retrieval process. Lastly, the digital interface plays a crucial role in facilitating communication between the memory and external devices, enabling seamless data transfer and control command execution.

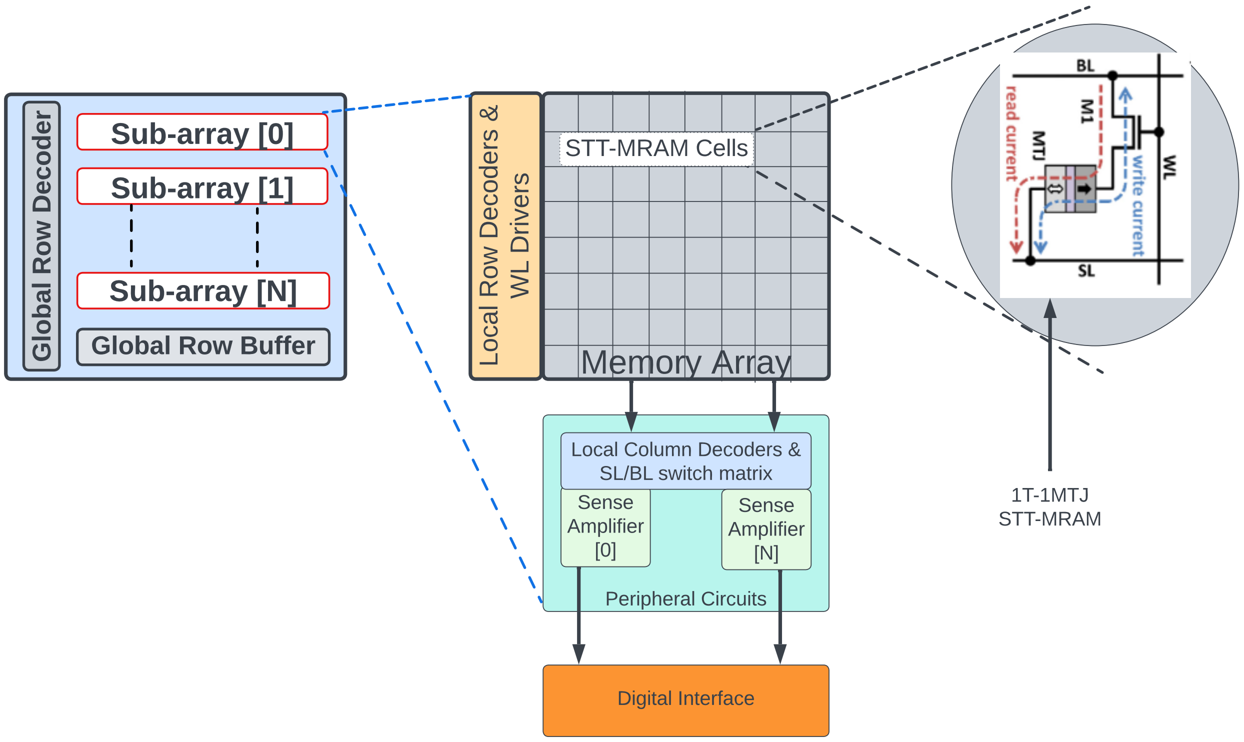

FIGURE 3: **Illustration of STT-MRAM in-memory computing architecture hierarchy. An STT-MRAM bank consists of several subarrays, a global new decoder, and a global row decoder. Each subarray contains of STT-MRAM cells, local row decoder, WL drivers, peripheral circuits (consisting of local column decoder, SL/BL switch matrix, SAs, etc.)**

*b: **Retention Time in STT-MRAM***

In general, the retention time in non-volatile memories refers to the data-retaining capability of their bit cell regardless of the presence of a power supply.

The motivation for reducing retention time in the STT-MRAM cache is to strike a balance between power consumption and performance. By decreasing how long data is stored before naturally fading, designers can optimize the cache for lower energy use and faster access. This approach is beneficial in scenarios where energy efficiency is crucial, like mobile devices. While it may risk data loss in some cases, it enables more aggressive power-saving strategies and better aligns with specific power-performance trade-offs.

In STT-MRAM, this retention time depends on the MTJ cell's thermal stability factor. Increasing the thermal stability factor correlates with extended data retention within the bit-cell. For example, with a Δ value of 60, it can retain the bit-cell content for 10 years [37], [38], [39], [41], [42], [43]. This Δ value can be modeled using:

$$\Delta = \frac{V.H_k.M_s}{2.K_B.T} \tag{2}$$

where V is the volume of the free layer, M_s is the saturation magnetization, K_B is the Boltzmann constant, T is the temperature in kelvin and H_k is the effective field anisotropy. The Δ can be adjusted either by changing the MTJ cell size during fabrication or by adjusting the M_s and H_k values at the material level during the stack development.

An MTJ cell with a high Δ value requires high switching latency and energy. This is because the height of the thermal barrier is greater for a high Δ value, requiring more current for a longer duration to perform the switching. Its relation with I_c can be modeled using the following equation [44]:

$$I_c = \frac{4.e.K_B.T}{h}.\frac{\alpha}{\eta}.\Delta.(1+\frac{4.\pi.M_{eff}}{2.H_k}) \tag{3}$$

where h is Planks constant, α is the Landau-Lifshitz- Gilbert (LLG) damping constant, which plays an important role in the spin dynamics of ferromagnetic systems [45], η is the STT-MRAM efficiency parameter and 4π. M_{eff} is the effective demagnetization field.

2) Challenges with STT-MRAM

With a low Δ value, the STT-MRAM write latency and energy can be significantly reduced. However, reducing the Δ value increases the retention failure rates and the possibility of read disturbance and write errors. These are challenges with STT-

MRAM design that might have significant implications for the design of STT-MRAM-based IMC architectures.

a: Retention Failure

The retention failures in STT-MRAM occur due to the inherent thermal fluctuation of the MTJ cell, which can lead to a change in its magnetic orientation. This can occur regardless of whether or not memory access is performed. The retention failure probability (PRF) for a given time period (t) can be calculated according to [37], [38], [39]:

$$P_{RF} = 1 - exp(\frac{-t}{\tau.exp(\Delta)}) \tag{4}$$

where τ is a constant equal to 1ns. According to the above equation, a relatively high Δ can significantly improve reliability.

b: Read Decision Failure

During a read operation in STT-MRAM, the stored bit value is determined by comparing the reference current with the bit-cell current. Therefore, there must be a reasonable gap between high- and low-resistance state currents. The read decision failure occurs when the read circuitry cannot determine the logic value of the bit-cell. This type of failure is often mitigated by keeping a higher margin between currents of two resistive states [46]. This is known as the TMR ratio, as described in section II-A1a

c: Read Disturbance

Since both read and write currents share the same path in the STT-MRAM bit-cell, the read current can accidentally switch the bit-cell content during the read operation time. The probability of read disturbance (P_{RD}) is strongly dependent on Δ according to [38], [39]:

$$P_{RD} = 1 - exp(\frac{-tr}{\tau.exp(\Delta).(1 - \frac{I_r}{I_c})}) \tag{5}$$

In this equation, Δ has an exponential influence on the probability of the read disturbance, such that a small reduction in Δ, for example, significantly increases the read disturb probability.

d: Write Error

The write operation in STT-MRAM typically occurs due to the stochastic switching nature of the bit-cells due to the inherent randomness associated with the STT phenomenon. In other words, during a write operation in STT-MRAM, the final state or value written to a memory cell is not entirely deterministic or fixed. Instead, it can vary to some extent due to various factors, including inherent physical properties, environmental conditions, and manufacturing variations. This stochastic nature of write operations implies that even when the same data is written to a memory cell multiple times, the exact outcome may not be consistent, leading to variations or uncertainty in the stored data. To address this, error correction and mitigation techniques are often employed to ensure data reliability despite the stochastic behavior. One of the techniques involves using a smaller Δ value, which can accelerate the switching of the MTJ cell due to the reduction in I_c value. Consequently, the required switch time of the MTJ reduces for the same target Write Error Rate (WER) [47], [48].

B. BACKGROUND ON DOMAIN-SPECIFIC ARCHITECTURES

1) Overview Of Domain-Specific Architectures

Moore's Law has been the main driving force behind innovations in computer architecture over the last 5 decades [49]. Today, it is generally believed that Moore's Law has primarily ended (or is finishing) [50], [51], and designers must explore alternative architectures with low overhead, such as Domain-Specific Architectures (DSAs). DSAs represent an emerging paradigm of architectures that optimize data flow for applications in a target domain through hardware acceleration while providing programming flexibility [52], [53]. Examples of recent domains for which DSAs can be highly beneficial include machine learning, artificial intelligence, signal/image processing, data encryption, etc. As such, much research effort is being placed on designing DSAs for these application domains.

In general, DSAs encompass any computing architecture that provides the following:

- Specialized processors to boost energy efficiency by handling common domain-specific tasks in dedicated hardware. For example, a hardware module designed for Fast Fourier Transform (FFT) speeds up these operations, while systolic matrix multiplication processors enhance AI and machine learning tasks. Google's Tensor Processing Unit (TPU) is a prime example, optimized for machine learning training and inference tasks, incorporating hardware, system configurations, and software frameworks to boost speed [54].
- Flexibility to enable adaptability to a wide range of applications, from machine learning to other tasks, although possibly less efficiently. For instance, DSAs designed for machine learning should handle various computations in models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). In addition, they should be capable of executing other neural network inference operations that cannot be easily implemented using specialized hardware. An example is a flexible and efficient deep learning processor (iFPNA) that contains a controller for data arrangement and 16 neuron slices for deep neural network computing operations such as multiply-and-accumulate (MAC), non-linear activation, element-wise operations, etc. [55].
- Heterogeneous processing elements to cater to contrasting application requirements such as low power, high performance, energy efficiency, and programmability.

2) Hardware Components of Domain-Specific Architectures

Hardware components of Domain-Specific Architectures include the following: fixed-function accelerators, specialized processors and domain-specific accelerators, general-purpose processors, and on-chip interconnects.

- Fixed Function Accelerators, specialized for specific tasks, maximize energy efficiency when targeting specialized kernels [56]. While excelling in performance and energy, they lose efficiency when generalized for multiple computations. For instance, in autonomous driving [57], they significantly accelerate deep neural network inference and image feature extraction, resulting in a 93x latency improvement. In image processing [58], they achieve up to a 133x speedup in operations like matrix computations, feature detection, and tracking.
- Domain-Specific Accelerators (DS-Acc) perform a group of related functions, enabling extensive design reuse. Examples include Darwin for bioinformatics [59] and Spatial [60]. DS-Accs are customized by incorporating domain knowledge to align hardware designs with domain kernels, offering tailored processing for specific applications [61], [53].
- General-purpose processors, used within Domain-Specific Architectures (DSAs), offer versatility beyond specific domain applications and specialized accelerators [62]. They possess diverse capabilities, including arithmetic logic units (ALUs) for math and logic operations, data-holding registers, control units for execution, clocks for synchronization, and cache memory for performance. They also handle Input/Output (I/O) with external devices. Some processors feature extras like virtual memory and security functions. These attributes make General-Purpose Processors adaptable for various computing needs.
- On-Chip Interconnect: One significant challenge with DSAs is that DSAs experience significant data movement between different processing elements (PEs). This data movement can account for up to 40% of the total execution time [63]. PEs are fundamental components in computing systems found in specialized accelerators responsible for executing instructions, performing calculations and carrying out tasks such as arithmetic and logic operations, data movement, and control flow operation. Therefore, it is extremely important that the DSA features efficient on-chip communication hardware, such that data is moved in a highly energy-efficient manner [64], [65].

III. STT-MRAM IMC HIERARCHY LEVEL

While much prior IMC research emphasizes memory as an accelerator with kernels running on in-memory processing units, real-world applications often require combining the processor with in-memory computing for efficient execution. This introduces extra data movement across the cache hierarchy and potential issues with address translation exceeding translation lookaside buffer (TLB) capacity. Previous solutions suggested using compute caches or Processing-in-Cache (PiC) with SRAM as caches. However, SRAM caches are power- and space-intensive, and adding processing units can be impractical, particularly in resource-limited systems [66].

Gajaria et al.'s study [67] investigates the trade-offs associated with STT-MRAM IMC across different levels of the memory hierarchy, including main memory and cache hierarchy. They specifically focus on PiC (Processing in Cache) using STT-MRAM caches with relaxed retention characteristics and PiM (Processing in Memory) for non-volatile STT-MRAM main memory. Their research encompasses eight distinct workloads categorized into three groups: CPU-dependent, CPU-dependent with high data reuse, and CPU-dependent with low data reuse.

The PiC architecture involves cache blocks, each comprising a retention counter, a tag, and data storage. These caches are organized into matrices and subarrays, with each subarray housing sense amplifiers for word lines (WLs) and computational logic circuits.

The takeaway points from this study and results are summarized below:

- STT-MRAM offers an excellent opportunity for energy- and area-efficient PiC while providing latency benefits similar to those of SRAM
- The study shows that the executing workloads' characteristics impact the choice of PiC or PiM. In their study, STT-MRAM PiC outperforms PiM latency optimization in CPU-dependent workloads with low instruction-level parallelism (ILP).
- The study shows that the STT-MRAM-based PiC offers much promise and warrants additional studies for effective implementation in emerging resource-constrained systems.

IV. DOMAIN-SPECIFIC IN-MEMORY COMPUTING USING STT-MRAM

In this section, we review some recently proposed implementations of STT-MRAM in-memory computing from the perspective of domain-specific architectures. Table 1 summarizes some of the cell architectures used in the surveyed articles, along with some representative references. To make the survey tractable, we focus on three domains where most of the existing work has been done. The domains are: machine learning (ML) (including different kinds of models), image or signal processing, and data encryption. Despite the limited range of domains in which STT-MRAM in-memory computing has been implemented so far, our survey shows that insights from these domains can be leveraged in a wider variety of domains.

A. MACHINE LEARNING

Machine learning [83] is increasingly popular for analyzing vast datasets. High-performance computing is needed to handle the complex math behind ML. IMC architectures are favored for ML due to their parallelism and data proximity. They reduce data transfers between memory and processor, boosting ML performance and efficiency. This benefits various ML areas like Binary Convolutional Networks (BCNN),

TABLE 1: Summary of STT-RAM Cell Design Architectures

Domain	Applications	References	STT-RAM Type	MTJ Type	MTJ Diameter	TMR	Logic Operation
Machine Learning	BNN	Yu Pan et al. [68]	1T-2M (1 small, 1 large MTJ)	Multi-level Cell (MLC)	60 nm (large MTJ) 40 nm (small MTJ)	170%	Modified Sensing Circuit
		Resh et al. [69]	3T-1M (memory/logic) (1T-1M) x 2 (a more efficient alternative to 3T-1M)	Interface Perpendicular	Modern P (45 nm) Future P (10 nm)	Modern (133%) Future (500%)	Bit-line sensing Inside memory array (no use of SA or external logic)
		Cai et al. [70]	4T-2M	Interface Perpendicular	40 nm	150%	CSA =>XNOR operation VSA =>XAC operation
	CNN	Cai et al. [71]	(1T-1M) x 3	Interface Perpendicular	40 nm	50% - 150%	Current Sense Amp
		Angizi et al. [72]	(1T-1M) x 3	Interface Perpendicular	65 nm	171.20%	Reconfigurable Sensing Amplifier (RSA)
		Kim et al. [73]	1T-1M	Interface Perpendicular	-	-	Bit-line sensing
		Kim et al. [74]	1T-1M	Interface Perpendicular	-	-	Bit-line sensing
	SNN	Cilasun et al. [75]	1T-1M (STT-CRAM) 2T-1M (SHE-CRAM)	-	-	-	Bit-line sensing
		Kang et al. [76]	1T-1M	-	-	150%	Pre-charge Before Summing Amplifier
		Nguyen et al. [77]	2T-2M	In-Plane	60nm x 60nm	100%	Bit-line sensing
		Agrawa et al. [78]	1T-1M	-	-	-	Voltage Sense Amp
Image Processing	Image Processing	Angizi et al. [79]	1T-1M	In-Plane	-	-	Voltage Sense Amp
		He et al. [80]	1T-1M	In-Plane	(65x 65 x 2)nm	171.20%	Modified Sense Amp
Data Encryption	Encryption	Parveen et al. [81]	1T-1M	Interface Perpendicular	(65x65x2)nm	171.20%	SA & 5T DWM
		Angizi et al. [72]	(1T-1M) x 3	Interface Perpendicular	(65x65x2)nm	171.20%	Reconfigurable sensing amplifier
		Cilasun et al. [82]	(2T-1M) x 3 SHE-CRAM	Interface Perpendicular	-	-	Bit-line sensing

Convolutional Neural Networks (CNN), and Spiking Neural Networks (SNN) using DS-IMC architectures.

1) Binary Neural Network (BNN)

Binary neural networks (BNN) [84] are neural networks that use binary weights and activations instead of floating-point numbers. This allows BNNs to use less memory and compute power, making them well-suited for deployment on resource-constrained devices such as mobile phones or Internet of Things (IoT) devices. Although utilizing binary weights and activations considerably reduces memory size and accesses, resulting in faster and more efficient inference [85], this is usually at the cost of a significant reduction in prediction accuracy.

Pan et al. [68] propose an MLC-STT-CIM accelerator for Binary Convolutional Neural Networks (BCNN) to reduce power consumption. Unlike other designs that store two addends separately [86], [87], this approach stores two bits in one cell and performs logic, and add operations between them in a single cell. While standard STT-MRAM stores one bit per cell, MLC-STT-MRAM stores two bits per cell, allowing it to store more data in the same space but making it more complex and slower.

The MLC-STT-MRAM cell structure includes small and large MTJs for information storage. The MLC-STT-CIM structure consists of an array of MLC-STT-MRAM, a modified sense circuit (MSC), and a mode controller, enabling logic, full-add, and memory modes. In memory mode, bits are read and written. In logic mode, various logic operations are realized. In full-add mode, sum and carryout are computed using XOR and AND operations.

The BCNN accelerator involves image and kernel banks, an auxiliary process unit, a global controller, and the MLC-STT-CIM. Input tensors are stored in image and kernel banks, and the MLC-STT-CIM handles convolutional operations, significantly reducing communication energy by eliminating read and write operations during convolution calculations.

This design offers power-efficient BCNN computations, making it a promising approach for AI applications [68].

Resch et al. [69] present PIMBALL, an adaptive BNN accelerator rooted in Processing In Memory (PIM) technology. PIMBALL uses computational RAM (CRAM) like spintronic PIM [88] but adds a transistor for efficient BNN operations. The CRAM design employs a 2T-1M architecture, combining memory and logic operations within the array. PIMBALL offers two array configurations: custom 1T-1M and 3T-1M. The 1T-1M option, more efficient than the 3T-1M, reduces transistors, maintaining array density. PIMBALL's PIM substrate handles key BNN components within the arrays. It excels in energy efficiency, offering high throughput with low power consumption. Tile sizes of 1024 x 1024 and 2048 x 2048 were evaluated, considering ideal and estimated peripheral circuitry. The research introduces memory cell and array designs for spintronic PIM, facilitating BNN processing, IoT binary neural networks, biomedical image analysis, and efficient inference through hardware acceleration, pruning, and quantization.

Cai et al. [70] introduce an approach for BNN in-memory computing using reconfigured foundry-supported bit-cells. They devise a digital in-MRAM method with a designer-friendly bit-cell array (BCA) configuration, transforming the standard 1T-1M cell into a pseudo-2T-1M structure by adding an extra transistor. This results in a 4T-2M structure that encodes synaptic weights as "+1" and "-1". The 4T-2M-based BCA enables one-step convolutions with in-memory XNOR operations, specifically designed for 3x3 convolution kernels. They also optimize the voltage sense amplifier (VSA) for BNN operations and introduce a dedicated bit-cell for XNOR-accumulation (XAC). With this energy-efficient system, recognition latency is improved by 21%, and energy consumption during operations is reduced by 30% compared to conventional XAC convolution on the MNIST dataset using a 16 nm FinFET process and MTJ compact model.

Pham et al. [89] present an STT-MRAM in-memory architecture tailored for Binary Neural Networks (BNNs) with a single-access Multiply-Accumulate (MAC) operation. This design allows unrestricted accumulation across rows, maximizing array utilization and enhancing BNN model scalability. Inputs pass through bitlines, and row-wise accumulation of products between inputs and weights occurs via source lines, enabling simultaneous activation of multiple rows. The architecture employs a 2T-2MTJ bitcell, binarizing both weights and neuron activations as +1 or -1, facilitating unrestricted accumulation and efficient use of memory array, allowing larger BNN models within a given memory capacity. Instead of power-intensive bitline current sensing, the select line voltage is used for BNN vector multiplication. The source line voltage is expressed using Norton theorem. Circuit enhancements, including time-based sensing (TBS) and boosting, are introduced in the periphery to improve energy, speed, and area robustness, ultimately enhancing BNN computation accuracy.

2) Convolutional Neural Networks (CNN)

CNN [90] is a type of deep learning model that uses convolutional layers that automatically and adaptively learn spatial hierarchies of features from input data. Over the years, there has been much research on improving the efficiency and performance of CNNs in both software and hardware. One such optimization involves leveraging STT-MRAM IMC for CNN applications to leverage the unique features of STT-MRAM to enable high-performance and low-overhead machine learning. This section reviews the state-of-the-art designs that leverage STT-MRAM IMC for CNNs.

Cai et al. [71] suggest a sparse approach for unreliable STT-MRAM in CNNs. Due to high failure rates in STT-MRAM, they propose a patch bank concept and an optimized flow for MTJ device circuits using sparsity. They employ cross-sensing to enhance STT-MRAM sensing reliability, dividing it into normal and patch banks. All 3x3 bit cells are read correctly within the access time in normal banks. In patch banks, marked bit cells and their neighbors are considered weak, reducing stored weights during sparse processing. The goal is to boost STT-MRAM performance and reliability.

The sparse strategy involves several steps to tackle weak bits and sensing delays in STT-MRAM. Initially, they obtain a pre-trained CNN model for target functions and detect weak bits, marking patch banks based on constraints like high sensing delays. To align these designs, they integrate errors from patch banks as regularization during the retraining of the CNN model. After N retraining epochs, the CNN model mirrors STT-MRAM's sparse pattern, accommodating error bits during training. This mitigates sensing errors in STT-MRAM, ensuring accurate data retrieval from each bit cell and improving model performance. These steps effectively reduce the impact of STT-MRAM sensing errors, enabling a high-performing model.

Angizi et al. [72] present MRIMA, an MRAM-based in-memory accelerator for efficient IMC. MRIMA repurposes STT-MRAM arrays into highly parallel units serving as both non-volatile memory and in-memory logic. It avoids complex logic integration, using bitline computing techniques for instant Boolean logic execution within a memory array in one clock cycle, sidestepping the multicycle logic problem seen in modern PIM platforms. MRIMA's structure includes multiple banks, each with memory matrices (mats) that share I/O and buffer within a chip. These mats contain processing-in-memory (PIM) subarrays, serving as the primary site for computational operations.

The authors demonstrate that MRIMA can enhance the speed of binary-weight CNNs (BWNNs) and low bit-width CNNs through its inherent in-memory bit-wise adder and convolver. To perform CNN operations, the kernels must undergo quantization before being mapped into the parallel computational subarrays for feature extraction using MRIMA's computation methods. The resulting feature map is then activated, generating the output feature maps. There are a few observations with this design:

- Grouping and transporting the pre-processed data into the same bank may lead to additional power consumption.
- Caching the post-processed data before utilizing it in the subsequent procedure can save valuable time in data

retrieval.

- Ensuring the precision of the reference current/voltage is crucial to guarantee the accuracy of the results, which can be a challenging task.
- The scalability of MRIMA to realize complex logic functions may be limited.

Kim et al. [73] propose BiMDiM, a bi-directional MRAM in-memory computing system that optimizes memory cell size during CNN operations. It reuses memory cells for intermediate sums and carries and uses a re-scheduling technique to reduce inefficient half-additions. Simulated on a 28nm CMOS process, BiMDiM achieves up to 53% better area efficiency than standard architectures.

BiMDiM consists of a weight storage array and four computing arrays (CAs) with 1T-1R STT-MRAM cells and logic operation circuits. Two CAs generate partial products, while the other two accumulate results. Current Lines (CLs) interconnect all arrays to facilitate parallel processing within memory rows, and transmission gates separate CLs within each section.

Unlike prior designs, such as [91], BiMDiM dramatically reduces computing array capacity requirements, improving overall area efficiency by optimizing resource utilization.

Kim et al. [74] present an energy-efficient PIM design that distributes computation between memory arrays and peripheral circuits, reducing data transfer during MAC operations. This design divides the memory into "groups" where partial sum operations happen simultaneously, followed by sequential accumulation. Complex accumulations involving extensive data transfer are handled by peripheral circuits. They also suggest further reducing energy consumption by using the "zero-skipping technique" called SnaPEA [92]. Simulations demonstrate a 50.4% energy reduction with the distributed accumulation scheme, plus an extra 8.2% reduction from SnaPEA integration.

3) Spiking Neural Networks (SNN)

Spiking neural networks (SNNs) aim to closely mimic how biological neurons work. They use spikes to process information, capturing temporal aspects well. This makes them ideal for studying brain-inspired computing [93] and enabling brain-like information processing, offering energy efficiency and suitability for neuromorphic hardware [94], [95]. Still, there are challenges in SNNs relevant to PIM designs, which researchers are tackling. Key challenges involve handling spike-based data complexity, devising efficient training methods due to non-differentiable spikes, designing scalable network architectures, and implementing specialized hardware for real-time spike-based computations.

Despite these challenges, solutions are emerging to leverage IMC for more efficient SNN processing. For instance, Cilasun et al. [75] propose a highly efficient SNN architecture using spintronic Computational RAM (CRAM) [96]. They employ non-volatile in-memory accelerators at the node level for parallel logic operations within memory, focusing on energy efficiency. Their communication-centric SNN accelerator minimizes data transfer overheads and adopts a scalable array connectivity. Through design exploration, they assess the accelerator's sensitivity to technology parameters and highlight energy consumption reductions compared to recent ASIC implementations.

The article introduces CRAM [96], a memory design capable of in-memory logic operations. CRAM has two variants: Spin Torque Transfer (STT) and Spin-Hall Effect (SHE). Both use a three-cell structure in a column, with even and odd bitlines (BLE/O) for reading and writing on MTJs. BLE/O determines input and output MTJs for logic operations, while logic lines (LL) link input and output cells for Boolean computation. Wordlines (WL) select rows for memory and logic ops. CRAM's unique advantage is its capacity for column-level and array-level parallelism, reducing the need for data transfer outside the array during logic processing.

The authors described how the Leaky-integrate-and-fire (LIF) neuron model could be implemented with SNN. In the LIF model, neurons have two main traits: leakage and firing. Leakage replicates the gradual loss of electrical charge in biological neurons, achieved by subtracting small values from the neuron's membrane potential over time. Firing happens when the membrane potential surpasses a threshold, causing the neuron to spike and transmit this signal to downstream neurons.

To implement this LIF model, the authors outlined the CRAM scheme in stages:

- Initialization: Setup involving parameters, constants, and data structures like lookup tables and local delays before processing.
- Dataflow: When pre-synaptic neuron spike data is received, it undergoes complex computations. This includes writing spike data into memory, followed by "AND" operations, summation, rounding, weight multiplication, and cascaded additions. These computations lead to the synaptic response current and membrane potential calculation.
- Routing and Connectivity: They used a Generalized De Bruijn Graph (GDBG) as the Network on Chip (NoC) architecture, connecting CRAM arrays. This topology choice minimizes data communication overheads within the SNN array.

Moreover, the authors employed Pairwise Spike-Timing-Dependent Plasticity (STDP) in their SNN setup. STDP is pivotal in SNNs, mirroring how synaptic connections evolve over time in biological neurons due to spike timing. It is a learning rule that models synaptic plasticity in SNNs. The core concept is that the timing relationship between pre-synaptic and post-synaptic spikes dictates whether the synaptic strength strengthens or weakens. Consistently, the connection strengthens if a pre-synaptic spike precedes a post-synaptic one. Conversely, the connection weakens if the pre-synaptic spike consistently lags behind the post-synaptic spike. This spike-timing dependence empowers SNNs to adapt and adjust their synaptic connections based on spike timing, aiding pattern recognition and temporal information encoding in the input data.

To implement STDP, the proposed CRAM scheme begins with initialization, where various parameters and constants are set based on the STDP equation. Following this, the engine processes spike distribution over time in the dataflow phase. Depending on whether it's pre-synaptic or post-synaptic, the operation occurs either to the left or right, with input representing spikes and output representing weights. The STDP equation is then applied to complete the process in a transposed layout.

Kang et al. [76] propose an STT-MRAM architecture design aimed at reducing area overhead and operation delay while optimizing peripheral circuits. In large synapse crossbar arrays (512x512) where multiple Multiply-and-Accumulate (MAC) operations run in parallel, there is a challenge of increased delay when reading the bit-cell voltage, negatively impacting overall performance. To overcome this issue, the authors suggest an optimized peripheral circuit design that incorporates the pre-charge technique.

The proposed design involves the addition of a simple PMOS transistor to each bitline (BL). This transistor charges both the selected and unselected bitlines to VDD during read operations, enabling faster driving of the selected bitline voltage when the read operations begin. By implementing this design, the authors achieve an 82% reduction in read voltage development delay, significantly improving read operation performance.

Nguyen et al. [77] present an approach called Binary Spiking Neural Network (BSNN) based on STT-MRAM. The BSNN incorporates residual learning using a surrogate gradient. In this framework, presynaptic spikes are fed to memory units through differential bitlines (BLs), while binarized weights are stored in a subarray of STT-MRAM. By utilizing the BLs for common inputs, vector-to-matrix multiplication can be performed in a single memory sensing phase, enabling high parallelism with low power consumption and minimal latency.

The authors introduce the concept of a dynamic threshold to simplify the implementation of synapses and neuron circuitry. This adjustable threshold facilitates the integration of the nonlinear batch normalization (BN) function into the integrate-and-fire (IF) neuron circuit. The incorporation of the BN function not only significantly enhances performance but also enables high regularity in circuit layouts.

Agrawal et al. [78] introduce SPARE, a design using ROM-embedded RAM tech to speed up SNNs. SPARE is a many-core architecture with a 2D PE array, global memory, and a central control unit. Each PE handles synapse and neuron tasks needed in various SNNs, and SNN layers are divided across PEs. Computation in SPARE happens in time steps, with only neuron data moving between PEs and global memory, while synapse data is read locally from the PE's RAM. This approach greatly cuts down data transfers compared to von Neumann systems.

The study explores two memory structures: R-SRAM and R-MRAM, both integrating read-only memory (ROM). R-SRAM modifies bit cells, adding an extra Word Line (WL), enabling operation in both RAM and ROM modes. Conversely, R-MRAM merges ROM with STT-MRAM, using an additional Bit Line (BL) for ROM data. While supporting RAM and ROM modes, R-MRAM prohibits simultaneous access. Performance tests reveal R-MRAM's superiority, particularly in Spiking Neural Networks for image classification, showing 1.75 times lower energy consumption than standard STT-MRAM arrays. This study underscores the advantages of integrating ROM into RAM-based in-memory hardware, exemplified in the SPARE project, advancing efficient cognitive computing.

B. IMAGE/SIGNAL PROCESSING

Image and signal processing are vital in various domains but face challenges. One is the need for efficient algorithms to handle large data volumes in real-time. This means developing parallel algorithms, hardware accelerators, and distributed processing frameworks. Another challenge is dealing with low-quality data, often noisy or corrupted. Advanced algorithms like noise reduction and signal filtering are essential. Efficient computation is crucial for low-overhead image processing. This section explores solutions to these challenges using STT-MRAM-based DS-IMC architectures.

Angizi et al. in [79] introduced PISA, a Processing-In-Sensor Accelerator. PISA is designed for real-time image processing in AI devices, emphasizing flexibility, energy efficiency, and high performance. PISA uses compute-pixels with non-volatile weight storage at the sensor side, enabling coarse-grained convolution operations within BWNNs. This significantly reduces power consumption during data conversion and transmission. PISA also includes a bit-wise near-sensor processing-in-DRAM computing unit for efficiently handling the remaining network layers. When an object is detected, PISA switches to a conventional sensing mode, using the near-sensor processing unit for image capture and fine-grained convolutions.

The authors introduce the Compute-Pixel (CP) and compute-add-ons for integrated sensing and processing. CP includes a pixel with three transistors and a photodiode. Compute add-ons have three transistors, two as deep triode region current sources and a 2:1 multiplexer controlled by an STT-MRAM device for storing binary weight data. PISA accelerates BWNN layers without off-chip feature map transmission, optimizing memory units into multiple banks with computational sub-arrays and local row buffers (LRBs). The array shares a digital processing unit (DPU) for quantization, pre-processing, linear batch normalization, and activation post-processing. Addressing reliability and latency, PISA incorporates a reconfigurable sense amplifier (SA) within the computational sub-array, enabling efficient (N)AND2 logic function implementation within a single cycle, leveraging DRAM cell charge-sharing.

He et al. [80] present a design for STT-MRAM arrays that serve as non-volatile memory while also enabling reconfigurable logic operations within the memory itself, eliminating the requirement for additional logic circuits. The output of the computation can be directly read from the memory cell using a modified peripheral circuit. This in-memory computation capability allows for localized data processing, followed by transferring the processed data to a primary processing unit like a CPU or GPU for more intricate and accurate computations. This approach significantly reduces power consumption by 8X, minimizes the need for long-distance communication, and enables extensive parallelism within the memory. The authors illustrate the effectiveness of their in-memory pre-processing approach through the implementation of an edge extraction algorithm.

C. DATA ENCRYPTION

Data encryption is crucial for protecting sensitive information, but applying it to STT-MRAM In-Memory Computing (IMC) poses challenges. Maintaining confidentiality is a top concern. Since encryption happens within memory, robust encryption algorithms and access controls are needed to prevent unauthorized access. Efficient key management is another challenge. This includes secure key storage, retrieval, protection, and key generation, distribution, and lifecycle management. The unique characteristics of STT-MRAM technology make these tasks complex. This section explores solutions for these challenges in STT-MRAM-based DS-IMC architectures, focusing on how they address confidentiality and key management concerns.

Parveen et al. [81] introduced the Highly Flexible In-Memory (HieIM) platform using STT-MRAM, operating in memory and computing modes with the 1T-1MTJ structure. In the 5T Magnetic Domain Wall (DMW) device, computation occurs in three stages: reset, compute, and sense. While efficient, this design takes 4 cycles, making it relatively slower. HieIM is assessed in two applications: in-memory data encryption with AES and in-memory bulk bit-wise Boolean vector logic operations. The latter offers energy savings and speed improvements but is destructive. For data encryption, HieIM consumes 51.5% less energy than CMOS-ASIC and 68.9% less than CMOL, a hybrid CMOS-nano device with unique properties [97]. HieIM also occupies 3.5x less area than DW-AES, requiring fewer cycles. However, data corruption risk exists in racetrack devices due to non-uniform domain wall velocities and substantial heating in larger devices.

Angizi et al. [72] evaluated their proposed MRIMA architecture using both a CNN accelerator and a data encryption accelerator (see Section IV-A2). In the evaluation as a data encryption accelerator, MRIMA demonstrates superior performance as an in-memory encryption engine that utilizes the Advanced Encryption Standard (AES) algorithm to encrypt data as it is written to the MRIMA architecture. AES typically applies four consecutive transformations to encrypt input data: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

These transformations are mapped to the MRIMA architecture as follows: In the SubBytes stage, each byte of the state matrix undergoes a look-up table (LUT)-based transformation within the MRIMA subarrays, utilizing consecutive read/write operations. In the ShiftRows stage, the state matrix undergoes a cyclical shift operation with a specific offset. In the MixColumns and AddRoundKey stages, parallel in-memory XOR2 operations are performed, along with implementing the Fast Row Copy (FRC) mechanism in memory mode.

V. SIMULATORS, TOOLS, AND EXPERIMENTAL APPROACHES USED WITH STT-MRAM IMC

Given the nascence of STT-MRAM, simulation and emulation tools play a significant role in the design of STT-MRAM IMC systems and have important implications for the quality of generated designs. These tools enable designers to explore, analyze, and optimize various aspects of system design, performance, power consumption, reliability, and integration. They provide a virtual environment for evaluating and refining STT-MRAM IMC systems, enabling the exploration of designs that are currently unrealized in practice, thereby contributing to the advancement of this promising technology.

This section reviews various state-of-the-art tools and simulators specifically utilized for in-memory computing using STT-MRAM. We will examine tools used at different architectural levels, including circuit level (or bit-cell level) and system level. This overview will guide researchers on the most appropriate tools for their investigations and facilitate practical research on STT-MRAM-based in-memory computing for different varieties of domains.

A. CIRCUIT-LEVEL TOOLS

As discussed in Section II-A1b, when changing the retention time of non-volatile memories, such as STT-MRAM, the thermal stability factor of the MTJ cell is usually adjusted based on Equation 2. To achieve the customization of the STT-MRAM, we have the flexibility of adjusting the volume of the free layer or the saturation magnetization depending on the application or domain. Here are some of the circuit-level tools utilized in the studied articles:

- Cadence Spectre: Cadence Spectre [98] is an Electronic Design Automation (EDA) environment that integrates various applications and tools into a unified framework, enabling support for all IC design and verification stages. This tool has been widely used for simulating CMOS and MTJ models. For instance, Angizi et al. [72] utilized this tool to simulate the 45 nm NCSU PDK library of their 1T-1R STT-MRAM device. Cai et al. [70] employed this tool to simulate their 16 nm FinFet and MTJ model. Parveen et al. and He et al. [81], [80] employed this tool, utilizing the 45 nm NCSU PDK as the CMOS library in their respective studies. Finally, Angizi et al. [79] used this tool to implement PISA with peripheral circuits using TSMC 65 nm-GP, achieving the desired performance parameters.
- SPICE Monte Carlo Simulations: SPICE Monte Carlo simulation is used in electronic circuit design to account for variations and uncertainties in component values and model parameters. It is a statistical analysis method that allows engineers to evaluate the performance and

TABLE 2: Different STT-RAM In-Memory Design Architectures

Domain	Applications	References	Array Size	Computation Operation	Simulated?	Tools Used
Machine Learning	BNN	Yu Pan et al. [68]			Simulated using 45 nm CMOS	Cadence Virtuoso CACTI, NVSim
		Resh et al. [69]	128KB => 1024 x 1024 512KB => 2048 x 2048	XNOR: 4 steps (3 temp values) Addition: 9n steps (using NAND) Subtraction: 5n+1 steps (and 5n temp bits) Batch Normalization: Multiply and add transformation	-	In-house simulator
		Cai et al. [70]	128 x 128 BCA	XNOR: WL input x In MRAM weight	Simulated using 16 nm FinFet process	Cadence Virtuoso
	CNN	Cai et al. [71]	3x3 bit cells		Simulated using 28 nm CMOS, 40 nm MTJ model	Cadence Virtuoso
		Angizi et al. [72]	512 x 256 bit-cells/sub-array 16 sub-arrays/mat, 2 x 2 mats/bank, 4 x 4 banks/group, 4 groups, 512 MB	In-memory bit-wise Adder In-memory bitwise convolver: Logic AND bitcount, and bitshift operations	Simulated using 45 nm NCSU PDK	Cadence Spectre/ Verilog-A NVSim, Design Compiler
		Kim et al. [73]		NAND and Full Addition	Simulated using 28 nm CMOS	Candence Virtuoso
		Kim et al. [74]	3 x 3 x64 kernel size MAC	MAC operation using NAND and Full Addition (with using distributed and Zero-skipping algorithm)	Simulated using 28 nm CMOS	-
	SNN	Cilasun et al. [75]	1024x512 cells	LIF model: Convolution, Multiplication, Addition Comparison, PRNG	-	NVSim
		Kang et al. [76]	512x512 cells	-	-	-
		Nguyen et al. [77]	32 x 288 cells => 32 channels of 3 x 3 kernel size	MAC operation	Simulated using 65 nm CMOS	-
		Agrawa et al. [78]		SNN computations: synapse model, Neuron model block, plasticity model block	Simulated using 45 nm CMOS	Verilog, Synopsy Design Compiler, CACTI, NVSim
Image Processing	Image Processing	Angizi et al. [79]	1024x256 cells 4x4/bank 16x16 banks	MAC operation		Verilog A, Cadence Spectre, PyTourch
		He et al. [80]	-	Binary Image: black and white color is detected and sent to MSA. Grayscale Image: In-memory OR operation is performed	Simulated using 45 nm NCSU PDK	NVSim, In-house developed C++
Data Encryption	Encryption	Parveen et al. [81]	4x4 banks, 2x2mats/bank 1024x512 sub-array/Mat	Consecutive transformations: SubBytes, Shiftrows, MixColumns AddRoundKey	-	In-house C++ code, NVSim, Gem5, McPAT, Synopsys Design Compiler
		Angizi et al. [72]	4x4 matrix (state matrix)	Consecutive transformations: SubBytes (LUT read/write), Shiftrows (Shift operation), MixColumns (XOR LUT) AddRoundKey (XOR)	-	Cadence Spectre/ Verilog-A, NVSim, Design Compiler
		Cilasun et al. [82]	32 Processing Units 1PU => 4x(512x512 CA) + 692x(64x64 PAs) Memory accelerator = >9.5MB	-	-	RAM Simulator, NVSim

robustness of a circuit by running multiple simulations with randomized parameter values. By running a large number of Monte Carlo simulations, typically ranging from hundreds to thousands or more, statistical measures such as mean values, standard deviations, and probability distributions can be obtained for circuit responses such as voltage, current, timing, and other performance metrics. These statistical results provide insights into the robustness and reliability of the circuit under real-world operating conditions.
Angizi et al. [72] utilized this simulation technique to verify their design using the 45nm NCSU PDK library. Similarly, in [79] the authors utilized this technique to simulate PISA's circuit-level variations at 300K temperature using 10,000 runs. Gajaria et al. [67] conducted Monte Carlo simulations using 10,000 samples with varying STT-MRAM cell resistance values to study the impacts of process variations.

- Verilog-A: Verilog-A [99], [100] is a hardware description language commonly used to model and simulate analog and mixed-signal systems. Regarding STT-MRAM, Verilog-A can be utilized to model the behavior and characteristics of STT-MRAM devices at a higher level of abstraction.
  Angizi et al. [72] utilized a Verilog-A model of 1T-1R STT-MRAM device to co-simulate with the interface CMOS circuits in Cadence Spectre using a 45 nm NCSU PDK library and SPICE to verify the proposed design and acquired the performance of their CNN design. Angizi et al. [79] utilized a Verilog-A model of NVM elements to co-simulate with CMOS circuits in Cadence Spectre and SPICE to evaluate their PISA design.
- Cadence Virtuoso: Cadence Virtuoso is a popular Electronic Design Automation (EDA) tool suite used for designing and verifying integrated circuits. It provides a comprehensive set of tools and features that can be utilized for STT-MRAM design. Cadence Virtuoso is typically used for the entire STT-MRAM design flow, including schematic design, layout design, device characterization, simulation, and physical verification. It provides a comprehensive platform for designing, analyzing, and optimizing STT-MRAM circuits, enabling efficient and reliable STT-MRAM designs.
  Yu Pan et al. [68] utilized this tool to evaluate the current and voltage parameters using 45nm CMOS technology in Cadence Virtuoso with their MLC-STT-CIM design in performing a full-add operation of their design. Cai et al. [70] utilized this tool to simulate their 16 nm FinFet process and MTJ compact models. Cai et al. [71] utilized the Spectre simulator in Cadence Virtuoso front-end to simulate the 28 nm CMOS and 40 nm MTJ compact model of their design. Kim et al. [73] employed this tool mainly for area estimation of the layout design of the memory cell to perform λ-based design rule to simulate their 28 nm CMOS design.
- Synopsys Design Compiler: Like Cadence Virtuoso, Synopsys Design Compiler is a widely used synthesis tool in the field of EDA. It is commonly employed in the design and optimization of digital integrated circuits, including STT-MRAM. It enables efficient translation of RTL descriptions into gate-level netlists, technology mapping to STT-MRAM library cells, timing optimization, power optimization, area optimization, and analysis of the synthesized design. The tool enhances the design process and helps achieve optimal performance, power consumption, and area utilization in STT-MRAM circuits.
  During their design evaluation, Angizi et al. [72] utilized this tool to synthesize their controllers and add-on circuits. Agrawal et al. [78] employed this tool using an IBM 45nm technology library to estimate the power and area consumption of their 1T-1M STT-MRAM design. Parveen et al. [81] utilized this tool to evaluate their AES in CMOS ASIC to evaluate the performance of their 32nm technology.

B. ARCHITECTURE/SYSTEM-LEVEL TOOLS

- Gem5: Gem5 [101], [102] is a popular simulation framework widely used for architectural research and performance evaluation of computer systems. It can be used to simulate STT-MRAM in conjunction with other components of the memory hierarchy. Gem5 can be utilized with STT-MRAM for various purposes, including file configuration, model integration, memory hierarchy configuration, workload execution, performance analysis, system-level experiments, and validation and verification. Several works (e.g., [81], [72], [67]) have used gem5 to model various portions of their STT-MRAM IMC designs, including simulation of workload binaries, implementation of relaxed retention STT-MRAM, comparison to SRAM, etc.
- McPAT (Microprocessor Performance, Power, and Area) [103] is a well-known tool for modeling architectural power and performance. Although McPAT does not directly model memory technologies like STT-MRAM, it's often used alongside STT-MRAM models to estimate the power and performance of systems that include STT-MRAM as a memory technology. The precision of these estimates relies on the accuracy of the STT-MRAM models and assumptions. Thus, reliable STT-MRAM models are crucial to obtaining meaningful power and performance projections for STT-MRAM-based systems. Previous studies (e.g., [81], [67]) have employed McPAT, often in conjunction with other simulators like gem5, to gauge system-level power and energy consumption.
- CACTI: CACTI [104] is a widely-used tool that provides a framework for estimating key metrics such as access latency, power consumption, area, and leakage power for different memory architectures and technology nodes. It considers various design parameters, including cache size, associativity, block size, and technology-specific parameters, to provide detailed performance and power estimates. The tool utilizes analytical models

and technology-specific parameters to predict memory performance and power consumption at different levels of the memory hierarchy, ranging from individual cells and bitlines to cache arrays and multi-level caches. Agrawal et al. [78] used this tool to model the memory unit's energy and power consumption using 45nm technology node. Angizi et al. [72] utilized this tool to estimate the performance of eDRAM and SRAM used to compare their MRIMA design. Pan et al. [68] utilized this tool to estimate the performance of their design, while Angizi et al. [79] used it to model timing, energy, and area of their IMC designs.

- NVSim: NVSim [105] is a simulation framework specifically designed for non-volatile memory (NVM) technologies, including STT-MRAM. NVSim allows researchers and designers to model and evaluate the performance, energy consumption, and other characteristics of STT-MRAM-based memory systems. NVSim can be used with STT-MRAM in many ways, which include system configuration, STT-MRAM modeling, power and performance analysis, sensitivity analysis, design space exploration, etc. Cilasun et al. [82] employed NVSim to estimate peripheral time and energy overhead from the SA and row decoders. Parveen et al. [81], He, et al. [80], Pan et al. [68], Agrawal et al. [78], and several other works have used NVSim to verify the system level performance as it relates to latency, dynamic energy, leakage power, and area of their designs.

VI. CHALLENGES WITH DS-IMC USING STT-MRAM

Creating an efficient, low-power, cutting-edge DS-IMC utilizing STT-MRAM presents numerous challenges. We provide a compilation of general design guidelines for in-memory computing with STT-MRAMs, gleaned from our extensive research, and suggestions for research directions to address existing knowledge gaps. These guidelines may vary according to specific technology nodes, design constraints, and implementation considerations but offer a starting point when embarking on STT-MRAM-based in-memory computing system design.

A. DESIGN GUIDELINES

1) Memory Mapping

Efficient memory mapping is crucial to exploit the advantages of STT-MRAM in IMC accelerators. Careful consideration should be given to mapping the relevant data structures and computations to STT-MRAM arrays to minimize data movement and maximize data locality. Mapping techniques such as data partitioning, data replication, and intelligent data placement can be employed to optimize memory access patterns and minimize energy and latency overheads. For example, Angizi et al. [72] investigated the utilization of data partitioning and intelligent data placement techniques in their MRIMA design. They achieved this by mapping either raw or pre-processed data to specific computational subarrays within the mats. This approach aimed to maximize data locality and minimize data movement. On the other hand, Resch et al. [69] employed a data duplication technique in their PIMBALL design to enable efficient computation of BNN.

2) Data Organization

The organization of data within STT-MRAM arrays can significantly impact the performance of IMC accelerators. Proper data organization techniques, such as data compression, data encoding, and data grouping, should be employed to reduce memory footprints, improve data access efficiency, and exploit the parallelism offered by STT-MRAM devices. For instance, Kim et al. [73] and Resch et al. [69] proposed a memory array design that incorporates the Multiply-Accumulate (MAC) operation. They achieved this by partitioning the memory array into multiple sub-computing arrays, which are utilized for various operations, such as adding partial products and accumulating results. This approach significantly enhances the data access efficiency in their respective designs. A significant challenge in bit-line computation is ensuring that data are aligned within the same sub-array and share the same bit-line to enable computation. Compiler enhancements could be developed to address this issue. For instance, Parveen et al. [81] introduced an architecture aimed at resolving this alignment problem, but it comes at the cost of not being able to utilize parallelism.

3) Parallelism and Pipelining

STT-MRAM IMC accelerators can benefit from exploiting parallelism at various levels. Instruction-level parallelism, data-level parallelism, and task-level parallelism should be considered to enable efficient and simultaneous execution of multiple operations. Pipelining techniques can also be employed to overlap computation and memory access stages, improving overall throughput. Resch et al. [69] implemented pipelining in their CNN computations by utilizing a 9-stage pipeline. They employed multiple memory arrays to simultaneously perform computations at each layer for different input images. Agrawal et al. [78] investigated the use of an inter-layer pipelining scheme in their SPARE design. This approach allowed them to execute their 2-dimensional PE array, mitigating potential performance issues effectively.

4) Address Translation and Mapping

Efficient address translation and mapping techniques are essential when using STT-MRAM in IMC accelerators. Address translation schemes, such as address remapping, virtual-to-physical address translation, and caching mechanisms, should be considered to reduce the overhead of memory address translation and improve overall performance. Angizi et al. [72] introduced a caching mechanism in their design by incorporating a local row buffer (LRB) and digital processing unit (DPU) between two adjacent computational subarrays. During the first half-cycle, data read from the STT-MRAM cell is stored in the LRB, known as the first-row copy (FRC). In the second half-cycle, this stored data is written back to the destination row, improving overall design performance.

5) Power/Energy Management

STT-MRAM devices can consume significant power, especially during write operations. Power management techniques, such as voltage scaling, power gating, and clock gating, should be employed to minimize power consumption without compromising performance. Based on workload characteristics and power-performance trade-offs, dynamic power management policies can be implemented to optimize power and energy usage in IMC accelerators. Cai et al. [70] conducted voltage scaling simulations, specifically focusing on the 1-bit XNOR operation. They determined the threshold voltage to be 0.6V and achieved a significant reduction of 54% in energy consumption. Gajaria et al. [67] achieved significant write energy saving by using reduced retention STT-MRAM in their design.

6) Error Correction and Fault Tolerance

STT-MRAM devices are susceptible to various types of errors, including write disturbance, read disturbance, and process variations. Error correction codes (ECC) and fault tolerance mechanisms, such as redundancy and error detection and correction techniques, should be implemented to ensure data integrity, reliability, and resilience in the presence of errors. For example, Resch et al. [69] addressed write disturbance issues in their PIMBALL design by increasing the switching current (Ic) by 1.5x, thereby improving the write current. In another study, Gajaria et al. [67] tackled the effects of process variation in their processing-in-cache and processing-in-memory designs. They accomplished this by incorporating multiple retention times through variations in the STT-MRAM parameters, such as the free layer thickness and anisotropy constant. To assess the impact of these variations, they conducted Monte Carlo simulations using 10,000 samples with varying STT-MRAM cell resistance values, employing SPICE as the simulation tool.

7) Integration and Interface

Designing an STT-MRAM-based IMC accelerator requires careful integration with the overall system architecture. Interfaces, protocols, and communication mechanisms should be designed to seamlessly connect the accelerator with the CPU, memory hierarchy, and other system components. Compatibility with existing bus protocols, memory coherence schemes, and interconnect architectures should be ensured. Cilasun et al. [75] proposed the use of a generalized De Bruijn graph (GDBG) topology to connect their CRAM arrays efficiently. This topology was designed to mitigate issues such as congestion and packet drops between each SNN layer. Angizi et al. [72] established a direct connection between their MRIMA hardware and the memory bus using the PCI-Express lane. They also introduced a virtual machine and instruction set architecture (ISA) to facilitate parallel thread execution. Basic instruction sets or commands were generated and written to predefined memory-mapped address ranges defined in the memory type range registers (MTRRs).

8) Verification and Testing

Rigorous verification and testing methodologies should be employed to validate the functionality, performance, and reliability of the STT-MRAM-based IMC accelerator. Verification techniques, including simulation, emulation, and formal methods, should be utilized to ensure correct operation and adherence to design specifications. Angizi et al. [72] conducted comprehensive verification and testing of their design by developing a device-to-architecture evaluation framework. They began by modeling the STT-MRAM bit cell at the device level and performed circuit-level simulations using various CMOS circuit simulation tools (refer to Section V and Table 2) to validate the functionality, performance, and reliability of the STT-MRAM cell design. Similarly, Pan et al. [68] adopted a similar verification approach for their MLC-STT-CIM cell. They modeled the cell and conducted circuit-level simulations using the simulation tools described in Section V and Table 2 to verify the functionality, performance, and reliability of the MLC-STT-CIM cell design.

B. CHALLENGES AND OPPORTUNITIES FOR FUTURE RESEARCH

1) Domain Representation

The critical first step in designing a DS-IMC's processing element (PE) is to analyze applications and categorize them into distinct target domains. This process involves extracting structural information or computational kernels from these applications of interest [106], [107]. Operations or computations that frequently occur or that are computationally expensive can be potential targets for specialized implementation, as the benefits derived from streamlining these repeated operations contribute to overall system efficiency.

By mapping out an application's flow graph, we can accurately detail the control and data flow dependencies, enabling the DS-IMC to harness any inherent parallelism and optimize performance [108]. Within this graph, nodes represent kernels, edges reflect the dependency between different kernels, and the weight assigned to the edges indicates the data volume communicated between two kernels [109], [110].

The rapid evolution of applications and domains in today's technology landscape makes manual analysis of these applications practically unfeasible. This is particularly true in rapidly evolving and high-impact areas such as machine learning, computer vision and image processing algorithms in autonomous driving applications, wireless and radar applications in communication and surveillance, etc. [111], [112], [113], [110]. Consequently, there is a compelling need for frameworks that can automatically determine the application domain, as well as extract kernels and data flow graphs [110]. This information about kernels and flow graphs assists designers in deciding the number and types of PEs required by the DS-IMC, or even assessing whether the DS-IMC itself can manage most of the necessary processing tasks.

2) Hardware Architecture and Design

Designing an energy-efficient hardware architecture for DSC-IMC using STT-MRAM presents a unique set of challenges, as this technology is relatively novel and complex. The hardware components of such architecture encompass fixed-function accelerators, specialized processors or DSAs or PEs, General-Purpose Processors, and on-chip interconnect, as discussed in Section II-B2.

There are challenges with STT-MRAM design in general that might have significant implications for the design of STT-MRAM-based IMC architectures. These challenges are described in II-A2. Furthermore, there are certain challenges when it comes to designing a novel compute unit architecture using STT-MRAM. One such challenge is understanding how different cell array architectures perform logic operations. For instance, the CRAM architecture [91], [88], [75] uses 2T-1M cells for computation. This process involves connecting three MTJ devices to the logic line, setting the output to "0" or "1", and applying the appropriate bias voltage to the MTJs' bit select lines (BSLs), while the output MTJ is grounded to implement "NAND" or "NOR" operations. This approach reduces the need for some hardware components, such as sense amplifiers and some logic gates, which might be necessary in other designs like those described in [79], [81], [72].

Reducing communication overhead poses another challenge, necessitating designs that enable logic operations across multiple rows instead of the same row, as seen in the original CRAM architecture [96]. Zabihi et al. proposed addressing this challenge by incorporating switches between logic lines, which, when turned on, would facilitate communication between MTJs in different rows and enhance computational efficiency [91].

Peripheral circuitry design introduces additional challenges, particularly in supporting in-memory computational tasks. This involves integrating sense amplifiers for determining logic operations [67], [86], [87] and bitline driver circuitry for applying required voltages in both memory and logic modes [91].

Furthermore, efficient implementation of complex computation units like a full adder (FA) poses a critical challenge, alongside exploring optimization schemes to minimize overhead. Wang et al. [96] implemented a NAND-based logic for an FA with 9 stages, while Zabihi et al. [91], [88] introduced a more efficient FA using majority logic and implemented multiplication operations using Wallace/Dadda trees. These challenges collectively shape the landscape of STT-MRAM-based IMC architectures, demanding innovative solutions for optimal performance and functionality.

3) Resource Management in DS-IMC

To harness the potential of DS-IMC, efficient utilization of available PEs for task execution is crucial[114], [115]. Resource management techniques for DS-IMC can be broadly categorized into static (or design time) and dynamic (or runtime) approaches. Static algorithms leverage design time information to manage resources, offering optimal solutions unconstrained by computation and latency. However, they lack access to runtime information, making them inefficient in certain scenarios. As DS-IMCs often support multiple simultaneous, runtime resource management becomes essential, demanding effective dynamic approaches.

DS-IMCs face resource management challenges. Evaluating diverse PEs for varied applications requires assessing multiple execution alternatives for optimal solutions. Managing resources must address the unique characteristics of concurrently running applications, ensuring they meet computing requirements, performance goals, deadlines, and power constraints. Applications like signal processing or autonomous driving, dealing with repetitive operations on streaming data frames, add complexity when incoming frames overlap with ongoing tasks, further complicating resource management.

Developing novel resource management techniques heavily relies on application types and hardware components chosen. Hence, domain representation and hardware design outputs become critical inputs to study task scheduling techniques and voltage-frequency scaling policies, aiding resource management in DS-IMC.

4) Software Development

Designing and programming DS-IMC architectures present significant challenges, as the goal is to achieve optimal performance and energy efficiency while abstracting platform-specific complexities for end-users. To address these challenges and maximize DS-IMC potential, the following considerations should be made when creating the software infrastructure:

- Utilizing Domain-Specific Languages (DSLs) is crucial for optimized code in specific application domains, enhancing platform performance and portability [116]. DSL-based compilers simplify code using domain-specific abstractions, improving accessibility for programmers [117]. Hyper, for instance, leverages LLVM for dynamic translation to machine code, reducing recursive function calls and enhancing data retention in registers for improved code and data locality [118].
- Addressing Programming Challenges: Programming DS-IMC architectures can be complex due to memory hierarchy, data movement, hardware heterogeneity, and identifying parallelism in applications and hardware [119], [120]. Frameworks like HPVM and those presented in [115] compile applications into dataflow graphs, leveraging inherent task- and data-level parallelism in applications and hardware to optimize energy efficiency and computation costs.
- Ensuring End-User Compatibility: PEs in DS-IMCs should expose applications programming interfaces (APIs) that allow existing applications to execute with minimal modifications [121]. Runtime frameworks play a crucial role, using application source code and state-of-the-art resource management techniques [122]. These frameworks bridge hardware components, manage resources, and utilize domain representation techniques in

DS-IMCs. During application execution, they identify domain-specific kernels, allocate them to PEs, and execute them on energy-efficient hardware. Integration of compiler, kernel library, and profiler aspects through software runtime frameworks further improves performance and user productivity.

- Utilizing Compilers for bit-line computing data alignment: Compilers play a crucial role in optimizing data alignment for bit-line computing using STT-MRAM. They achieve this by reorganizing memory layouts, performing loop transformations, enabling vectorization and parallelization, managing the memory hierarchy efficiently, providing data alignment directives, automatically transforming data structures, and generating target-specific optimizations. These compiler techniques are essential for ensuring that data required for bit-line operations are stored contiguously and share the same bit-line within STT-MRAM, enhancing data locality, reducing realignment overhead, and ultimately maximizing the benefits of STT-MRAM in energy-efficient and high-performance computing applications.
- A unique Instruction Set Architecture (ISA) is designed for STT-MRAM control, offering specific instructions for reading, writing, erasing, and overall control. It includes memory addresses, control signals, and various addressing modes. Microcode or firmware translates these instructions into low-level signals for the processor, enhancing memory management and data processing through streamlined interaction with STT-MRAM.

VII. CONCLUSION

The rapid growth of big data and the increasing demand for high-performance computing have driven the exploration of novel computing architectures. In-memory computing architectures, which capitalize on the high-density and low-latency characteristics of modern memory technologies, have emerged as promising solutions for domain-specific computing applications. Among these architectures, STT-MRAM stands out due to its non-volatility, high endurance, and low power consumption. This survey paper has provided a comprehensive overview of the current state of the art in STT-MRAM-based in-memory computing architectures for domain-specific computing. We have addressed the challenges, opportunities, and trade-offs associated with these architectures, shedding light on their potential for future development.

REFERENCES

[1] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing data where it makes sense: Enabling in-memory computation," *Microprocessors and Microsystems*, vol. 67, pp. 28–41, 2019.

[2] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "Gpus and the future of parallel computing," *IEEE micro*, vol. 31, no. 5, pp. 7–17, 2011.

[3] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[4] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[5] J. T. Pawlowski, "Hybrid memory cube (hmc)," in *2011 IEEE Hot chips 23 symposium (HCS)*. IEEE, 2011, pp. 1–24.

[6] J. Kim and Y. Kim, "Hbm: Memory solution for bandwidth-hungry processors," in *2014 IEEE Hot Chips 26 Symposium (HCS)*. IEEE, 2014, pp. 1–24.

[7] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.

[8] D. Ielmini and G. Pedretti, "Device and circuit architectures for in-memory computing," *Advanced Intelligent Systems*, vol. 2, no. 7, p. 2000040, 2020.

[9] A. Limaye and T. Adegbija, "Dosage: generating domain-specific accelerators for resource-constrained computing," in *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2021, pp. 1–6.

[10] Y. Chi, W. Qiao, A. Sohrabizadeh, J. Wang, and J. Cong, "Democratizing domain-specific computing," *Communications of the ACM*, vol. 66, no. 1, pp. 74–85, 2022.

[11] H.-S. P. Wong and S. Salahuddin, "Memory leads the way to better computing," *Nature nanotechnology*, vol. 10, no. 3, pp. 191–194, 2015.

[12] J.-G. Zhu, "Magnetoresistive random access memory: The path to competitiveness and scalability," *Proceedings of the IEEE*, vol. 96, no. 11, pp. 1786–1798, 2008.

[13] S. W. Fong, C. M. Neumann, and H.-S. P. Wong, "Phase-change memory—towards a storage-class memory," *IEEE Transactions on Electron Devices*, vol. 64, no. 11, pp. 4374–4385, 2017.

[14] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide rram," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[15] S. P. Park, S. Gupta, N. Mojumder, A. Raghunathan, and K. Roy, "Future cache design using stt mrams for improved energy efficiency: Devices, circuits and architecture," in *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 492–497.

[16] K. Kuan and T. Adegbija, "Energy-efficient runtime adaptable l1 stt-ram cache design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 6, pp. 1328–1339, 2020.

[17] D. Gajaria and T. Adegbija, "Arc: Dvfs-aware asymmetric-retention stt-ram caches for energy-efficient multicore processors," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 439–450. [Online]. Available: https://doi.org/10.1145/3357526.3357553

[18] P. Chi, S. Li, Y. Cheng, Y. Lu, S. H. Kang, and Y. Xie, "Architecture design with stt-ram: Opportunities and challenges," in *2016 21st Asia and South Pacific design automation conference (ASP-DAC)*. IEEE, 2016, pp. 109–114.

[19] K. Zhang, K. Cao, Y. Zhang, Z. Huang, W. Cai, J. Wang, J. Nan, G. Wang, Z. Zheng, L. Chen *et al.*, "Rectified tunnel magnetoresistance device with high on/off ratio for in-memory computing," *IEEE Electron Device Letters*, vol. 41, no. 6, pp. 928–931, 2020.

[20] K. S. Mohamed and K. S. Mohamed, "Near-memory/in-memory computing: pillars and ladders," *Neuromorphic Computing and Beyond: Parallel, Approximation, Near Memory, and Quantum*, pp. 167–186, 2020.

[21] H. Amrouch, N. Du, A. Gebregiorgis, S. Hamdioui, and I. Polian, "Towards reliable in-memory computing: From emerging devices to post-von-neumann architectures," in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2021, pp. 1–6.

[22] H. Cai, B. Liu, J. Chen, L. Naviner, Y. Zhou, Z. Wang, and J. Yang, "A survey of in-spin transfer torque mram computing," *Science China Information Sciences*, vol. 64, no. 6, p. 160402, 2021.

[23] S. Salehi, D. Fan, and R. F. Demara, "Survey of stt-mram cell design strategies: Taxonomy and sense amplifier tradeoffs for resiliency," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, pp. 1–16, 2017.

[24] B. Li, B. Yan, and H. Li, "An overview of in-memory processing with emerging non-volatile memory for data-intensive applications," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, 2019, pp. 381–386.

[25] N. P. Jouppi, C. Young, N. Patil, and D. Patterson, "A domain-specific architecture for deep neural networks," *Communications of the ACM*, vol. 61, no. 9, pp. 50–59, 2018.

[26] W. J. Dally, Y. Turakhia, and S. Han, "Domain-specific hardware accelerators," *Communications of the ACM*, vol. 63, no. 7, pp. 48–57, 2020.

[27] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Ai accelerator survey and trends," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2021, pp. 1–9.

[28] Y. Li, T. Bai, X. Xu, Y. Zhang, B. Wu, H. Cai, B. Pan, and W. Zhao, "A survey of mram-centric computing: From near memory to in memory," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[29] F. Staudigl, F. Merchant, and R. Leupers, "A survey of neuromorphic computing-in-memory: architectures, simulators, and security," *IEEE Design & Test*, vol. 39, no. 2, pp. 90–99, 2021.

[30] Y. Zhang, L. Zhang, W. Wen, G. Sun, and Y. Chen, "Multi-level cell stt-ram: Is it realistic or just a dream?" in *Proceedings of the International Conference on Computer-Aided Design*, 2012, pp. 526–532.

[31] J. Choe, "Memory technology 2021: Trends & challenges," in *2021 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. IEEE, 2021, pp. 111–115.

[32] H. Noguchi, K. Ikegami, K. Kushida, K. Abe, S. Itai, S. Takaya, N. Shimomura, J. Ito, A. Kawasumi, H. Hara *et al.*, "7.5 a 3.3 ns-access-time 71.2 μw/mhz 1mb embedded stt-mram using physically eliminated read-disturb scheme and normally-off memory architecture," in *2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*. IEEE, 2015, pp. 1–3.

[33] H.-C. Yu, K.-C. Lin, K.-F. Lin, C.-Y. Huang, Y.-D. Chih, T.-C. Ong, J. Chang, S. Natarajan, and L. C. Tran, "Cycling endurance optimization scheme for 1mb stt-mram in 40nm technology," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*. IEEE, 2013, pp. 224–225.

[34] S. Chatterjee, M. Rasquinha, S. Yalamanchili, and S. Mukhopadhyay, "A scalable design methodology for energy minimization of sttram: A circuit and architecture perspective," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 19, no. 5, pp. 809–817, 2010.

[35] W. Butler, X.-G. Zhang, T. Schulthess, and J. MacLaren, "Spin-dependent tunneling conductance of fe| mgo| fe sandwiches," *Physical Review B*, vol. 63, no. 5, p. 054416, 2001.

[36] S. Motaman, S. Ghosh, and N. Rathi, "Impact of process-variations in sttram and adaptive boosting for robustness," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 1431–1436.

[37] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*. IEEE, 2011, pp. 50–61.

[38] D. Apalkov, A. Khvalkovskiy, S. Watts, V. Nikitin, X. Tang, D. Lottis, K. Moon, X. Luo, E. Chen, A. Ong *et al.*, "Spin-transfer torque magnetic random access memory (stt-mram)," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 9, no. 2, pp. 1–35, 2013.

[39] Z. Diao, Z. Li, S. Wang, Y. Ding, A. Panchula, E. Chen, L.-C. Wang, and Y. Huai, "Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory," *Journal of Physics: Condensed Matter*, vol. 19, no. 16, p. 165209, 2007.

[40] A. Amirany, M. Meghdadi, M. H. Moaiyeri, and K. Jafari, "Stochastic spintronic neuron with application to image binarization," in *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*. IEEE, 2021, pp. 1–5.

[41] A. Antonyan, S. Pyo, H. Jung, and T. Song, "Embedded mram macro for eflash replacement," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–4.

[42] L. Wei, J. G. Alzate, U. Arslan, J. Brockman, N. Das, K. Fischer, T. Ghani, O. Golonzka, P. Hentges, R. Jahan *et al.*, "13.3 a 7mb stt-mram in 22ffl finfet technology with 4ns read sensing time at 0.9 v using write-verify-write scheme and offset-cancellation sensing technique," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019, pp. 214–216.

[43] A. Raychowdhury, D. Somasekhar, T. Karnik, and V. De, "Design space and scalability exploration of 1t-1stt mtj memory arrays in the presence of variability and disturbances," in *2009 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2009, pp. 1–4.

[44] Y. Jin, M. Shihab, and M. Jung, "Area, power, and latency considerations of stt-mram to substitute for main memory," in *Proc. ISCA*, 2014.

[45] E. Barati, M. Cinal, D. Edwards, and A. Umerski, "Calculation of gilbert damping in ferromagnetic films," in *EPJ Web of Conferences*, vol. 40. EDP Sciences, 2013, p. 18003.

[46] S. M. Nair, R. Bishnoi, and M. B. Tahoori, "Mitigating read failures in stt-mram," in *2020 IEEE 38th VLSI Test Symposium (VTS)*. IEEE, 2020, pp. 1–6.

[47] S. Seyedfaraji, J. T. Daryani, M. M. S. Aly, and S. Rehman, "Extent: Enabling approximation-oriented energy efficient stt-ram write circuit," *IEEE Access*, vol. 10, pp. 82 144–82 155, 2022.

[48] T. Pramanik, U. Roy, P. Jadaun, L. F. Register, and S. K. Banerjee, "Write error rates of in-plane spin-transfer-torque random access memory calculated from rare-event enhanced micromagnetic simulations," *Journal of Magnetism and Magnetic Materials*, vol. 467, pp. 96–107, 2018.

[49] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[50] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th annual international symposium on Computer architecture*, 2011, pp. 365–376.

[51] E. Track, N. Forbes, and G. Strawn, "The end of moore's law," *Computing in Science & Engineering*, vol. 19, no. 2, pp. 4–6, 2017.

[52] A. Amarnath, S. Pal, H. T. Kassa, A. Vega, A. Buyuktosunoglu, H. Franke, J.-D. Wellman, R. Dreslinski, and P. Bose, "Heterogeneity-aware scheduling on socs for autonomous vehicles," *IEEE Computer Architecture Letters*, vol. 20, no. 2, pp. 82–85, 2021.

[53] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture," *Communications of the ACM*, vol. 62, no. 2, pp. 48–60, 2019.

[54] T. Norrie, N. Patil, D. H. Yoon, G. Kurian, S. Li, J. Laudon, C. Young, N. Jouppi, and D. Patterson, "The design process for google's training chips: Tpuv2 and tpuv3," *IEEE Micro*, vol. 41, no. 2, pp. 56–63, 2021.

[55] C. Chen, X. Liu, H. Peng, H. Ding, and C.-J. R. Shi, "Ifpna: A flexible and efficient deep learning processor in 28-nm cmos using a domain-specific instruction set and reconfigurable fabric," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 346–357, 2019.

[56] N. Challapalle, K. Swaminathan, N. Chandramoorthy, and V. Narayanan, "Crossbar based processing in memory accelerator architecture for graph convolutional networks," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[57] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 751–766.

[58] D. K. Mandal, S. Jandhyala, O. J. Omer, G. S. Kalsi, B. George, G. Neela, S. K. Rethinagiri, S. Subramoney, L. Hacking, J. Radford *et al.*, "Visual inertial odometry at the edge: A hardware-software co-design approach for ultra-low latency and power," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 960–963.

[59] Y. Turakhia, G. Bejerano, and W. J. Dally, "Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 199–213, 2018.

[60] D. Koeplinger, M. Feldman, R. Prabhakar, Y. Zhang, S. Hadjis, R. Fiszel, T. Zhao, L. Nardi, A. Pedram, C. Kozyrakis *et al.*, "Spatial: A language and compiler for application accelerators," in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2018, pp. 296–311.

[61] J. Hennessy and D. Patterson, "A new golden age for computer architecture: domain-specific hardware/software co-design, enhanced," in *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018.

[62] A. Krishnakumar, S. E. Arda, A. A. Goksoy, S. K. Mandal, U. Y. Ogras, A. L. Sartor, and R. Marculescu, "Runtime task scheduling using imitation learning for heterogeneous many-core systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 4064–4077, 2020.

[63] Y. S. Shao, S. L. Xi, V. Srinivasan, G.-Y. Wei, and D. Brooks, "Co-designing accelerators and soc interfaces using gem5-aladdin," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–12.

[64] S. K. Mandal, A. Krishnakumar, and U. Y. Ogras, "Energy-efficient networks-on-chip architectures: Design and run-time optimization," *Network-on-Chip Security and Privacy*, pp. 55–75, 2021.

[65] A. Stevens, "Quality of service (qos) in arm® systems: An overview," *ARM, Cambridge, UK, White Paper*, 2014.

[66] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Compute caches," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 481–492.

[67] D. Gajaria, K. A. Gomez, and T. Adegbija, “A study of stt-ram-based in-memory computing across the memory hierarchy,” in *2022 IEEE 40th International Conference on Computer Design (ICCD)*. IEEE, 2022, pp. 685–692.

[68] Y. Pan, P. Ouyang, Y. Zhao, W. Kang, S. Yin, Y. Zhang, W. Zhao, and S. Wei, “A multilevel cell stt-mram-based computing in-memory accelerator for binary convolutional neural network,” *IEEE Transactions on Magnetics*, vol. 54, no. 11, pp. 1–5, 2018.

[69] S. Resch, S. K. Khatamifard, Z. I. Chowdhury, M. Zabihi, Z. Zhao, J.-P. Wang, S. S. Sapatnekar, and U. R. Karpuzcu, “Pimball: Binary neural networks in spintronic memory,” *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 16, no. 4, pp. 1–26, 2019.

[70] H. Cai, Z. Bian, Z. Fan, B. Liu, and L. Naviner, “Commodity bit-cell sponsored mram interaction design for binary neural network,” *IEEE Transactions on Electron Devices*, vol. 69, no. 4, pp. 1721–1726, 2021.

[71] H. Cai, J. Chen, Y. Zhou, X. Hong, B. Liu, and L. A. de Barros Naviner, “Sparse realization in unreliable spin-transfer-torque ram for convolutional neural network,” *IEEE Transactions on Magnetics*, vol. 57, no. 2, pp. 1–5, 2020.

[72] S. Angizi, Z. He, A. Awad, and D. Fan, “Mrima: An mram-based in-memory accelerator,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1123–1136, 2019.

[73] D. Kim, Y. Jang, T. Kim, and J. Park, “Bimdim: Area efficient bi-directional mram digital in-memory computing,” in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 74–77.

[74] D. Kim and J. Park, “Distributed accumulation based energy efficient stt-mram based digital pim architecture,” in *2022 19th International SoC Design Conference (ISOCC)*. IEEE, 2022, pp. 29–30.

[75] H. Cılasun, S. Resch, Z. I. Chowdhury, E. Olson, M. Zabihi, Z. Zhao, T. Peterson, K. K. Parhi, J.-P. Wang, S. S. Sapatnekar *et al.*, “Spiking neural networks in spintronic computational ram,” *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 18, no. 4, pp. 1–21, 2021.

[76] M. Kang and J. Park, “Peripheral circuit optimization with pre-charge technique of spin transfer torque mram synapse array,” in *2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*. IEEE, 2021, pp. 1–3.

[77] V.-T. Nguyen, Q.-K. Trinh, R. Zhang, and Y. Nakashima, “Stt-bsnn: an in-memory deep binary spiking neural network based on stt-mram,” *IEEE Access*, vol. 9, pp. 151 373–151 385, 2021.

[78] A. Agrawal, A. Ankit, and K. Roy, “Spare: Spiking neural network acceleration using rom-embedded rams as in-memory-computation primitives,” *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1190–1200, 2018.

[79] S. Angizi, S. Tabrizchi, and A. Roohi, “Pisa: A binary-weight processing-in-sensor accelerator for edge image processing,” *arXiv preprint arXiv:2202.09035*, 2022.

[80] Z. He, S. Angizi, and D. Fan, “Exploring stt-mram based in-memory computing paradigm with application of image edge extraction,” in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 439–446.

[81] F. Parveen, Z. He, S. Angizi, and D. Fan, “Hielm: Highly flexible in-memory computing using stt mram,” in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2018, pp. 361–366.

[82] H. Cılasun, S. Resch, Z. I. Chowdhury, M. Zabihi, Z. Zhao, T. Peterson, J.-P. Wang, S. S. Sapatnekar, and U. R. Karpuzcu, “Seeds of seed: H-cram: In-memory homomorphic search accelerator using spintronic computational ram,” in *2021 International Symposium on Secure and Private Execution Environment Design (SEED)*. IEEE, 2021, pp. 70–75.

[83] Z.-H. Zhou, *Machine learning*. Springer Nature, 2021.

[84] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, “Binary convolutional neural network on rram,” in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 782–787.

[85] X. Lin, C. Zhao, and W. Pan, “Towards accurate binary convolutional neural network,” *Advances in neural information processing systems*, vol. 30, 2017.

[86] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, “Computing in memory with spin-transfer torque magnetic ram,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 470–483, 2017.

[87] S. Angizi, Z. He, F. Parveen, and D. Fan, “Imce: Energy-efficient bit-wise in-memory convolution engine for deep neural network,” in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2018, pp. 111–116.

[88] Z. Chowdhury, J. D. Harms, S. K. Khatamifard, M. Zabihi, Y. Lv, A. P. Lyle, S. S. Sapatnekar, U. R. Karpuzcu, and J.-P. Wang, “Efficient in-memory processing using spintronics,” *IEEE Computer Architecture Letters*, vol. 17, no. 1, pp. 42–46, 2017.

[89] T.-N. Pham, Q.-K. Trinh, I.-J. Chang, and M. Alioto, “Stt-bnn: A novel stt-mram in-memory computing macro for binary neural networks,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 569–579, 2022.

[90] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, “Recent advances in convolutional neural networks,” *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[91] M. Zabihi, Z. I. Chowdhury, Z. Zhao, U. R. Karpuzcu, J.-P. Wang, and S. S. Sapatnekar, “In-memory processing on the spintronic cram: From hardware design to application mapping,” *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1159–1173, 2018.

[92] V. Akhlaghi, A. Yazdanbakhsh, K. Samadi, R. K. Gupta, and H. Esmaeilzadeh, “Snapea: Predictive early activation for reducing computation in deep convolutional neural networks,” in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 662–673.

[93] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, “Deep learning in spiking neural networks,” *Neural networks*, vol. 111, pp. 47–63, 2019.

[94] W. Maass, “Networks of spiking neurons: the third generation of neural network models,” *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[95] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[96] J.-P. Wang and J. D. Harms, “General structure for computational random access memory (cram),” Dec. 29 2015, uS Patent 9,224,447.

[97] Z. Abid, A. Alma’Aitah, M. Barua, and W. Wang, “Efficient cmol gate designs for cryptography applications,” *IEEE transactions on nanotechnology*, vol. 8, no. 3, pp. 315–321, 2009.

[98] A. J. L. Martin, “Cadence design environment,” *New Mexico State University, Tutorial paper*, p. 35, 2002.

[99] Y. A. Belay, A. Cabrini, and G. Torelli, “A comprehensive verilog-a behavioral model of spin-transfer torque memory cell,” in *2016 12th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME)*. IEEE, 2016, pp. 1–4.

[100] R. Garg, D. Kumar, N. Jindal, N. Negi, and C. Ahuja, “Behavioural model of spin torque transfer magnetic tunnel junction, using verilog-a,” *International Journal of Advanced Scientific and Technical Research*, vol. 1, no. 6, 2012.

[101] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, “The gem5 simulator,” *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[102] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, B. Beckmann, S. Bharadwaj *et al.*, “The gem5 simulator: Version 20.0+,” *arXiv preprint arXiv:2007.03152*, 2020.

[103] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, “Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures,” in *Proceedings of the 42nd annual ieee/acm international symposium on microarchitecture*, 2009, pp. 469–480.

[104] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, “Architecting efficient interconnects for large caches with cacti 6.0,” *IEEE micro*, vol. 28, no. 1, pp. 69–79, 2008.

[105] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, “Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, 2012.

[106] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams *et al.*, “The landscape of parallel computing research: A view from berkeley,” 2006.

[107] D. Green *et al.*, “Heterogeneous integration at darpa: Pathfinding and progress in assembly approaches,” *ECTC, May*, 2018.

[108] J. Ferrante, K. J. Ottenstein, and J. D. Warren, “The program dependence graph and its use in optimization,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 9, no. 3, pp. 319–349, 1987.

[109] R. Uhrie, D. W. Bliss, C. Chakrabarti, U. Y. Ogras, and J. Brunhaver, "Machine understanding of domain computation for domain-specific system-on-chips (dssoc)," in *Open Architecture/Open Business Model Net-Centric Systems and Defense Transformation 2019*, vol. 11015. SPIE, 2019, pp. 180–187.

[110] B. Boroujerdian, Y. Jing, D. Tripathy, A. Kumar, L. Subramanian, L. Yen, V. Lee, V. Venkatesan, A. Jindal, R. Shearer *et al.*, "Farsi: An early-stage design space exploration framework to tame the domain-specific system-on-chip complexity," *ACM Transactions on Embedded Computing Systems (TECS)*, 2022.

[111] S. E. Arda, A. Krishnakumar, A. A. Goksoy, N. Kumbhare, J. Mack, A. L. Sartor, A. Akoglu, R. Marculescu, and U. Y. Ogras, "Ds3: A system-level domain-specific system-on-chip simulation framework," *IEEE Transactions on Computers*, vol. 69, no. 8, pp. 1248–1262, 2020.

[112] P. Bose, A. Vega, S. Adve, V. Adve, S. Misailovic, L. Carloni, K. Shepard, D. Brooks, V. J. Reddi, and G.-Y. Wei, "Secure and resilient socs for autonomous vehicles," in *Proceedings of the 3rd International Workshop on Domain Specific System Architecture (DOSSA)*, 2021, pp. 1–6.

[113] H. Lategahn, A. Geiger, and B. Kitt, "Visual slam for autonomous ground vehicles," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1732–1737.

[114] A. Vega, A. Amarnath, J.-D. Wellman, H. Kassa, S. Pal, H. Franke, A. Buyuktosunoglu, R. Dreslinski, and P. Bose, "Stomp: A tool for evaluation of scheduling policies in heterogeneous multi-processors," *arXiv preprint arXiv:2007.14371*, 2020.

[115] Y. Xiao, S. Nazarian, and P. Bogdan, "Self-optimizing and self-programming computing systems: A combined compiler, complex networks, and machine learning approach," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 27, no. 6, pp. 1416–1427, 2019.

[116] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM computing surveys (CSUR)*, vol. 37, no. 4, pp. 316–344, 2005.

[117] K. J. Brown, A. K. Sujeeth, H. J. Lee, T. Rompf, H. Chafi, M. Odersky, and K. Olukotun, "A heterogeneous parallel framework for domain-specific languages," in *2011 International Conference on Parallel Architectures and Compilation Techniques*. IEEE, 2011, pp. 89–100.

[118] T. Neumann, "Efficiently compiling efficient query plans for modern hardware," *Proceedings of the VLDB Endowment*, vol. 4, no. 9, pp. 539–550, 2011.

[119] M. Kotsifakou, P. Srivastava, M. D. Sinclair, R. Komuravelli, V. Adve, and S. Adve, "Hpvm: Heterogeneous parallel virtual machine," in *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2018, pp. 68–80.

[120] L. Suriano, D. Madroñal, A. Rodríguez, E. Juárez, C. Sanz, and E. de la Torre, "A unified hardware/software monitoring method for reconfigurable computing architectures using papi," in *2018 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 2018, pp. 1–8.

[121] J. Mack, S. Hassan, N. Kumbhare, M. Castro Gonzalez, and A. Akoglu, "Cedr: A compiler-integrated, extensible dssoc runtime," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 2, pp. 1–34, 2023.

[122] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, "Starpu: a unified platform for task scheduling on heterogeneous multicore architectures," in *Euro-Par 2009 Parallel Processing: 15th International Euro-Par Conference, Delft, The Netherlands, August 25-28, 2009. Proceedings 15*. Springer, 2009, pp. 863–874.

ALABA YUSUF received his B.S. and M.S. degrees in Electrical Engineering from the University of Alabama at Birmingham, in 2012 and Gannon University in Erie, PA in 2015 respectively. He is a Ph.D. student in the Department of Electrical and Computer Engineering at the University of Arizona.

Mr. Yusuf is a Principal Electrical Engineer working with the Flight Termination Systems group at Raytheon Technologies. His interests include computer architecture and domain-specific architectures.

TOSIRON ADEGBIJA received his M.S. and Ph.D. in Electrical and Computer Engineering from the University of Florida in 2011 and 2015, respectively, and his B.Eng in Electrical Engineering from the University of Ilorin, Nigeria in 2005.

Dr. Adegbija is currently an Associate Professor of Electrical and Computer Engineering at the University of Arizona, USA. His research interests are in computer architecture, with an emphasis on brain-inspired computing, adaptable computing, low-power embedded systems design and optimization methodologies, and domain-specific architectures. He received the CAREER Award from the National Science Foundation in 2019. He is a Senior Member of the IEEE.

DHRUV GAJARIA received his M.S. and Ph.D. in Electrical and Computer Engineering from the University of Arizona in 2019 and 2023, respectively, and his B.Eng in Electronics Engineering from the University of Mumbai, India, in 2017.

Dr. Gajaria is a Post-Doctoral Research Associate with the High-Performance Computing Group at Pacific Northwest National Lab, USA. His research interests include hardware-software co-design, computer architecture, simulation and performance analysis, and domain-specific architectures.