frontiers | Frontiers in Science

TYPE Frontiers in Science Lead Article
PUBLISHED 16 December 2025
DOI 10.3389/fsci.2025.1611658

OPEN ACCESS

EDITED BY
Saptarshi Das,
The Pennsylvania State University (PSU), United States

REVIEWED BY
Khaled Nabil Salama,
King Abdullah University of Science and Technology, Saudi Arabia
John Paul Strachan,
Forschungszentrum Jülich GmbH, Germany

*CORRESPONDENCE
Kaushik Roy
kaushik@purdue.edu

RECEIVED 14 April 2025
REVISED 31 July 2025
ACCEPTED 13 November 2025
PUBLISHED 16 December 2025

CITATION
Roy K, Kosta A, Sharma T, Negi S, Sharma D, Saxena U, Roy S, Raghunathan A, Wan Z, Spetalnick S, Liu C-K and Raychowdhury A. Breaking the memory wall: next-generation artificial intelligence hardware. *Front Sci* (2025) 3:1611658. doi: 10.3389/fsci.2025.1611658

Breaking the memory wall: next-generation artificial intelligence hardware

Kaushik Roy[1]*, Adarsh Kosta[1], Tanvi Sharma[1], Shubham Negi[1], Deepika Sharma[1], Utkarsh Saxena[1], Sourjya Roy[1], Anand Raghunathan[1], Zishen Wan[2], Samuel Spetalnick[2], Che-Kai Liu[2] and Arijit Raychowdhury[2]

[1]Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, United States, [2]School of Electrical and Computer Engineering, College of Engineering, Georgia Institute of Technology, Atlanta, GA, United States

Abstract

The relentless advancement of artificial intelligence (AI) across sectors such as healthcare, the automotive industry, and social media necessitates the development of more efficient hardware solutions that can implement diverse learning algorithms. This lead article explores the evolution of AI learning algorithms and their computational demands, using autonomous drone navigation as a case study to highlight the limitations of traditional hardware. Traditional hardware, based on the von Neumann architecture, suffers from limited computational efficiency due to the separation of compute units and memory, also known as the "memory wall" problem. To overcome this barrier, this article discusses novel approaches to AI hardware design, focusing on compute-in-memory (CIM) techniques and stochastic hardware. CIM offers a promising solution to the memory wall problem by integrating computing capabilities directly into the memory system. This article details state-of-the-art developments in CIM for different memory types and at various levels of the memory hierarchy to support essential AI compute functions. We also discuss the use of CIM in developing neuromorphic hardware capable of accelerating biologically inspired algorithms, such as spiking neural networks. Furthermore, we highlight how stochastic hardware can exploit the error resilience of AI algorithms to enhance energy efficiency. Encompassing the full stack of AI systems, from learning algorithms to circuit and device-level techniques and architectures, this article provides a comprehensive roadmap for future research and development in AI hardware.

KEYWORDS

artificial intelligence, neural network acceleration hardware, memory wall, spiking neural networks, hardware algorithm co-design, compute-in-memory, approximate computing

Key points

- Efficient artificial intelligence (AI) hardware is crucial for resource-constrained applications such as healthcare and transportation, where it enhances performance, reduces costs, and supports real-time decision-making.
- Overcoming the memory wall in traditional hardware is critical for enhancing AI computational efficiency, reducing latency, and enabling faster, more effective processing of complex algorithms.
- Compute-in-memory paradigms using different memory technologies, such as embedded non-volatile memory, static random-access memory, dynamic random-access memory, and flash memory, help develop energy-efficient AI hardware by tackling the memory wall problem.
- Stochasticity in AI algorithms (e.g., via spike timing-dependent plasticity or STDP) and hardware (e.g., via spin–orbit transfer magnetic tunnel junctions or SOT-MTJs) can be leveraged to improve energy efficiency for diverse workloads and could unlock novel capabilities.
- Co-designing hardware and algorithms to optimize energy, latency, and accuracy will lead to the development of a "converged platform" for artificial neural networks and spiking neural networks, suitable for diverse AI applications.

Introduction

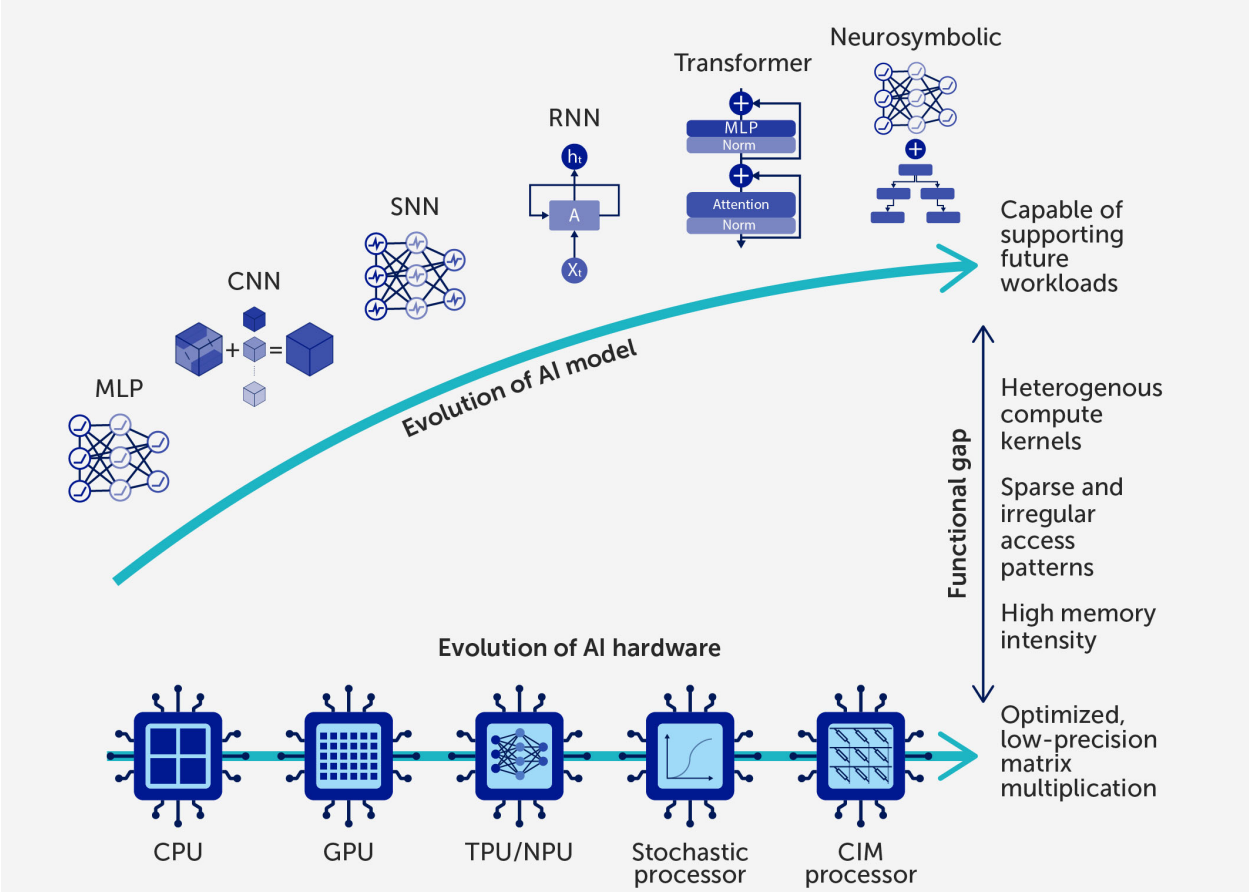
Artificial intelligence (AI) has emerged as one of the most transformative technologies of the 21st century, reshaping numerous aspects of everyday life. Driven by the overarching goal of replicating human intelligence, developers have created multiple generations of AI algorithms. Machine learning (ML) algorithms are particularly notable, as they draw inspiration from human brain functions, enabling computers to learn and generalize from input data. Recent advances in ML aspire to equip computers with cognition, perception, and reasoning abilities that potentially match or exceed those of humans (1–9).

In 1989, LeCun and colleagues at Bell Labs made a significant breakthrough by training a neural network to classify handwritten digits (10). Since then, the development of neural network training algorithms has continued to evolve, resulting in architectures such as multi-layer perceptrons (MLPs), convolutional neural networks (CNNs) (1), long short-term memory models (LSTMs) (2), and transformers (3). This evolution has led to an explosion in both the parameters and computational demands of these models, culminating in bottlenecks during training on traditional central processing units (CPUs). In a pivotal move, in 2012, graphics processing units (GPUs) began to be utilized for their superior parallelism capabilities, specifically for efficient matrix-vector multiplication (MVM), the primary computation in neural networks (1). This, along with advances in transistor technology, has facilitated the training of multi-billion parameter networks using extensive server farms within data centers.

With the development of large language and vision models (LLMs and LVMs) based on the transformer architecture (3), AI has excelled in applications such as language translation (4), text prediction/generation (5), text-to-image synthesis (6), and image generation (7). However, these multi-billion parameter models require significant computational and energy resources, raising concerns about their sustainability in real-world scenarios. Moreover, they are less efficient than the human brain owing to their dense, synchronous, and high-precision computations and the need for extensive data movement between compute and memory units. To address these challenges, efforts have been made to achieve brain-like computation, such as spiking neural networks (SNNs), which can perform sparse and event-driven computations similar to the brain (11). Nevertheless, efficient implementation of fundamental operations required by ML workloads cannot be realized without considering optimization at the underlying hardware level.

Today's AI hardware solutions are rooted in the von Neumann architecture, which distinctly separates computational and memory units. It has become evident that shuttling data between memory and compute units, known as the "memory wall" problem, is responsible for the majority of the energy consumption and latency in computing systems. Rethinking devices, circuits, and computing architectures to incorporate a "compute-in-memory" (CIM) paradigm, by performing compute operations within the memory array itself, holds significant promise for alleviating this issue (12). Concurrently, the error-resiliency of AI algorithms can be utilized to develop hardware that is approximate but enables faster operations with reduced energy consumption while maintaining system-level accuracy (13). As AI applications become more integrated into our daily lives, designing brain-inspired AI algorithms and hardware necessitates a mutual consideration of constraints and requirements, promoting an algorithm-hardware co-design approach. Figure 1 illustrates the evolution of AI models and hardware over time, highlighting the functional gap relative to the brain.

Existing surveys on autonomous navigation have provided comprehensive overviews of AI methods in the field. A notable survey by Nahavandi et al. (14) offers an in-depth examination of autonomous navigation for mobile robots, covering standard sensing technologies, a variety of robotic platforms, simulation environments, and navigation fundamentals and presenting a thorough treatment of Simultaneous Localization and Mapping (SLAM) algorithms. Separately, another work by Rezwan et al. (15) focuses specifically on unmanned aerial vehicle (UAV) autonomous navigation, comparing traditional mathematical optimization approaches with recent learning-based methods, and delves into UAV-specific considerations such as navigation models and input modalities. In contrast, this article emphasizes the evolution of AI applications and advances in AI algorithms to emphasize the requirements for specialized AI hardware. Using autonomous drone navigation as an illustrative application, it contextualizes the



Abbreviations: **AI**, artificial intelligence; **CIM**, compute-in-memory; **CNN**, convolutional neural network; **CPU**, central processing unit; **GPU**, graphic processing unit; **MLP**, multi-layer perceptron; **NPU**, neural processing unit; **RNN**, recurrent neural network; **SNN**, spiking neural network; **TPU**, tensor processing unit

FIGURE 1
Evolution of artificial intelligence (AI) models and hardware over time. With the increasing complexity and demands of AI workloads, AI models also expand in size and complexity to maintain performance, resulting in high energy and latency implementations. To address this, advancements in AI hardware aim to offer novel solutions that are fast and efficient. However, this evolution in hardware also leads to a widening functional gap between the AI models and the hardware. The gap can be characterized by factors such as a variety of computational kernels, unpredictable and sparse access patterns, and high memory density requirements. This motivates the need to co-design AI algorithms and AI hardware in a converged platform across the entire stack, from devices and circuits to architecture and algorithms.

discussion on efficient AI hardware design, directly linking algorithmic demands to hardware innovation.

On the hardware side, previous reviews (16) have primarily explored the material properties of devices employed in CIM crossbars without addressing the broader accelerator architecture or issues of application-level integration. Furthermore, in contrast to other works (17, 18), this lead article additionally surveys emerging low-power neural network architectures, such as spiking neural networks (SNNs), which are highly relevant for energy-constrained edge platforms like drones. Ultimately, this article explores new computing paradigms—namely, in-memory and approximate computing—that offer the potential for substantial performance enhancements over existing AI hardware. It also explores neuromorphic hardware for SNNs, highlighting the advancements necessary in hardware technologies such as CIM and stochastic devices. The discussion concludes with an examination of brain-inspired solutions as promising avenues for achieving efficient AI implementations.

Evolution of AI applications and algorithms

AI systems of today are incredibly versatile, capable of both simple tasks, such as voice-activated lighting, and complex ones, such as generating realistic videos of imagined scenes. This section explores the historical as well as ongoing development of AI applications and the algorithms that power them. We focus on autonomous drone navigation as an exemplary application to demonstrate AI's potential and examine the learning algorithms best suited for fast and efficient deployment on edge devices.

AI applications

The cognitive capabilities of AI have recently experienced a significant leap, leading to unprecedented performance across various domains. Notably, these include computer vision (encompassing image and video analysis, captioning, denoising, and inpainting) and natural language processing (including language translation, text summarization, and chatbots). As a result, AI has been applied in a variety of industries, such as healthcare, finance, transportation, retail, and manufacturing. The dynamic landscape of AI applications is continually evolving as more tasks valuable to human society are consistently enhanced by AI.

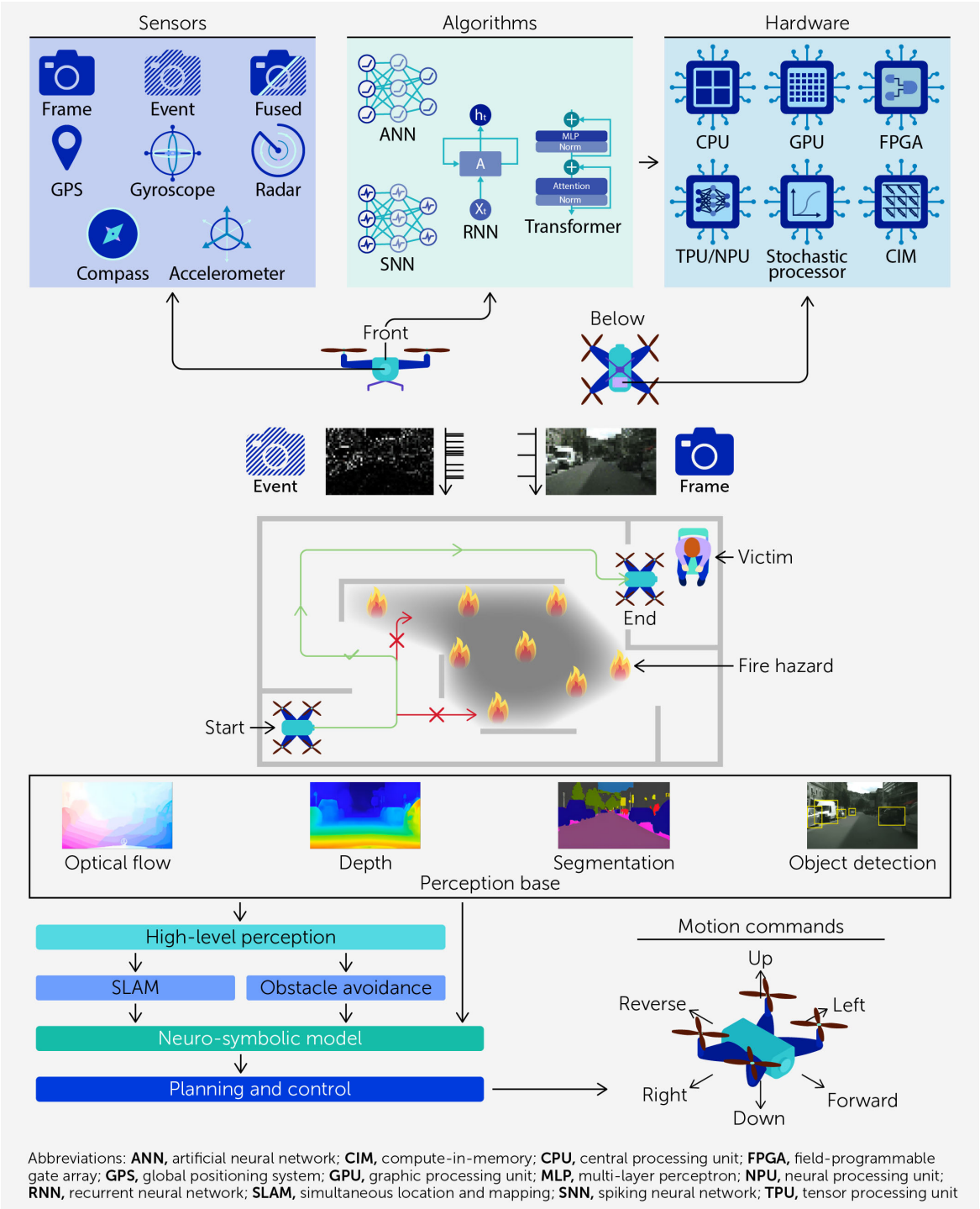
In the current era, dominated by data, acquiring relevant information is essential for the effective operation of AI systems. This information usually comes from a vast corpus of language, image, and speech data on the Internet. However, it can also be derived in real-time, for example, from the activity history of end-users on various ML-backed social media platforms or directly from the physical world through an array of sensors, including cameras, global positioning system (GPS), radar, lidar, sonar, and inertial measurement units (IMUs). Among these, vision sensors are the most prominent across various real-world applications. In recent years, a new category of vision sensors known as event-based cameras has gained significant attention (19, 20). Unlike traditional frame-based cameras that synchronously sample dense frames, event-based cameras provide an asynchronous and sparse stream of binary events based on the pixel-wise change in log-scale intensity. This leads to better temporal resolution, dynamic range, and power consumption (21). These characteristics position event cameras as promising candidates for low-power and low-latency sensing elements in resource-constrained systems.

Consider an exemplary application of a UAV deployed in a hazardous environment for autonomous search and rescue (Figure 2) (22). The UAV is required to navigate seamlessly at high speed, plan, reason, and make decisions without any human supervision. To accomplish this, it needs to have a detailed understanding of its environment by carrying out several underlying perception tasks—such as optical flow, depth estimation, semantic segmentation, and object detection—to construct a perception base. This perception base is further used by high-level perception modules, planning subsystems, and neurosymbolic models to appropriately control the behavior of the drone (23, 24). This entire pipeline needs to be executed in real-time to enable high-speed navigation under all environmental conditions, which is quite challenging. From the algorithm standpoint, first, the architectures must be small enough to satisfy the resource constraints at the edge yet powerful enough to achieve satisfactory performance (25). Second, when considering real-world deployment for tasks such as optical flow and object tracking, it is crucial that the underlying algorithms and architectures are capable of effectively capturing the temporal dependencies in inputs over time. Third, since many of the subtasks are interdependent, where the output of one task serves as the input for another, it is important to carefully consider the design of the overall architecture and optimize it for efficient implementation. On the hardware front, optimized implementations capable of accelerating algorithms with these characteristics are vital, necessitating a shift from traditional CPU/GPU-based approaches to application-specific hardware.

AI deployment

AI models are trained using extensive real-world data in centralized data centers, often employing clusters of GPUs before they are deployed for repeated inference in the real world. AI models can be deployed either at the edge or in the cloud. Deploying AI models in the cloud has become a common strategy, primarily due to the unparalleled application accuracy achievable with the use of large, complex models. While practical for creating AI applications, this approach faces several challenges, such as the requirement for an active Internet connection, privacy concerns when transmitting sensitive data, and high energy and latency costs for extensive back-and-forth communications. Conversely, edge AI involves deploying AI algorithms entirely on edge systems, eliminating the need for communication with the cloud. However, due to their physical constraints, limited power budget, and computing capabilities, edge AI models must be scaled down, inevitably resulting in performance loss (26). As edge computing becomes more widespread, it is crucial to develop both algorithms and hardware together, with the goal of creating AI platforms that are both fast and efficient.

To facilitate the efficient deployment of AI models, quantization and sparsity have emerged as highly effective techniques, leading to models with substantially reduced size and computational demands. Quantization reduces the numerical precision of model parameters and activations from high-precision floating-point formats (e.g., FP32) to compact representations such as 8-bit integer (INT8), 4-bit integer, or even binary. Historically, early neural networks relied on handcrafted fixed-point arithmetic for embedded inference; however, modern quantization extends this principle with sophisticated calibration, error correction, and mixed-precision strategies (27, 28) that preserve accuracy at extreme compression ratios. Techniques like dynamic per-channel scaling, learned quantization parameters, and low-rank residual correction (29) demonstrate that quantization is no longer a heuristic post-processing step but an integral part of model design and training. By aligning model precision with hardware arithmetic capabilities, quantization enables dense compute units, such as tensor cores and systolic arrays, to operate near their theoretical throughput limits while dramatically lowering memory bandwidth and energy costs. Sparsity, on the other hand, exploits the empirical observation that many weights and activations in neural networks are redundant or contribute minimally to output quality (30). Structured sparsity (e.g., block or channel pruning) allows direct hardware acceleration, while unstructured sparsity achieves finer-grained compression through pruning and re-parameterization (31). Lately, hardware systems have added support for fine-grained unstructured sparsity (32). The introduction of sparse attention mechanisms and mixture-of-experts models reflects a paradigm shift and computation is no



Abbreviations: **ANN,** artificial neural network; **CIM,** compute-in-memory; **CPU,** central processing unit; **FPGA,** field-programmable gate array; **GPS,** global positioning system; **GPU,** graphic processing unit; **MLP,** multi-layer perceptron; **NPU,** neural processing unit; **RNN,** recurrent neural network; **SLAM,** simultaneous location and mapping; **SNN,** spiking neural network; **TPU,** tensor processing unit

FIGURE 2
An exemplary application involving a search-and-rescue mission carried out by an unmanned aerial vehicle (UAV). The UAV is equipped with a variety of sensors for accurate and rapid sensing, and it also employs efficient AI models on suitable brain-inspired hardware for perception, planning, and symbolic tasks. Given these capabilities, the UAV can navigate seamlessly in a previously unseen environment by determining an optimal motion path, while avoiding obstacles and hazardous areas.

longer uniformly distributed but dynamically allocated to the most informative aspects of inputs. Sparsity thus transforms inference from a static to an adaptive process, allowing AI systems to scale capacity without linearly scaling cost.

When viewed through the lens of AI evolution, quantization and sparsity represent complementary pathways toward efficiency, minimizing both the representation and computation costs. As AI progresses toward neuromorphic and analog paradigms, these

principles extend beyond digital optimization to form the foundation of energy-aware intelligence, where precision, density, and selectivity co-evolve as intrinsic properties of learning systems.

Learning algorithms: UAV navigation

Given the challenges in autonomous UAV navigation (Figure 2), and drawing parallels with efficient biological systems, such as the fruit fly (33), it is logical to seek inspiration from the brain. Tasks such as optical flow estimation require that we determine the movement of pixel intensities over time, whole motion segmentation necessitates classifying pixels into object categories in an image sequence, and object tracking involves identifying and tracking a moving object. These tasks are inherently sequential. Thus, the underlying AI algorithm must possess the ability to learn temporal dependencies between successive inputs.

Taking optical flow as an example, traditional AI models such as CNNs (1, 34) prove inherently unsuitable owing to their inability to capture temporal dependencies. Architectures integrating memory, such as recurrent neural networks (RNNs) (35) and LSTM networks (2), are more fitting but suffer from heightened network complexity and an intricate training process. These shortcomings arise because these systems do not encompass the fundamental working principles of the brain, which operates in a sparse and event-driven manner, seamlessly integrating compute and memory within the same physical substrate.

In contrast, advancements in neuroscience have led to the development of bio-plausible algorithms such as SNNs, which can efficiently process sequential data through sparse and event-driven computations, similar to the brain. SNNs represent a computationally simpler alternative to RNNs or LSTMs, utilizing a unique mechanism—membrane potential—which serves as a lightweight form of internal memory (11). In SNNs, inputs to each neural network layer are spikes (0 or 1) over time, necessitating only an accumulation operation, unlike the multiply-and-accumulate operation in artificial neural networks (ANNs) (36). However, deep SNNs also face challenges such as vanishing spikes and non-differentiable activations (37, 38), making training difficult. Fortunately, there have been several successful efforts toward developing techniques such as ANN-to-SNN conversion (37, 39), learnable neuronal dynamics (36, 40), and surrogate gradient learning (38, 41), which have simplified SNN training and improved their application performance. Furthermore, SNNs excel at processing data from previously discussed asynchronous sensors (event-based cameras).

Event-based optical flow estimation is typically carried out using an encoder-decoder multi-scale architecture based on U-Net, as introduced by Ronneberger et al. (42). Fully ANN models, inspired by Zhu et al. (43, 44), serve as the baselines and require representation of the event bins in the channel dimension (Figure 3A). This approach discards any temporal dependence between input events, resulting in suboptimal performance. In comparison, fully SNN models, such as Adaptive-SpikeNet (40) (Figure 3B), can capture temporal information effectively while utilizing layer-wise learnable neuronal dynamics to mitigate the challenges associated with SNN training. This results in improved application performance (~20% lower error) with the same model size or extremely lightweight (0.27-M parameter compared with 13 M) and efficient (~10× lower energy) models with similar performance, underscoring the efficacy of SNNs over ANNs in capturing temporal dynamics.

In parallel, there have been efforts to explore hybrid SNN–ANN models to simplify training. Works such as Spike-FlowNet (45) and SSLN (47) fall into this category. Spike-FlowNet (Figure 3C) consists of just an SNN-encoder and outperforms the fully ANN approach (43), offering 1.21× lower energy consumption. There have also been efforts to combine information from frame- and event-based cameras into a sensor-fusion model. Works such as Fusion-FlowNet (46) (Figure 3D) use events over time as inputs to an SNN encoder and grayscale frames over channels as inputs to an ANN-encoder. This enables superior feature extraction, leading to significantly improved performance and smaller model sizes. In fact, Fusion-FlowNet, with 7.55 million parameters, attains 40% lower error with a 1.87× energy reduction compared to fully ANN methods (43). Along similar lines, DOTIE (48) offers a lightweight object detection pipeline, and HALSIE (49) uses sensor fusion for semantic segmentation. Although explained for specific tasks, the discussed principles can be applied to a broad spectrum of perception tasks. In the context of UAV navigation applications, all these approaches constitute notable advancements toward achieving brain-like edge intelligence.

While the above works demonstrate the unique potential of SNNs and hybrid architectures, it is important to note that these advantages are domain specific and should not be generalized. SNNs work well with tasks involving temporal information, and event-based sensors work well when sparse binary information is sufficient for carrying out the task at hand. In contrast, applications such as face recognition, image classification, image analysis, and 3D reconstruction, which are data intensive, are still dominated by ANNs.

In line with recent advancements in AI, including LLMs/LVMs (3), diffusion models (7), and neural architecture search (NAS) (50), which are much more powerful and power hungry than previously discussed traditional approaches, there have been few initiatives toward low-cost implementations such as SNN transformers (51, 52). However, the energy efficiency of these initiatives remains suboptimal. This underscores the need for innovative approaches in developing brain-inspired architectures from the ground up, effectively utilizing the unique temporal dimension of SNNs rather than simply replicating ANN-based models.

Even with meticulously designed and highly optimized algorithms, achieving efficiency at the hardware level remains challenging. Although traditional hardware is advancing, a significant gap persists between AI hardware capabilities and the rigorous demands of AI applications across various domains. Beyond standard operations such as matrix-vector multiplications (MVMs) and transcendental functions, brain-inspired workloads require operations like frequent fetching and updating of membrane potential and modeling neuronal dynamics. These

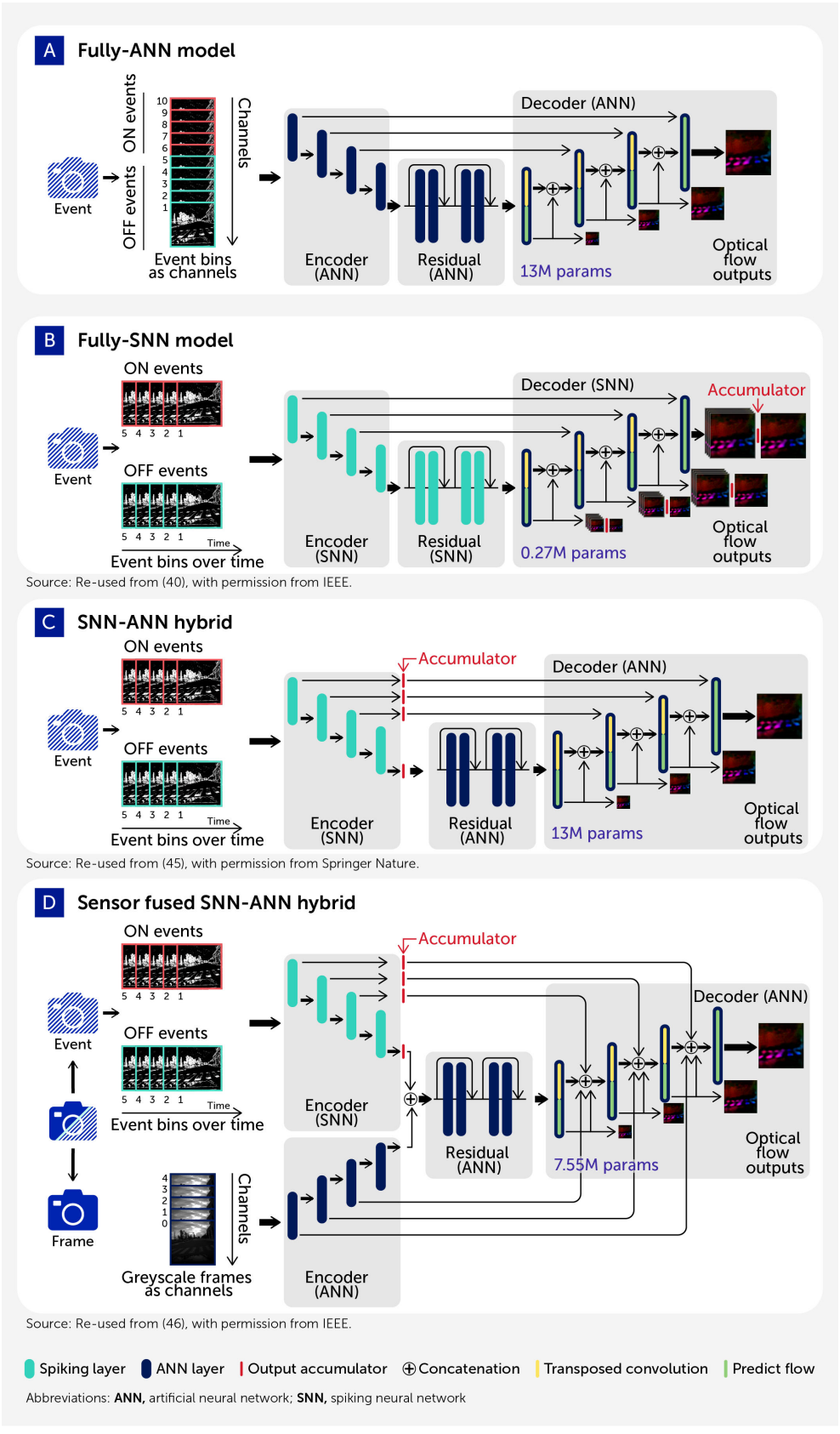


FIGURE 3
Architectures for optical flow estimation based on U-Net from Ronneberger et al. (42). **(A)** Fully artificial neural network (ANN) architecture operating on event bins over channels. **(B)** Fully spiking neural network (SNN) architecture—Adaptive-SpikeNet—operating on events over time with output spikes accumulated at the last decoder layer. Re-used from (40), with permission from IEEE. **(C)** Hybrid SNN–ANN architecture—Spike-FlowNet—operating on events over time. The output spikes are accumulated at the SNN-encoder. Re-used from (45), with permission from Springer Nature, **(D)** Sensor-fusion architecture—Fusion-FlowNet— utilizing data from events over time and grayscale frames over channels. Re-used from (46), with permission from IEEE.

demands present significant challenges for von Neumann architectures and underscore the need to explore CIM approaches to provide an energy-efficient, converged platform for both SNN and ANN-based AI algorithms. The remainder of this article explores potential hardware solutions at various levels, from specialized CIM architectures and digital/analog memory technologies to stochastic hardware and emerging devices.

Compute in/near memory for efficient AI hardware design

The concept of CIM dates back to the 1990s (53, 54). However, with the advancements in AI applications, the limitations of traditional computing architectures have become increasingly apparent. Traditional systems, based on the von Neumann architecture, face the memory wall challenge (55), where the separation between processing and memory units leads to significant data movement overheads. As an alternative, CIM paradigms have been proposed to reduce the higher cost of memory accesses (56–58).

CIM, also known as processing-in-memory (PIM), fundamentally alters the computing paradigm by bringing computations closer to, or inside, synaptic memory where data reside, effectively addressing the memory wall challenge. CIM architectures can perform massively parallel multiply-and-accumulate (MAC) operations on inputs and synaptic weights, the predominant operation in AI models. Additionally, CIM technology shows promise for efficiently executing transcendental functions such as exponential, logarithmic, or trigonometric calculations. It is also particularly advantageous for SNN algorithms by significantly reducing the costs associated with storing and fetching neuronal membrane potentials.

Figure 4A illustrates an example of a CIM accelerator with its spatial architecture (59). This architecture consists of multiple tiles connected via a network on chip (NoC), each containing *N* CIM cores. A CIM core (Figure 4B) comprises CIM arrays, referred to as matrix-vector multiplication units (MVMUs), along with other functional units and an instruction execution pipeline (Fetch, Decode, Execute, and Memory). Following the fetching of an instruction from the instruction memory, it is decoded and executed in the appropriate functional unit, which could be the scalar functional unit (SFU), vector functional unit (VFU), or the pipelined MVMU. The output is stored in the memory unit (MU), which facilitates communication with other cores. Hence, CIM arrays provide a framework for designing energy-efficient hardware tailored for AI applications.

CIM can be integrated into memory through different methods, also known as computing schemes. These schemes are broadly categorized into analog and digital, depending on whether the values computed in the memory array are continuous or discrete, respectively. Additionally, the benefits of CIM vary with the memory technology based on their read–write energy, area, latency, and other specifications. The following subsection describes various types of CIM, particularly focusing on analog and digital CIM computing schemes, and reviews the differences in various memory types. The subsequent subsection details the advancements in CIM designs for different memory technologies across a memory hierarchy.

Types of CIM

Computing schemes

CIM can be broadly categorized into analog (or mixed-signal) and digital (binary) types. Figure 4D depicts an example of analog CIM (60), for performing the matrix-vector multiplication (MVM) operation.

Several methods are employed to perform such *in situ* MAC operations in an analog manner. The technique shown in Figure 4D involves current-mode computing, which takes advantage of the property of linear addition of currents in circuits, namely, Kirchhoff's current law. Mathematically, this can be represented as Equation 1 follows:

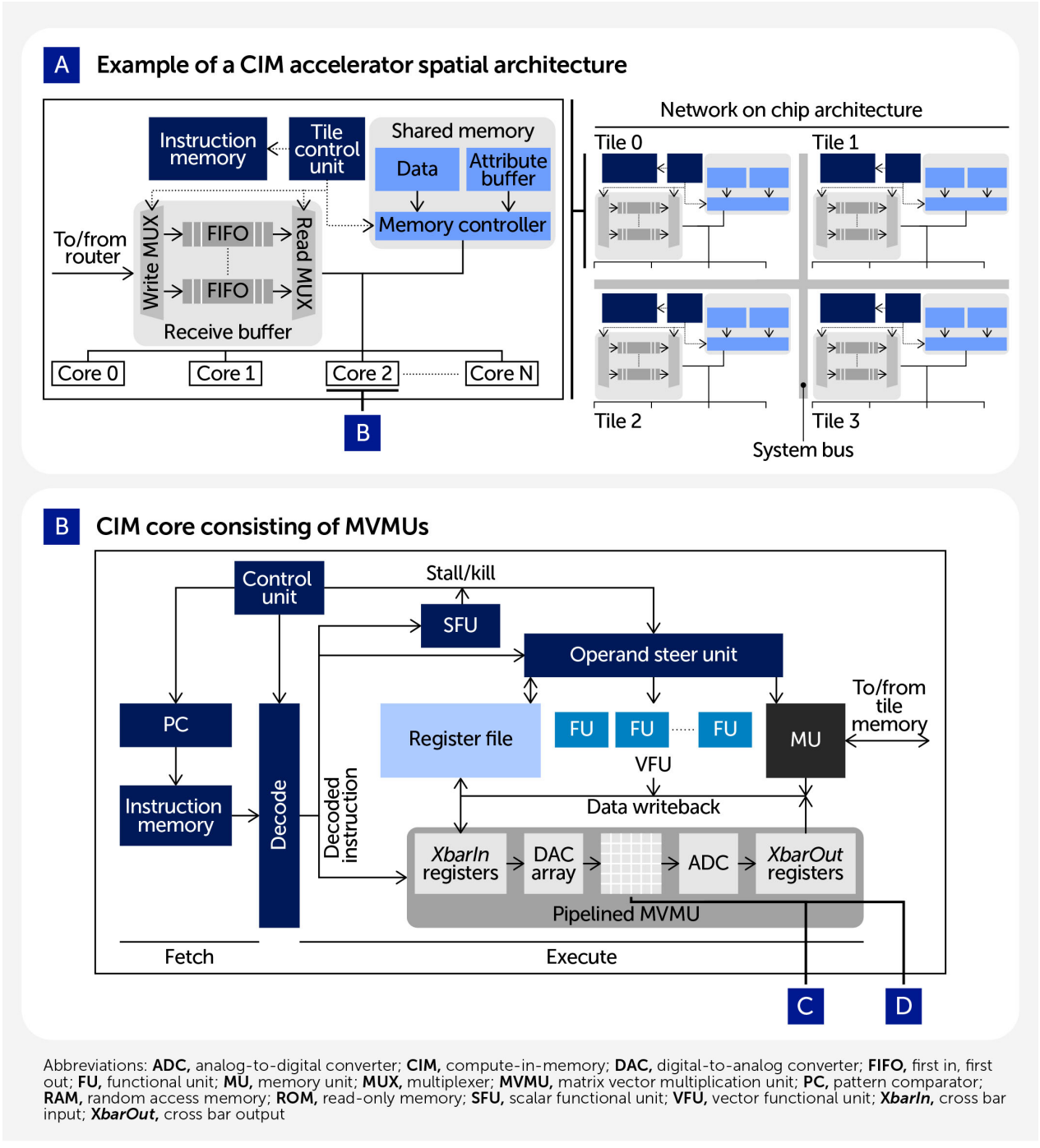
$$I_{BL,i} = \sum_{j=0}^{N} V_j \times G_{i,j} \tag{1}$$

The encoded inputs are supplied to each memory row through the source of the access device, contributing to V_j. The resistance of each cell (or its inverse, the conductance G_{ij}) corresponds to the weights of the neural network and can be single or multi-bit, depending on the bit capacity of the memory cell. The choice of the memory cell in a CIM array could be embedded non-volatile memory (eNVM) devices or any type of static random-access memory (SRAM) cell (Figure 4D). By activating multiple wordlines (WLs) simultaneously in an array, the current from each row accumulates at the bitline (BL) ($I_{BL,i}$) to produce the MAC output between inputs and weights. The analog current is then converted to the voltage domain via a transimpedance amplifier (TIA) and finally to the digital domain. Another method is charge-based computing (61), which exploits the capacitive properties of memory cells to accumulate charges, thereby obtaining the MAC output.

The analog nature of communication is prone to errors; therefore, the MAC output is converted to the digital domain via an analog-to-digital converter (ADC) to facilitate noise-resilient and robust communication between different CIM cores. Similarly, inputs arrive in digital form and are encoded using a digital-to-analog converter (DAC) to convert the bit-streamed inputs into the voltage domain. However, ADCs often introduce significant area, power, and performance overheads in mixed-signal CIM systems, as their operation requires high precision and fast conversion rates to maintain overall system efficiency. The distribution of energy and area for a typical spatial architecture (Figure 4E), such as PUMA (59), reveals that the full precision ADC occupies more than 80% of the area and consumes more than 50% of the energy in the CIM accelerator. While digital components rank second highest in terms of area and energy, the contribution from the crossbar is minimal. Here, the crossbar refers to the memory array modified for analog CIM.

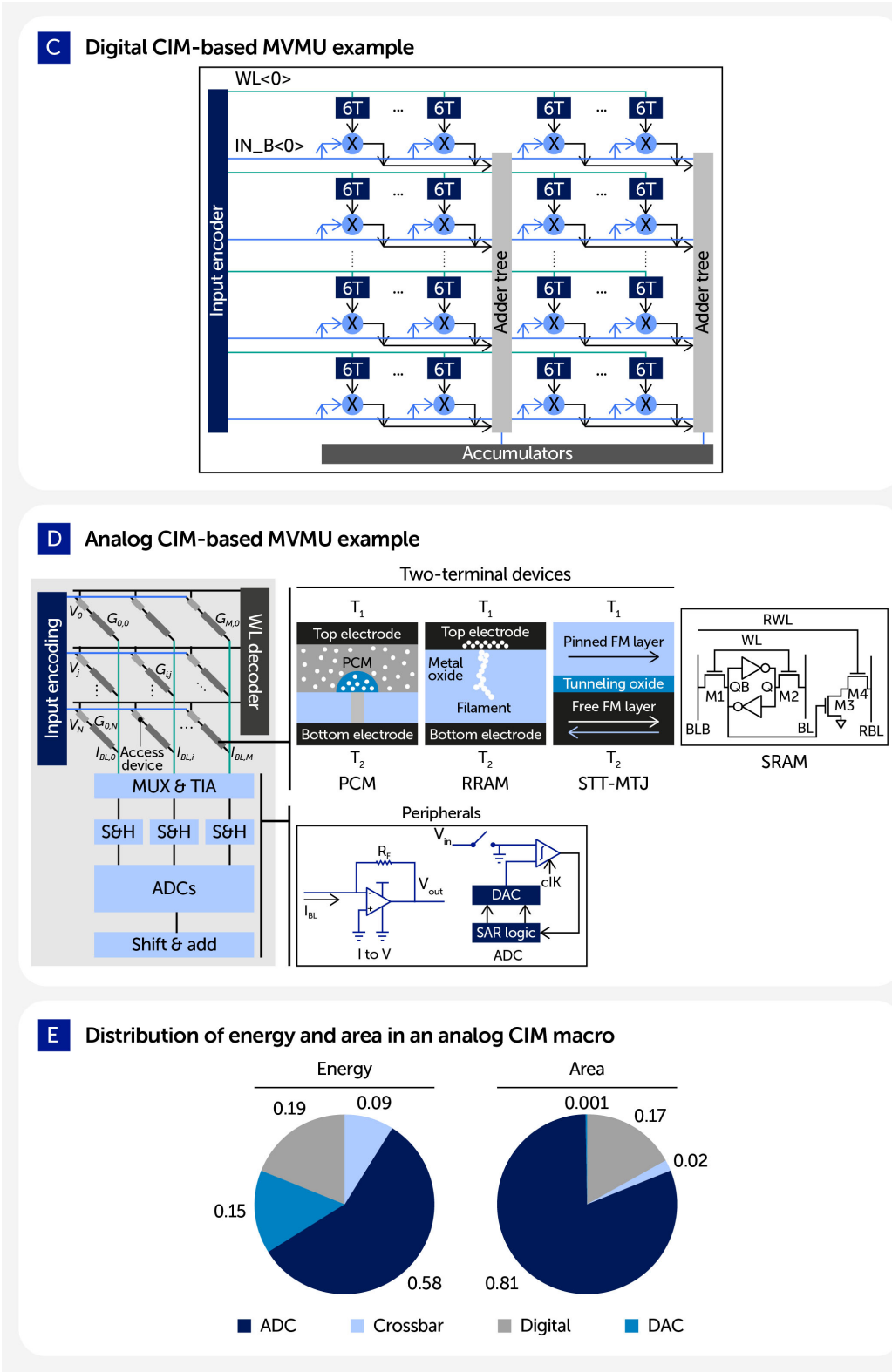
In contrast, digital CIM (Figure 4C) provides a fully digital alternative to analog and mixed-signal methods. In this architecture, digital CIM macros perform MAC operations using digital arithmetic

and logic operations either within or in close proximity to the memory array. In one variant of digital CIM, two wordlines (WLs) are activated, and their corresponding bitwise operations, such as NAND, XOR, and NOR, can be accessed at the periphery of the memory array through modified sense amplifier circuits. Such multiple bitwise operations can be combined to execute arithmetic operations such as multiply and accumulate. This approach, as noted by Wang et al. (62), necessitates storing inputs and weights in a bit-aligned manner within the memory



Abbreviations: **ADC,** analog-to-digital converter; **CIM,** compute-in-memory; **DAC,** digital-to-analog converter; **FIFO,** first in, first out; **FU,** functional unit; **MU,** memory unit; **MUX,** multiplexer; **MVMU,** matrix vector multiplication unit; **PC,** pattern comparator; **RAM,** random access memory; **ROM,** read-only memory; **SFU,** scalar functional unit; **VFU,** vector functional unit; ***XbarIn,*** cross bar input; ***XbarOut,*** cross bar output

FIGURE 4 (Continued)
Design of compute-in-memory (CIM)-based artificial intelligence (AI) accelerators using analog and digital computing schemes. **(A)** An example of a CIM-based AI accelerator with spatially distributed cores connected through a network on chip (NOC) is depicted. **(B)** Each CIM core consists of CIM-based matrix vector multiplication units (MVMUs). CIMs can be either digital or analog. **(C)** An example of a digital CIM MVMU consisting of 6T SRAM cells modified to perform bit-wise multiplications near each memory cell. **(D)** An example of an analog CIM MVMU, with different choices for memory cells and details of peripherals. **(E)** Analog-to-digital converters (ADCs) present in the peripherals dominate the computation costs in analog CIM, consuming the highest area and power.



Abbreviations: **ADC,** analog-to-digital converter; **BL,** bitline; **BLB,** bitline bar; **CIM,** compute-in-memory; **DAC,** digital-to-analog converter; **FM,** ferromagnetic; **I to V,** current to voltage converter; **M_{ij},** transistor number; **MUX,** multiplexer; **MVMU,** matrix vector multiply unit; **PCM,** phase change memory; **Q,** primary output; **QB,** complementary output; **R,** resistor; **RBL,** read bitline; **RRAM,** resistive random access memory; **RWL,** read wordline; **S&H,** sample and hold; **SAR,** successive approximation register; **SRAM,** static random access memory; **STT-MTJ,** spin-transfer torque-magnetic tunnel junction; **TIA,** transimpedance amplifier; **V_{OUT},** output voltage; **WL,** wordline

FIGURE 4 (Continued)

array and incurs a small area overhead due to minor changes in the peripheral circuitry. Figure 4C illustrates another method for a digital CIM core, which involves extensively modifying the memory array to incorporate multipliers and adders adjacent to each memory cell (18). The partial sum outputs are subsequently combined using adder trees and accumulators. The details of this digital CIM macro are further elucidated in the SRAM-based CIM subsection.

When comparing digital and analog CIM approaches, numerous trade-offs become apparent. Beyond the high area, energy, and latency overheads introduced by ADCs, analog CIM is susceptible to functional errors due to various non-idealities, including IR drop within the memory array, device non-ideal characteristics, leakage currents, and others (57, 58, 63, 64). Conversely, a digital computation environment is more predictable and easier to control, leading to more accurate and reliable system behavior. However, digital CIM may not achieve the high energy efficiency of analog CIM. Moreover, the repetitive bitwise additions required to generate the final MAC output in digital CIM can increase the overall compute latency.

Memory technologies

CIM technology can be applied in various ways depending on the type of memory, each offering unique advantages and specific implementations. The different types of memories include SRAM, dynamic random-access memory (DRAM), eNVMs, and flash memory. SRAM is commonly used as cache memory in processors due to its high speed and stability. An SRAM cell typically consists of 6 to 10 transistors and stores data in a binary format. In contrast, a DRAM cell comprises a single transistor and a capacitor to store data, making it cost-effective for high-density and large-capacity main memory in systems. However, it requires periodic refreshes to store data correctly, reducing its speed.

While both SRAM and DRAM are volatile memories, non-volatile memories such as hard disk drives (HDDs) or flash memory, found in solid-state drives (SSDs), can retain data when power is lost. Flash memory, devoid of moving parts, is faster and more durable compared to mechanical hard disks. Other alternatives, such as eNVMs—including resistive random-access memory (RRAM), phase-change memories (PCMs), spin–transfer torque magnetic RAM (STT-MRAM), and ferroelectric field-effect transistors (FeFETs)—are often noted for their low power consumption and high-density storage. These eNVMs, such as RRAM and PCM, can store more than 1 bit of data per cell and differentiate the state stored in the cell by switching between a high-resistance state (HRS) and a low-resistance state (LRS), triggered by an electrical stimulus such as a current or voltage pulse. The underlying physics of each technology is unique: STT-MRAM depends on the parallel and anti-parallel alignment of two magnetic layers separated by a thin tunneling insulator layer, PCM utilizes chalcogenide materials to switch between crystalline and amorphous phases, and RRAM relies on the formation and rupture of conductive filaments in the insulator between two electrodes. FeFET (65) relies on the polarization of the ferroelectric layer (e.g., Hafnium materials), thus storing different threshold voltages (V_{TH}).

Each of these memory technologies presents a unique set of trade-offs with respect to power efficiency, performance, cost, and ease of integration, which should be considered when integrating computing into memory for specific AI applications (Supplementary Table 1).

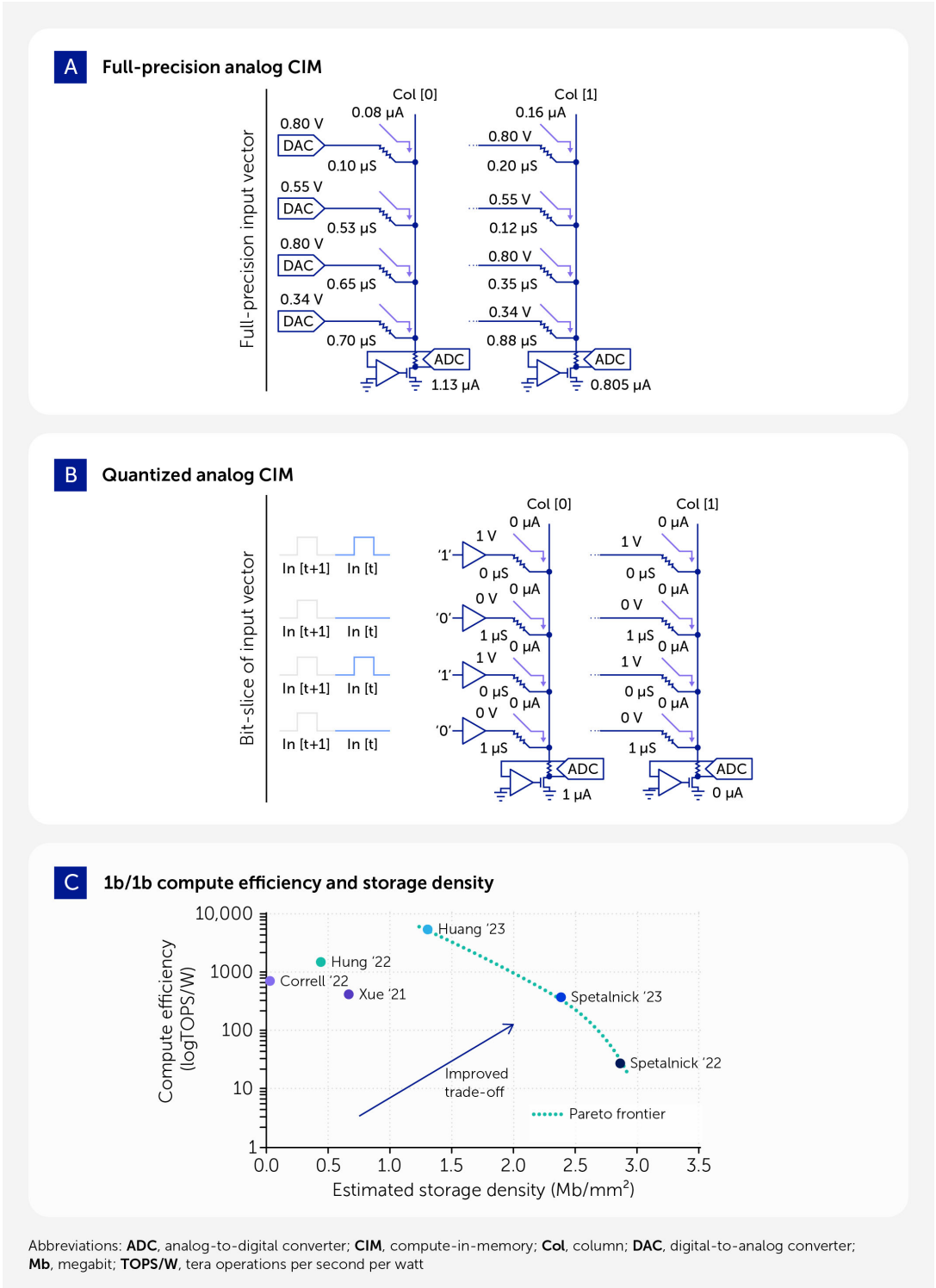
CIM across memory hierarchy: opportunities and challenges

Every computing system includes a memory subsystem with different memory levels to hide the memory latency during compute. For instance, the memory closest to compute is small and fast (e.g., SRAM) while the farthest memory is large and slow (e.g., DRAM). The last memory level is generally non-volatile, such as flash or using emerging devices. The aforementioned differences in memory characteristics necessitate examination of the CIM technologies for each memory technology separately. This subsection details the developments in CIM implementations with different memory technologies for performing matrix-vector multiplications.

eNVM-based CIM

Recent advancements in eNVM technologies (66–70) have garnered considerable interest for their application in power-sensitive and data-heavy environments (71). These technologies are characterized by low energy consumption, high storage density, non-volatility, and the capacity for *in-situ* parallel operations. eNVMs are employed in hardware specialized for various AI applications, including convolutional neural networks (59), graph neural networks (GNN) (72), SNNs (8, 73), and transformers (74, 75). These studies reveal key insights regarding CIM-based application-specific fabrics: (a) in some cases, the intrinsic stochasticity of eNVM can be advantageous, as demonstrated in specific applications (76, 77); (b) although the analog crossbar architecture offers attractive benefits, notably its fast MVM capability, the associated challenges with full precision analog CIM might impede its suitability for supporting large-scale applications in the future; and (c) carefully deciding the circuit and device parameters for the CIM array may help in alleviating some of these obstacles while designing energy-efficient edge computing hardware. The details of stochastic hardware are discussed in a later section, whereas the implications of (b) and (c) are discussed further here.

The strongest form of CIM with eNVM is the full-precision analog variant that performs an N-wide vector–vector dot product in each of the M active columns of cells during each read cycle. It offers the greatest potential advantages for compute efficiency and bandwidth but suffers from practical constraints on density and accuracy, as shown in the column view in Figure 5A. Practical considerations limit the feasibility of this full-precision analog approach in nano-scale nonvolatile array macros. For example, implementing precise analog inputs requires the cells and array metallization to support voltage biasing of individual storage elements. Typical 1T1R arrays natively support binary inputs to individual cells, using the WLs to toggle high-gain selector devices



Abbreviations: **ADC**, analog-to-digital converter; **CIM**, compute-in-memory; **Col**, column; **DAC**, digital-to-analog converter; **Mb**, megabit; **TOPS/W**, tera operations per second per watt

FIGURE 5
Design trade-offs in variants of embedded non-volatile memory (eNVM)-based analog compute-in-memory (CIM). **(A)** Full-precision analog CIM paradigm with eNVM, where full-precision inputs are reproduced using wordline (WL) digital-to-analog converters (DACs), while full-precision weights are stored in memory cells in the conductance domain. Each column represents a full 4-wide vector/vector dot product, resulting in a 1 × 4 by 4 × 2 matrix product with a 1 × 2 vector result. **(B)** Quantized analog CIM paradigm with eNVM: single-bit inputs are driven by buffers in place of DACs, and single-bit cell states are programmed into the memory columns. In the 2b case, each column now stores a single bit of data per cell. After two cycles, the result is a 1 × 4 by 4 × 1 product at INT2 input/weight precision. In both scenarios, **(A, B)**, a mapping between numerical values and voltage/conductance domain values must be chosen by designers. **(C)** Trade-off between compute efficiency [throughput per watt (TOPS/W, scaled to 1b/1b)] vs. estimated storage density (Mb/mm^2) of reported CIM macros using resistive random-access memory (RRAM), showing how the analog CIM variants differ from each other.

(78). In addition, the full-precision variant demands precise peripheral circuits, including TIA, ADC, and DAC, to maintain the accuracy of data bits.

To overcome the challenges of implementing full-precision analog CIM, three knobs can be tuned: reduce the input precision of P_x bits, reduce the weight precision of P_w bits, and reduce kernel width P_{WL}. This "quantized" variant is illustrated in Figure 5B. Note that completing a full vector–vector dot product with arbitrary input and weight precisions B_x and B_w and width N thus requires external shift-adding circuitry (Figures 4D, C) read operations Equation 2:

$$C = \left\lceil \frac{N}{P_{WL}} \right\rceil \times \left\lceil \frac{B_w}{P_w} \right\rceil \times \left\lceil \frac{B_x}{P_x} \right\rceil \tag{2}$$

In the most aggressively quantized mode, WLs and cell values are binary ($P_w = 1$ and $P_x = 1$). The read-out bitline current thus becomes the following Equation 3:

$$I_{BL\,.\,i} \propto \sum_{j=1}^{N} 1 \left| X_i = W_{ij} = 1 \right. \tag{3}$$

Several CIM macros using this family of partially "quantized" approaches with RRAM have been demonstrated in scaled nodes (79–89). Designers typically implement current-based readout schemes, as in traditional NVM designs, which operate by clamping the cell bias voltage, typically just the BL voltage, to an approximate target value and measuring the resulting current. Some outlier works avoid current-sensing and measure the BL voltage itself. Yoon et al. (83) used a voltage-based scheme with feedback, while Hung et al. (87) allowed the BL to slew downward due to selected memory cells without a pull-up transistor to reduce power while necessitating a carefully designed readout system.

Another approach to tackle the associated challenges with analog CIM includes co-designing crossbars to achieve the highest energy efficiency with minimal loss in accuracy. Chakraborty et al. (57, 58) and Kim et al. (90) showed how the analog nature of computing results in several non-idealities, such as non-linearity from access transistors, eNVM device non-linear characteristics, and parasitic resistance. They proposed a data-dependent approach and an analytical modeling approach, respectively, to capture such non-idealities, which lead to functional errors in the MVM computation output.

Sharma et al. (91) highlighted the impact of co-designing device-circuit parameters for reducing functional and quantization errors for crossbar arrays consisting of spin–orbit transfer magnetic tunnel junctions (SOT-MTJs). SOT-MTJs have high endurance, similar to their variant STT-MRAM, but offer a higher ratio of R_{off} by R_{on} due to their decoupled read and write paths. Their quantitative analysis shows that smaller crossbar sizes and the optimum size of R_{on} could lead to the least accuracy degradation in crossbars, depending on the input activation sparsity. In addition, higher sparsity can reduce energy consumption by leveraging a lower precision ADC without affecting accuracy. More approaches to reducing the ADC bottleneck are discussed in the next section.

Beyond macro-level works, several system-level CIM with RRAM designs have been demonstrated. Chang et al. (92) combined 288-unit CIM macros, each with eight channels and activating up to nine binary WLs in parallel, for a total of 2.25 MB of NVM storage on-die along with 768 KB of traditional SRAM. Power switching at each macro allows power to scale an order of magnitude as macros are enabled and disabled during intermittent computing. The power gating idea is extended by Chang et al. (93) and Lele et al. (94), who described two-tier power gating using 32 clusters of five macros each for a total of 1.25 MB of RRAM along with 1.25 MB of traditional SRAM. This second work also includes a lighter-weight SNN-based component utilized to trigger gated RRAM macros to wake up for heavier-weight CNN computing. Huang et al. (89) reported a relatively large die area (30.6 mm^2) and a large 4 MB of on-die nonvolatile memory to support slightly larger networks. They focused on a configurable macro with the ability to implement both in-memory and near-memory computing and achieved a very high 76.5 throughput per watt (TOPS/W) efficiency at INT8 precision at the macro level.

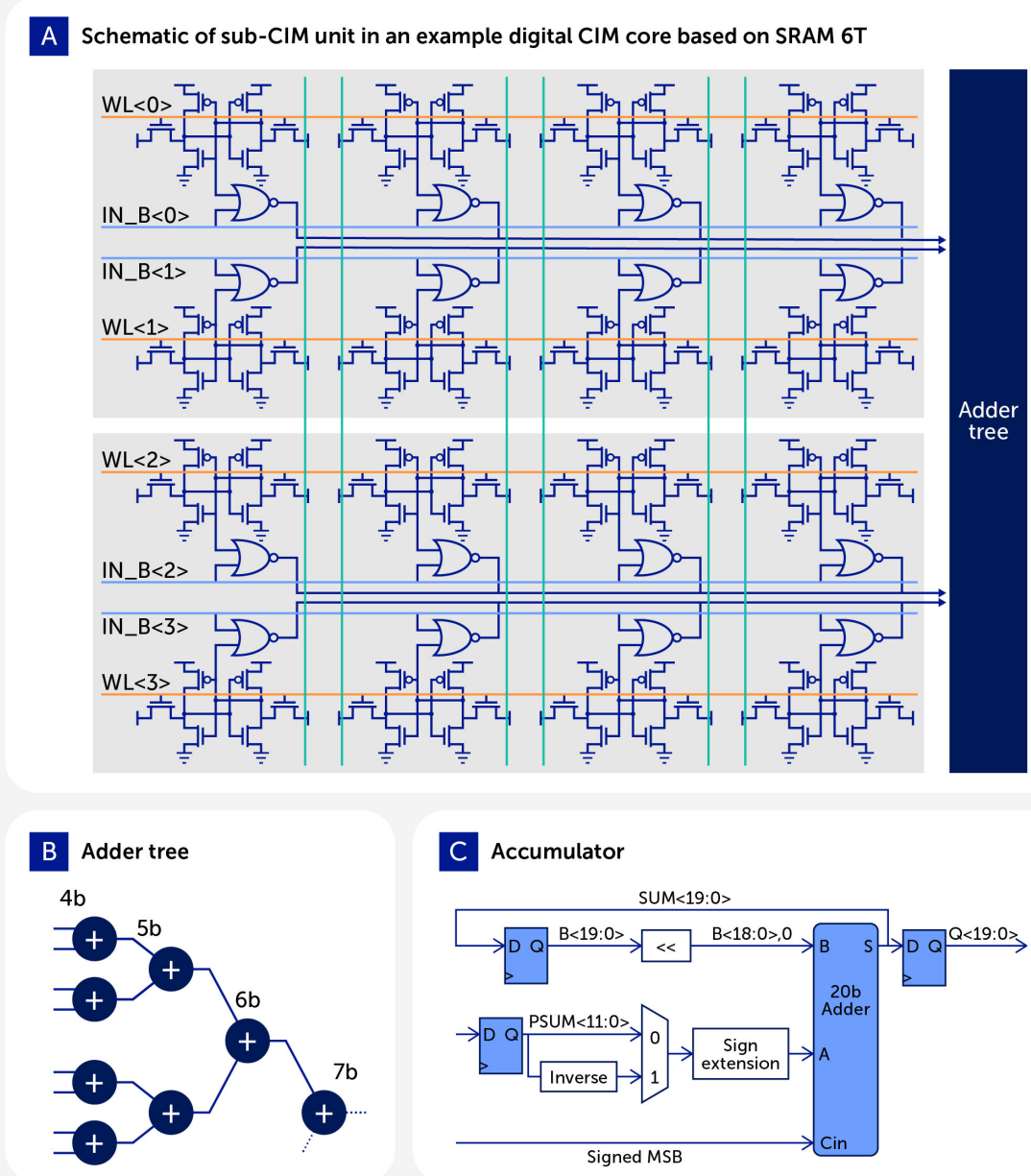
However, much of the above efforts have focused on specialized CIM designs tailored to specific workloads, which may diverge from the broader trajectory of hybrid SNN and ANN hardware development. To address the variety of operations used in such workloads, (i) Singh et al. (95) demonstrated architecture support for hybrid ANN and SNN workloads, (ii) Chen et al. (96) proposed a multifunctional CIM macro that uses the same memory array with combined peripherals to support logic-in-memory, content addressable memory (97), and dot-product computations, (iii) Wan et al. (98) presented a taped-out RRAM chip for multiple neural network architectures, and (iv) Shin et al. (99) demonstrated a taped-out RRAM chip for reconfigurable RNN-CNN workloads.

However, CIM with NVM is not a fully mature design concept, and the possibility of MAC-level accuracy degradation poses a significant risk to the future of the paradigm. Beyond accuracy issues, a major risk of CIM with NVM is a reduction in data density. Compared with traditional memory macros with binary read-out channels, CIM designs require more complex peripheral designs, such as medium-precision ADCs (100), complex sensing circuits (101), and isolation circuitry for high-voltage domains (88), which diminish array area efficiency. Furthermore, the top-level NVM data densities of these CIM systems, ranging from 0.49 to 1.31 Mbit/mm^2, compare poorly with those of digital systems using NVM, reported at 1.92 and 2.80 Mbit/mm^2 (102, 103). Figure 5C depicts an efficiency versus data density frontier observed empirically in published works, mostly macros.

SRAM-based CIM

Initial CIM efforts focused on modifying SRAM 6-transistor (6T) arrays, a primary type of memory storage, to act as classifiers or MLPs for tasks such as digit recognition (104, 105). While such efforts focused on approximate analog CIM inference accelerator designs, ADCs were incorporated later to maintain result accuracy, leading to more complex designs with higher data precision (106). Notably, SRAM 6T cells have coupled read–write paths, which can cause bit flips through a short circuit when all WLs are enabled in an array. To address these read-disturb issues, Kang et al. (104)

employed staggered activation and asymmetric sense amplifiers to avoid short circuits and detect bitwise NAND, NOR, and so on, respectively. Recently, Chih et al. (18) proposed a full-precision digital CIM macro using SRAM 6T, where the multipliers are connected directly to the BLs of each cell (Figure 6A). This approach overcomes the analog CIM overheads without any loss in computation accuracy. The entire array is divided into multiple sub-CIM units via the adder tree (Figure 6B). Each sub-CIM unit activates all input activations simultaneously (represented by a blue line), and the partial sum of 4-bit MAC from each row is sent to the adder tree. For the subsequent stream of inputs, the partial sums are combined using an accumulator (Figure 6C), which bit-shifts the last computed output and adds the current partial sum from the adder tree. Follow-up work (107) further increased the TOPS/W by



Source: Re-used from (18), with permission from IEEE.

Abbreviations: **CIM**, compute-in-memory; **IN_B**, input bits; **MSB**, most significant bit; **PSUM**, partial sum; **SRAM**, static random-access memory; **WL**, wordline

FIGURE 6
Circuit details of a digital compute-in-memory (CIM) macro example with static random-access memory (SRAM)-6T cells. **(A)** Sub-unit in an SRAM-6T array where each cell, storing weights, has a multiplier (NOR gate) associated with it. **(B)** The partial sum of 4-bit inputs and 4-bit weights goes to the adder tree. **(C)** Eventually, multiple partial sums are accumulated in the accumulator. Thus, each sub-unit in the array performs a 4-bit – 4-bit multiply and accumulate (MAC) computation. Re-used from (18) with permission from IEEE.

including pipelining, bit-width flexibility, and simultaneous weight updating in the digital CIM macro.

As SRAM 6 T-based CIM designs have evolved, the SRAM-based CIM classifiers have expanded to include other types of SRAM cells, thereby eliminating the read-disturb issues associated with SRAM 6T. This expansion is evident in numerous simulation works and silicon prototypes consisting of 8T (60, 62, 108), 10T (109), and 12T (110) cells. Although larger SRAM cells compromise on storage density, they provide greater read-disturb stability. For instance, SRAM 8T has decoupled paths for reading and writing to the cell. In another effort, a charge-domain computing approach was shown to be more efficient for such analog CIM macro designs compared to the earlier used current-based computing, which necessitates bulky transimpedance amplifiers (TIAs) for current-to-voltage domain conversion (61, 111). Additionally, Wang et al. (62) employed a digital CIM approach by activating two rows at a time in an SRAM-8 T-based memory array to calculate logic operations such as NOR and AND at the end of the BL.

Subsequently, SRAM CIM inference accelerators or macros have expanded to processor-level design (18), programmable chip designs (112), and floating-point accelerators (113). To balance the accuracy of fully digital CIM and the energy efficiency of analog designs, researchers have proposed analog prototypes with reconfigurable ADC precision (114). These reconfigurable ADC designs employ high ADC precision for scenarios with lower data sparsity and lower ADC precision for high data sparsity. Thus, the reconfigurable ADC design can achieve optimal accuracy and energy efficiency, varying according to the signal-to-noise ratio and data sparsity.

Nevertheless, ADCs alone consume as much as 60% of the energy and occupy nearly 80% of the area in analog CIM accelerators (115). Due to the significant area requirements of ADCs, they are shared across multiple columns in the crossbar, which limits the throughput of the crossbar arrays. Researchers have explored various strategies to mitigate the limitations imposed by ADCs in CIM accelerators (28, 116–119). These studies have primarily focused on reducing the precision of ADCs through partial sum quantization (PSQ) to lower bit levels, thereby saving power and area. Despite these adjustments to lower-precision ADCs, the challenge of inefficiency persists. Building on the PSQ technique, recent research has demonstrated that partial sums can be quantized to just 1 (binary) or 1.5 bits (ternary) (28, 120). This method eliminates the need for ADCs by adopting learned-step quantization (121), utilizing trainable floating-point scale factors. The binary or ternary value (p) at the columns of crossbars is multiplied by a scale factor (s) to bring the quantized values to a similar range as the actual floating-point data. The binary and ternary quantization values for p are delineated in Equation 4, where α represents a trainable threshold.

$$p_b = \begin{cases} 1 & if\ \ ps \geq 0 \\ -1 & if\ \ ps < 0 \end{cases} \quad p_t = \begin{cases} 1 & \text{if } ps \geq \alpha \\ 0 & -\alpha < ps < \alpha \\ -1 & \text{if } ps \leq -\alpha \end{cases} \tag{4}$$

The partial sum is shifted and accumulated across all the input bit streams to obtain the final partial sum value (PS) at a crossbar column. Experiments underline the importance of using scaling factors in PSQ and reveal that binary and ternary PSQ with scaling factors achieve higher accuracy than 2-bit quantized partial sums. This partial sum quantization approach has also been applied to SNNs when deployed on analog CIM accelerators (122). In this context, it has demonstrated up to 72× lower inference energy and 10× lower latency compared to deployments on the NVIDIA Jetson TX2 board.

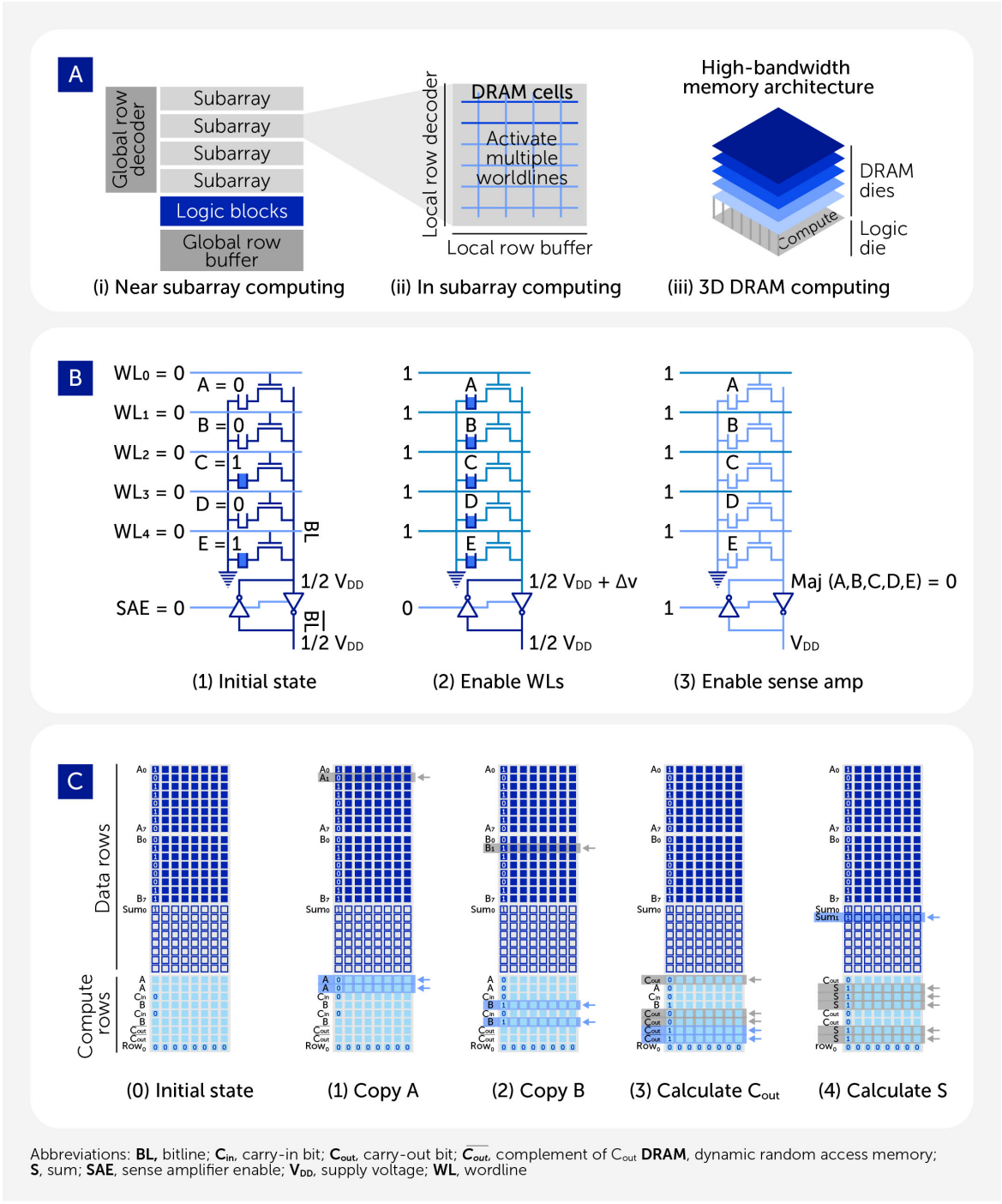
DRAM-based CIM

In most ML applications, the DNN models and data processed are too large to fit on-chip, which implies that the majority of the model and data reside in DRAM. The movement of data to and from DRAM consumes significant time and energy in current AI hardware, including GPUs and specialized ML accelerators such as tensor processing units (TPUs) or neural processing units (NPUs). This makes processing inside or near the DRAM a promising approach to improving the efficiency of future AI hardware. There are different flavors of DRAM-based CIM; they differ based on where and how processing logic is integrated into the DRAM subsystem (Figure 7A).

i. Computation is performed near DRAM subarrays [Figure 7A(i)]. Data are read from the subarrays and fed to logic (computation units) near the subarrays for processing. Despite the logic being slower, as it is being realized in a DRAM process, overall processing efficiency is improved by reducing the distances data travel and by utilizing the higher internal bandwidth available within the DRAM.
ii. Computation is performed in DRAM subarrays by activating two or more wordlines, as shown in Figure 7A (ii). The results are obtained at the columns of the subarray with the help of the local sense amplifiers. Not all functions may be suitable for realization within subarrays, in which case a combination of in-subarray and near-subarray computing may be used.
iii. Compute units are placed on logic dies that are integrated with 3D-stacked DRAM dies as shown in the high bandwidth memory architecture of Figure 7A(iii). The through-silicon via (TSV) connections provide high-bandwidth data transfer between the compute units and DRAM.

Each approach has its benefits, and recent efforts have explored all three approaches for accelerating AI workloads.

In-subarray computing utilizes the maximum internal DRAM bandwidth, as computation occurs at local sense amplifiers; ideally, this should lead to a more enhanced performance compared with near-subarray and 3D DRAM variants. Initial efforts (123–127) focused on evaluating Boolean computations at local sense amplifiers by activating multiple rows of the DRAM subarray. Ambit (123) specifically accelerates workloads through bulk bitwise Boolean operations (AND, OR, and NOT). The DRAM-based Reconfigurable *In-Situ* Accelerator (DRISA) (124) features a



Abbreviations: **BL,** bitline; C_{in}, carry-in bit; C_{out}, carry-out bit; $\overline{C_{out}}$, complement of C_{out} **DRAM**, dynamic random access memory; **S**, sum; **SAE**, sense amplifier enable; V_{DD}, supply voltage; **WL**, wordline

FIGURE 7
Example demonstrating in/near memory computing for dynamic random-access memory (DRAM). **(A)** A variety of in/near memory computing is available for DRAM based on where or how processing logic is integrated. (i) One method involves processing near the DRAM subarray in the DRAM banks by feeding data from the subarrays to the logic units. (ii) An alternative approach does the computation in the subarray by activating two or multiple wordlines. (iii) In 3D architectures, processing is done with the help of computation units in the logic die. **(B)** The majority of DRAM cell operations involve the simultaneous activation of multiple worldlines and the subsequent activation of the sense amplifiers after charge sharing. **(C)** This figure shows the steps of a single bit in an n-bit bit serial addition operation (128). The first two steps are to copy the operands to the compute rows, followed by a triple row and a quintuple row activation to obtain the carry out and sum bit.

reconfigurable *in-situ* accelerator designed for performing Boolean operations. Conversely, DRAM based Accelerator for Accurate CNN Inference (DrAcc) (125) implements ternary neural networks through in-DRAM bit operations. Efficient and Low Power Processing-in-Memory (ELP2IM) (126) presents techniques for the low-power realization of bitwise operations in DRAM. Single Instruction Multiple Data DRAM (SIMDRAM) (127) introduces an end-to-end, flexible, general-purpose

framework for bit-serial (Single Instruction Multiple Data) (SIMD) computing in DRAM.

The novel approach to bit-serial addition operations within DRAM subarrays introduced by Ali et al. (128) serves as an illustrative example of in-subarray computing. This approach requires data within DRAM to be organized in a transposed format, aligning all bits of the two multi-bit operands in the same column. This arrangement facilitates parallel n-bit addition operations across all columns within the subarray, as well as across subarrays. In each column, an addition operation is realized bit-serially, with a full adder function decomposed into majority operations, realized by activating multiple wordlines (Figure 7B). Five DRAM cells store data (A, B, C, D, and E) in their initial state, and the corresponding five wordlines are simultaneously activated to allow charge sharing. Following this, sense amplifiers activate to evaluate the majority. Both sum and carry outputs for bit-serial addition are evaluated through a majority operation. Due to the destructive nature of these operations, data are first copied into dedicated compute rows for processing (Figure 7C). An arbitrary bit-width addition requires nine additional compute rows alongside the data rows, incurring less than 1% area overhead. Figure 7C outlines all steps involved in single-bit addition, starting with copying operand bits to compute rows A and B to preserve the original data. The carry-out bit (Cout) is evaluated through a triple-row activation of A, B, and the carry-in bit (Cin). The sum bit results from a quintuple-row activation involving A, B, Cin, Cout, and Cout complement. Each of the four steps in a single-bit addition operation requires one AAP (activate-activate-precharge) operation. For an n-bit bit-serial addition operation, 4n+1 AAP operations are necessary. A system-level evaluation of this approach demonstrated up to an 11.5-fold performance improvement compared with conventional von Neumann machines when running the k-Nearest Neighbor (kNN) algorithm on the MNIST handwritten digit classification dataset.

Building upon this approach, Roy et al. (129) proposed an accelerator for ML inference workloads that combines in-subarray computing with near-subarray computing. A novel multiplication primitive was introduced without significant changes to the DRAM subarray. Additionally, an architecture comprising reconfigurable adder trees and special function units was proposed to fully realize DNNs within DRAM itself. A mapping scheme was also introduced alongside the architecture to maximize utilization and, consequently, performance.

Near-subarray computing has been explored in several efforts (127, 130, 131). Newton's proposal by He et al. (130) involves an in-DRAM accelerator architecture incorporating multiply and accumulate units along with buffers within the DRAM. RecNMP, by Ke et al. (131), accelerates personalized recommendation systems through near-memory processing.

3D DRAM computing was explored in various efforts (132–134). Notably, Neurocube, by Kim et al. (133), introduced a programmable and digitally scalable accelerator built upon 3D high-density memory for efficient neural network acceleration. Gao et al. (134) proposed a scalable NN accelerator using 3D memory.

It is expected that some of the above computing in memory techniques will make their way into commercial DRAM products. For example, Samsung has recently incorporated computing in memory into their high-bandwidth memory (HBM) products (135), enabling parts of AI workloads to be accelerated in DRAM.

Flash-based CIM

Similar to other memory technologies, flash memory, a type of non-volatile storage, can be utilized to perform various computational operations, such as bitwise operations (AND, OR, and XOR), basic arithmetic operations, and more complex tasks, including searching and pattern matching. These operations are typically executed using the charge storage properties of flash cells. Predominantly available in two types, NAND and NOR, flash memory is structured around floating-gate transistors. NAND flash, characterized by its high density, is commonly used for data storage, while NOR flash, known for its faster read capabilities, is often employed in code storage and execution.

Xiang et al. (136) proposed an efficient, mixed-signal, NOR flash memory-based CIM design for SNNs. Since SNNs compute only spikes, which can be represented in binary format, such a design can eliminate the use of ADC/DAC, thus conserving energy and area. Similarly, Choi et al. (137) utilized two single-level NAND floating gate cells to store complementary data for a binary neural network algorithm. In S-FLASH, Kang et al. (138) employed a full mixed-signal CIM design and optimized energy by modifying the bit width of partial multiplication and exploiting many zero partial multiplication results. Additionally, works such as ParaBit by Gao et al. (139) and 3D-FPIM by Lee et al. (140) have performed end-to-end evaluations of NAND flash-based CIM architecture, demonstrating their potential to achieve higher performance and energy efficiency compared to RRAM-based accelerators.

Accelerating functions beyond matrix multiplication

In addition to matrix multiplication operations, memory can be repurposed to accelerate other AI application operations. The various neural network models discussed above (e.g., RNNs, SNNs, and transformers) often require repetitive evaluation of transcendental functions, such as tanh, exponential, and softmax. Evaluating these functions through the Maclaurin series expansion is deemed impractical. Consequently, previous studies have employed range reduction techniques and mathematical tables (141). For enhanced energy efficiency, it is preferable to store these tables on-chip rather than in off-chip DRAMs. However, the integration of these tables as on-chip read-only memory (ROM) necessitates additional silicon area, thus increasing costs. Lee et al. (142) proposed a ROM-embedded STT-MRAM bit-cell that includes an additional bitline for ROM operation: R-MRAM. This design effectively combines ROM and RAM functionalities within the same layout, allowing both types of memory to be utilized simultaneously without increasing the array size. The memory configuration might contain, for example, X MB of RAM and an equal amount of ROM. The circuit features an extra bitline compared to standard STT-MRAM, as shown in Figure 8. ROM data are stored by selectively connecting the magnetic tunnel junction (MTJ) to one of the two Vread bitlines. Note that a

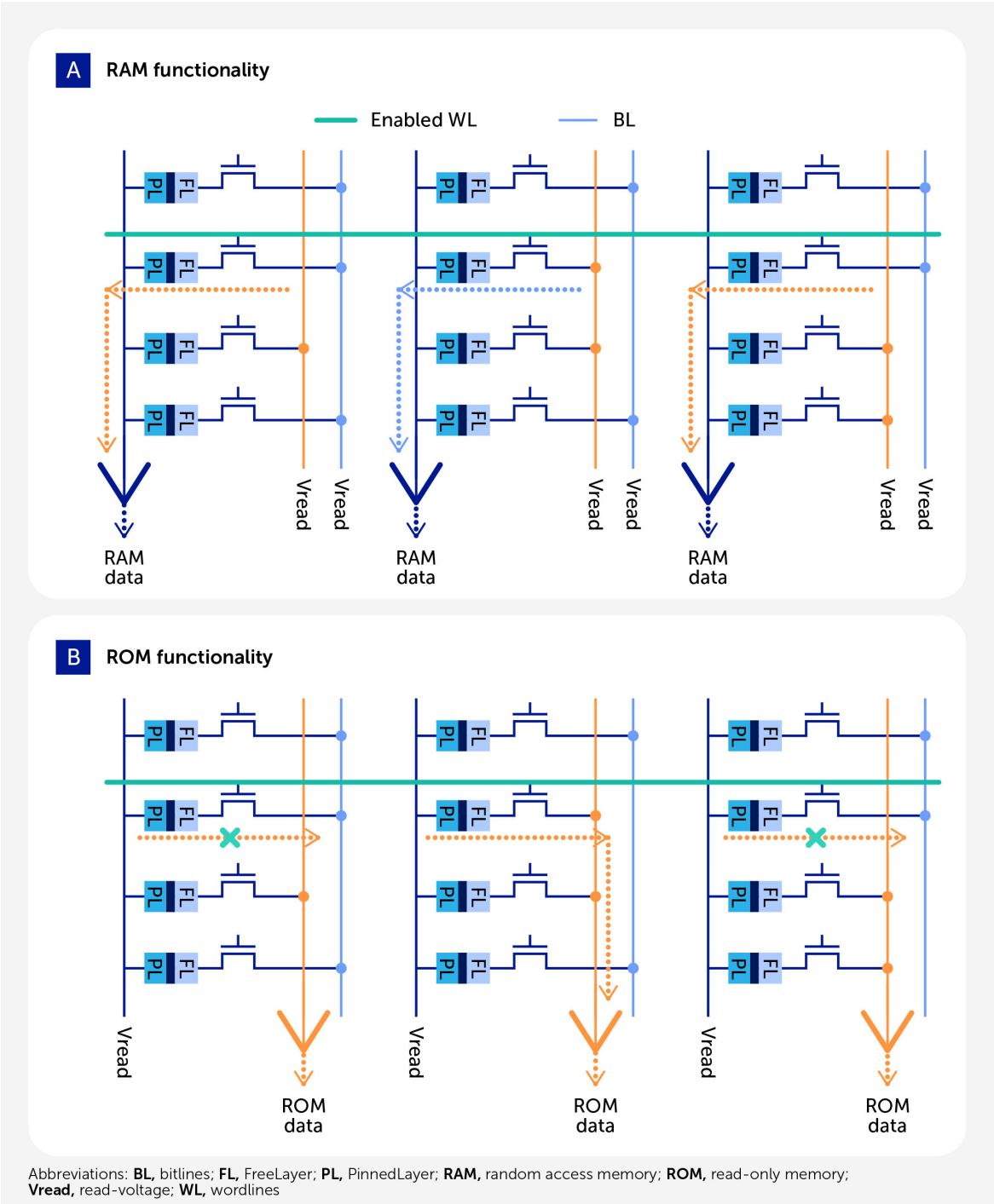


FIGURE 8
Depiction of read-only memory (ROM) embedded random-access memory (RAM) with spin–orbit transfer magnetic tunnel junction (SOT-MTJ) cells. The connection of the MTJ cell with the Vread bitline decides the value of the data stored in ROM. **(A)** RAM functionality is achieved by enabling both bitlines, and **(B)** ROM functionality by enabling only one bitline, respectively, with the proper wordline (shown horizontally).

typical layout of STT-MRAM bit-cell has enough routing space for this extra bitline since it requires a large access transistor to satisfy the bidirectional switching current requirements of the MTJ. This additional bitline enables standard RAM read/write operations as well as ROM read operations in R-MRAM. In the RAM mode, both bitlines are activated simultaneously with the proper wordline, while for ROM retrieval, only one of the bitlines is activated. Hence, during ROM retrieval, the data read from a particular cell is either the resistance of the MTJ (either in the on or off state) or infinite if the cell is disconnected from the bitline. The requirements for peripheral sensing circuitry differ slightly from those in standard RAM. Notably, such a ROM configuration could be applied to other

non-volatile memory technologies, such as PCM and RRAM, and volatile memories, such as complementary metal-oxide semiconductor (CMOS)-based SRAM.

For implementing ROM in SRAM technology, Lee and Roy (143) proposed using two wordlines per 6-T cell, where the ROM data are stored based on whether the left access or the right access transistor is connected to wordline1 or wordline2. Note, during RAM mode, both wordlines are activated for read operations; however, for writing, only one of the wordlines is activated, effectively making the cell a 5-T cell during writing. Since the cells are volatile, during ROM data retrieval, the RAM data are first stored in a buffer and written back to the cell after the ROM read.

Dutta suggested a novel modification to the standard 8T SRAM cell that enables it to store ROM data without affecting its normal functionality or requiring extra silicon area (144). Building upon this, they demonstrate the use of this modification in an 8T-SRAM-based MVM unit. Storing "1" and "0" in the MVM unit requires an additional terminal, RCON. For cells storing a value of "1", the source line (SL) is connected to RCON, and for cells storing "0", SL is connected to the ground (GND). When RCON is connected to GND, the SL of all cells connects to GND, making the array function like a conventional SRAM array. However, if RCON is connected to the power supply/drain voltage (VDD) and all cells in that row store "'1", specific read bitlines (RBLs) are prevented from discharging and activating the ROM mode. For ROM operation, the rows of the 8T SRAM array must store "1" as their value, which erases any pre-existing data. Thus, the data from the SRAM array should first be transferred to a temporary buffer and then restored post-ROM data read. Therefore, using an 8T-SRAM for dual functionality (normal SRAM and ROM-embedded RAM) necessitates four cycles for a ROM data read: (i) copying SRAM row data to a buffer, (ii) writing "1" to all cells in the row, (iii) reading ROM data, and (iv) writing the data back from the buffer to the SRAM. This novel 8T SRAM-based design has also been evaluated at the system level to accelerate Transformer models by Kim et al. (145), demonstrating up to a 10× improvement in throughput compared to the NVIDIA A40 GPU (145).

Neuromorphic computing: designing brain-inspired AI hardware

As observed in the exemplary application of autonomous drone navigation, the network architecture plays a critical role in determining the parameters, accuracy, energy consumption, and latency of the system. In fact, the optimal architecture may be a hybrid of SNNs, ANNs, and LSTMs. Although some of the previously discussed ANN hardware can support SNNs, achieving optimal performance requires the development of specialized hardware for neuromorphic computing (or SNNs).

Neuromorphic computing, which draws inspiration from the brain's ability to unify computation and memory storage within the same physical substrate, its highly dense and recurrent connections, and its sparse, spike-based computation and communication, results in a remarkably efficient system. In recent years, SNNs, inspired by biological neurons, have emerged as promising candidates for processing sequential tasks with an asynchronous and sparse event stream (36). Central to SNNs is a neuron model characterized by an internal state known as membrane potential (8). Leaky integrate-and-fire (LIF) is the most commonly used neuron model. In a LIF neuron, the membrane potential accumulates input and decays at a certain rate governed by the leak at each timestep, emitting a spike when it exceeds a threshold. This process enables SNNs to learn the input timing information without any explicit temporal encoding, making them a special, less complex case of RNNs.

Earlier, SNNs were used for static tasks, such as image classification, where they lagged behind ANNs in accuracy. However, recent research has shown that SNNs excel in dynamic tasks, such as gesture recognition and optical flow estimation, surpassing ANNs and RNNs in accuracy (47). Their spike-based computation, focusing on accumulation operations, consumes less energy than the multiplications required by ANNs. However, the additional membrane potential parameter and temporal dimension introduce unique requirements and challenges that differentiate SNNs from standard ANNs, as summarized below:

i. Temporal processing requirements: SNNs operate in discrete timesteps, encoding information in the timing of spikes and processing weights and membrane potential data structures across several timesteps. Consequently, they require precise hardware to manage complex temporal processes, such as accurate spike timing and event-driven communication, while minimizing memory access.
ii. Neuron activation functions: Unlike ANNs, which employ continuous activation functions, such as sigmoid or rectified linear unit (ReLU), SNNs utilize complex activation functions that account for the temporal dynamics of spikes. This demands specialized hardware tailored to their unique time and frequency dependencies.
iii. High temporal and spatial sparsity: SNNs demonstrate inherent sparsity in time, with spikes occurring intermittently, and in space, with only a few neurons active at any given time. Accelerating SNNs necessitates optimizing computations for this dual sparsity, which can pose challenges for traditional hardware platforms.

Efficient SNN acceleration requires a comprehensive understanding of these challenges and characteristics. To that end, several hardware solutions have been proposed, tailored to the distinct requirements of these biologically plausible neural networks. Asynchronous event-driven architectures have emerged as a transformative approach in this domain. IBM's TrueNorth (146), with a million neurons and 256 million synapses, exemplifies low-power, real-time processing. Its asynchronous communication with synchronous neurosynaptic cores, connected through a distributed 2D mesh architecture, emphasizes parallelism and efficiency in handling spiking events. Similarly, Intel's Loihi (147) features a network of asynchronous neuromorphic cores interconnected through an asynchronous mesh network. Its

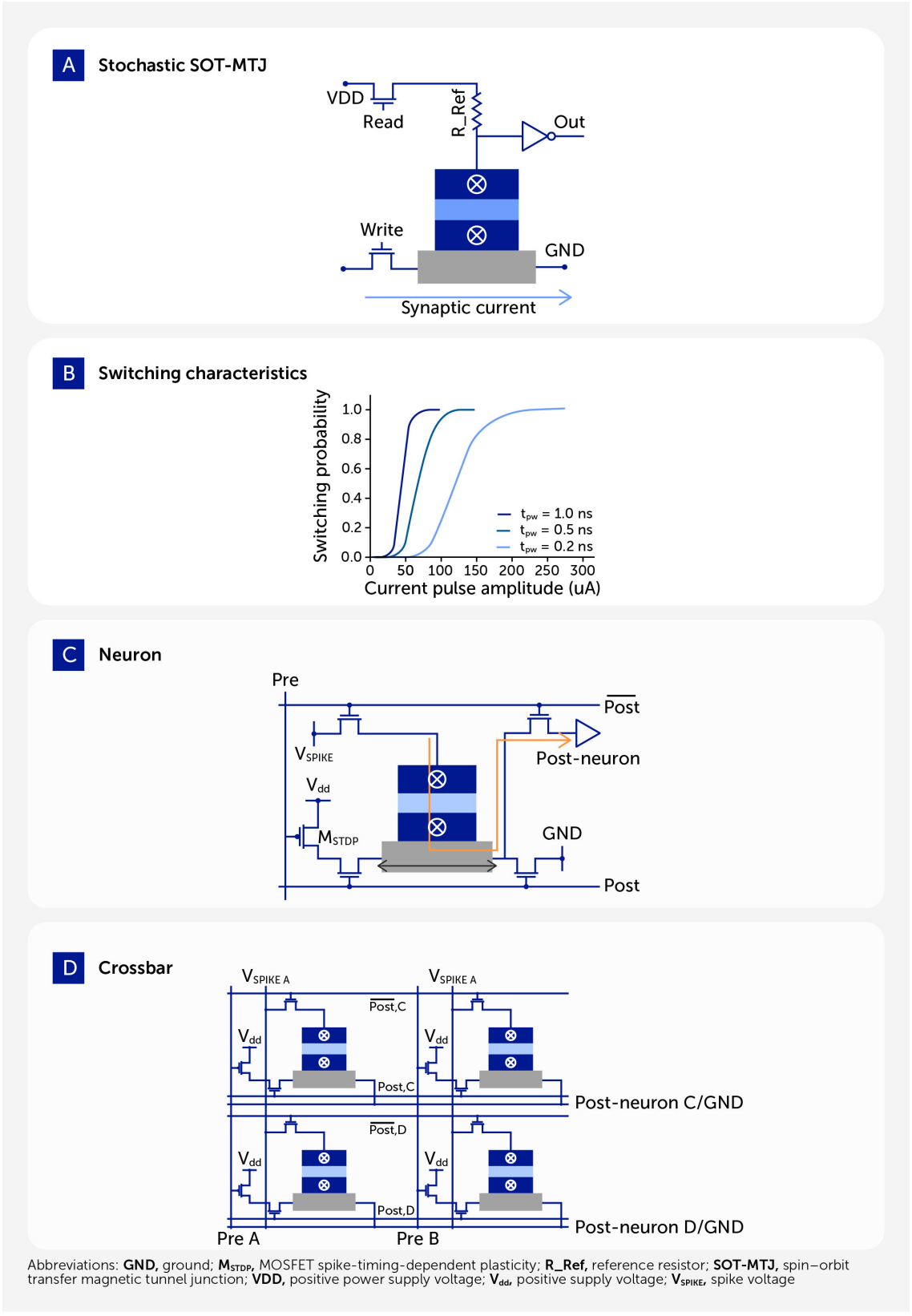
second iteration enhances scalability and energy efficiency, further supporting advanced SNN research and applications (148). Several other designs have also explored asynchronous event-driven approaches for large-scale brain simulations (149–151), and ultra low-power, resource-constrained edge applications (152–155).

As previously discussed in the CIM section, in-memory computing reduces weight and activation data movement in ANNs. In the SNNs case, there is additional data movement resulting from the temporal dimension and the membrane potential data structure. In light of this, Agrawal et al. (156) proposed an SRAM-based digital in-memory computing macro designed specifically for SNNs. This macro integrates essential SNN inference operations, such as accumulation, thresholding, spike-check, and reset, within a fused weight and membrane potential memory. The proposed macro employs a staggered layout to support 6-bit weight to 11-bit Vmem additions using reconfigurable column peripherals. It also supports multiple neuron models, including IF, LIF, and residual membrane potential (RMP) neurons, using in-memory operations. Such a macro can serve as a computational unit for a large-scale hierarchical mesh architecture (59). While early SNN research primarily emphasized biologically inspired neuron models and circuit primitives, recent advances underscore the importance of a system-level design to fully realize the benefits of temporal sparsity and event-driven computing. This includes dataflow-aware accelerators that align compute scheduling with sparse, asynchronous spike events (157, 158), algorithm-hardware co-design strategies that leverage CIM for efficient spike-based inference (90, 155), and hybrid architectures that integrate near- and in-memory computing with support for both frame- and event-based modalities (93). Recent efforts such as NeuroBench (159) aim to standardize evaluation and benchmarking of neuromorphic systems across datasets, tasks, and hardware backends, further enabling fair comparison. These holistic approaches enable scalable and energy-efficient deployment of SNNs beyond small-scale benchmarks and represent a critical bridge between neuroscience-inspired models and practical machine intelligence.

Stochastic hardware

While traditional neuromorphic computing models have predominantly relied on deterministic neural and synaptic primitives, recent efforts have adapted these models to incorporate stochastic elements, leading to models that are compact while maintaining accuracy for a class of applications (160–163). The integration of stochasticity into algorithms has been shown to improve the energy efficiency of diverse AI workloads. One notable advancement involves leveraging stochasticity for information encoding over time, achieved through probabilistic synaptic or neural updates. This facilitates the state compression of neural and synaptic units, enabling their implementation through single-bit technologies. For example, Roy et al. (163) developed fully binary neural networks using stochastic activation functions. Srinivasan et al. focused on training SNNs using a local learning rule (spike timing-dependent plasticity or STDP) (164) based on stochasticity (160). STDP updates synapses proportionately based on the timing difference between their input and the output spiking of the neuron. If the time difference is positive, the synapses are proportionately potentiated, while if it is negative, the synapses are proportionately depressed. Stochastic STDP has been used for synaptic updates where the time difference between input switching and output switching time is used as a measure of the probability of switching. Stochastic binary multi-layer networks exhibit heightened representation capacity and excel in classification tasks compared with their deterministic counterparts (165). The regularization effect induced by stochasticity plays a pivotal role in achieving such results. As researchers delve deeper into the interplay between stochasticity and algorithmic design, the prospect of unlocking novel capabilities in AI systems becomes increasingly promising.

At the hardware level, the core of stochastic computing circuits comprises controllable true random number generators (TRNGs), known as stochastic bits. The physics of various non-volatile memories demonstrates inherent stochasticity, which can be leveraged to implement stochastic algorithms with high efficiency. This stochasticity is intimately tied to the unique switching mechanisms of these devices. For instance, spintronic devices utilize thermal noise for magnetic orientation switching (166). Similarly, ReRAM devices depend on conductive filament formation or the movement of oxygen ions/vacancies (167), and PCM devices switch through a process that involves heating and abrupt quenching (168). All these mechanisms inherently exhibit stochastic characteristics. Furthermore, the non-volatility of such devices eliminates the need for separate storage of stochastic bits and enables their use in training neural networks. One particularly promising approach is the utilization of SOT-MTJs, which are three-terminal, read–write separable devices distinguished by their stochastic sigmoid-like switching characteristics (162). Figure 9A illustrates an SOT-MTJ device, where a magnetic tunnel junction comprising two nano-magnets (the bottom being the free layer and the top being the fixed layer, separated by a very thin oxide layer) is situated atop a heavy metal, depicted in gray. The current flowing through the heavy metal potentially switches the bottom nano-magnet. Depending on the magnitude and width of the current pulse, there is a probability that the SOT-device (the free layer magnet) switches, as shown in Figure 9B. Such a device can serve as a neuronal activation function, as previously mentioned. Figure 9C presents a circuit designed by Srinivasan et al. that implements stochastic STDP updates for synapses in a crossbar array (Srinivasan et al., 2020). The gate of the transistor M_{STDP} receives a voltage ramp input with an input spike. The two bottom pass transistors activate only when the POST signal turns high with an output spike. Consequently, the current flowing through the heavy metal (through transistors M_{STDP} and the two pass transistors) is proportional to the time difference between the input and output spike times, leading to the required stochastic switching of the device and, hence, stochastic STDP. Figure 9D displays a crossbar-type circuit array incorporating the stochastic learning circuit (Figure 9C). MTJs with high barrier magnets—a barrier height greater than 40 K_BT (where K_B is the Boltzmann constant and T is the operating temperature)—have a high retention time of approximately 7 years. However, they require an external input



Abbreviations: **GND,** ground; **M_{STDP},** MOSFET spike-timing-dependent plasticity; **R_Ref,** reference resistor; **SOT-MTJ,** spin–orbit transfer magnetic tunnel junction; **VDD,** positive power supply voltage; **V_{dd},** positive supply voltage; **V_{SPIKE},** spike voltage

FIGURE 9
Elements of stochastic hardware using spin–orbit transfer magnetic tunnel junctions (SOT-MTJs). **(A)** The structure of an SOT-MTJ with read and write circuitry to utilize it as a stochastic synapse. **(B)** The switching probability of a stochastic synapse between anti-parallel and parallel states with different current pulses. **(C)** A circuit consisting of SOT-MTJ for implementing spike timing-dependent plasticity (STDP) updates for a synapse in a crossbar-type array shown in **(D)**.

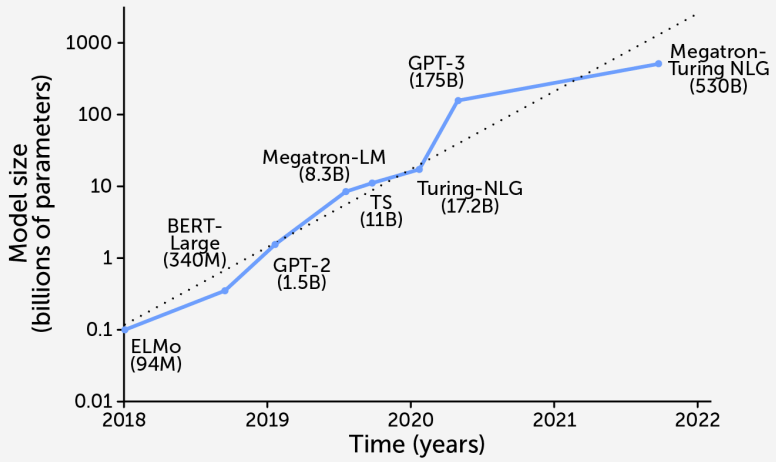
(current) to assist in switching. Conversely, low barrier magnets—a barrier height less than 15 K_BT—demonstrate random switching solely due to thermal noise. Such low barrier MTJs, used for random number generation (169, 170), exhibit high switching times on the order of milliseconds, leading to very slow operations and susceptibility to process variations (170). Additionally, owing to the low retention time of low barrier magnets, they are unsuitable as synapses but can be used as neurons (170).

Conclusion

The availability of large volumes of data from sensors all around us, coupled with an insatiable demand for handling new workloads for emerging applications, has transformed the landscape of AI. To meet this demand, AI models are growing at an alarming rate—as illustrated by the 5,000-fold increase in the size of natural language processing models over the last 4 years (Figure 10) (171).

The energy consumption of such implementations in today's hardware far exceeds the limits that edge devices, constrained by computing and energy, can handle. Hence, there is an urgent need to rethink the hardware, which can lead to quantum improvements in energy and latency while still maintaining accuracy. While today's CPUs, GPUs, TPUs, and field programmable gate arrays (FPGAs) have undergone updates to better handle matrix–matrix multiplication, sparsity in activation, pruning, and quantization of weights, they are still bottlenecked by the memory wall problem. Moreover, models such as transformers also require better and more effective handling of the attention mechanism in hardware, necessitating extensive calculations for softmax operations. Such operations require several trips to the main memory to fetch weights (parameters) and mathematical tables. These trips for fetching data are costly in terms of both energy and latency. To this end, it is believed that in-memory computing at various hierarchies of memory needs to be explored to determine where, how, and when such technology can be utilized to alleviate the memory wall and achieve quantum improvements in energy and latency (172).

On the other hand, for the exemplary application of autonomous drone navigation, an integrated approach to sensing and computation is necessary to perform all computations onboard rather than in the cloud, as cloud computing incurs expensive communication costs and leads to increased latency. To achieve the required SWAP (size, weight, and power) and accuracy, there is a need for "tiny" networks. To handle different types of inputs from various sensors and to efficiently compute both sequential and static tasks required for drone navigation, tiny hybrid networks utilizing both standard ANNs and more biologically plausible SNNs can be crucial. As discussed, SNNs, with their asynchronous event-driven computing, show great potential for extracting spatiotemporal features from event streams, especially for sequential applications, while ANNs are very effective for static tasks. Hence, from a hardware perspective, there is a need for a "converged platform" that can efficiently implement both ANNs and SNNs. While SNNs can use the in-memory computing primitives described earlier, there is an additional data structure—"membrane potential," responsible for keeping track of temporal information that must be fetched and updated at every time-step in SNNs. Computing in SNNs is asynchronous in nature, and, hence, hardware suitable for asynchronous or event-driven computation can reap the benefits of the sparse nature of spikes to achieve energy efficiency. While experimental hardware solutions such as Spinnaker, TrueNorth, and Loihi have been developed, there is still a path ahead to incorporate the



Source: Re-used from (171), with permission from NVIDIA.

Abbreviations: **BERT,** bidirectional encoder representations from transformers; **ELMo,** embeddings from language models; **GPT,** generative pre-training transformer; **LM,** language model; **NLG,** natural language generation

FIGURE 10
Trend of language model sizes over the past four years, showing an increase by a factor of 5,000. Re-used from (171), with permission from NVIDIA.

need for both ANNs and SNNs in a converged hardware platform. We believe that significant improvements in SWAP are only possible by co-designing the algorithms and hardware.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fsci.2025.1611658/full#supplementary-material

Statements

Author contributions

KR: Conceptualization, Formal Analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing.

AK: Conceptualization, Formal Analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

TS: Conceptualization, Formal Analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

SN: Conceptualization, Formal Analysis, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

DS: Conceptualization, Formal Analysis, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

US: Conceptualization, Formal Analysis, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

SR: Conceptualization, Formal Analysis, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

AnR: Conceptualization, Formal Analysis, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing.

ZW: Conceptualization, Formal Analysis, Methodology, Resources, Validation, Visualization, Writing – original draft, Writing – review & editing.

SS: Software, Validation, Visualization, Writing – original draft, Writing – review & editing, Conceptualization, Formal Analysis, Methodology.

C-KL: Conceptualization, Formal Analysis, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

ArR: Conceptualization, Formal Analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author. The data that support the findings of this work are publicly available and free to download from their respective sources. Code written to produce the results of this work is available upon reasonable request to the corresponding author.

Funding

The authors declared that financial support was received for this work and/or its publication. All authors received funding from the Center for the Co-Design of Cognitive Systems (COCOSYS), a JUMP 2.0 center (AWD-004311-S4). KR's work is also in part funded by the Semiconductor Research Corporation (2023-AI-3152) and National Science Foundation (2402983-CCF & 2023-AI-3152). The funders were not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

Conflict of interest

The authors declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The authors declared that generative AI was used in the creation of this manuscript. The AI tools ChatGPT-4o and 4o-mini from OpenAI were used to improve the vocabulary and formatting of text in certain sections of the manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

References

1. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in Neural Information Processing Systems 25 (NIPS 2012). New York, NY: Curran Associates, Inc. (2012). 1097–105. Available at: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

2. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput* (1997) 9(8):1735–80. doi: 10.1162/neco.1997.9.8.1735

3. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Guyon I, Von Luxburg U, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in Neural Information Processing Systems: Proceedings of the 31st International Conference on Neural Information Processing Systems. New York, NY: Curran Associates, Inc. (2007). 6000–10. Available at: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

4. Devlin J, Chang MW, Lee K, Toutanova K. BERT. Pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: human language technologies, volume 1 (long and short papers). Minneapolis, MN: Association for Computational Linguistics (2019). 4171–86. doi: 10.18653/v1/N19-1423

5. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language models are few-shot learners [article 159]. Advances in Neural Information Processing Systems 33 (NeurIPS 2020). New York, NY: Curran Associates, Inc. (2020). 1877–901. Available at: https://papers.nips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

6. Betker J, Goh G, Jing L, Brooks T, Wang J, Li L, et al. Improving image generation with better captions. *Comput Sci* (2023) 2(3):8. Available at: https://cdn.openai.com/papers/dall-e-3.pdf

7. Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models [article 574]. Advances in Neural Information Processing Systems 33 (NeurIPS 2020). New York, NY: Curran Associates, Inc. (2020) 33:6840–51. Available at: https://papers.nips.cc/paper_files/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

8. Roy K, Jaiswal A, Panda P. Towards spike-based machine intelligence with neuromorphic computing. *Nature* (2019) 575(7784):607–17. doi: 10.1038/s41586-019-1677-2

9. Rathi N, Chakraborty I, Kosta A, Sengupta A, Ankit A, Panda P, et al. Exploring neuromorphic computing based on spiking neural networks: algorithms to hardware. *ACM Comput Surv* (2023) 55(12):1–49. doi: 10.1145/3571155

10. LeCun Y. Generalization and network design strategies. In: Pfeifer R, Schreter Z, Fogelman-Soulie F, Steels L, editors. *Connectionism in perspective*. New York, NY: Elsevier (1989). 143–55.

11. Maass W. Networks of spiking neurons: the third generation of neural network models. *Neural Netw* (1997) 10(9):1659–71. doi: 10.1016/S0893-6080(97)00011-7

12. Yu S, Jiang H, Huang S, Peng X, Lu A. Compute-in-memory chips for deep learning: recent trends and prospects. *IEEE Circuits Syst Mag* (2021) 21(3):31–56. doi: 10.1109/MCAS.2021.3092533

13. Mrazek V, Sarwar SS, Sekanina L, Vasicek Z, Roy K. Design of power-efficient approximate multipliers for approximate artificial neural networks [article 81]. In: Proceedings of the 35th International Conference on Computer-Aided Design (ICCAD); 2016 Nov 7–10; Austin, TX, USA. New York, NY: Association for Computing Machinery (2016). 1–7. doi: 10.1145/2966986.2967021

14. Nahavandi S, Alizadehsani R, Nahavandi D, Mohamed S, Mohajer N, Rokonuzzaman M, et al. A comprehensive review on autonomous navigation. *ACM Comput Surv* (2025) 57(9):1–67. doi: 10.1145/3727642

15. Rezwan S, Choi W. Artificial intelligence approaches for UAV navigation: recent advances and future challenges. *IEEE Access* (2022) 10:26320–39. doi: 10.1109/ACCESS.2022.3157626

16. Haensch W, Raghunathan A, Roy K, Chakrabarti B, Phatak CM, Wang C, et al. Compute in-memory with non-volatile elements for neural networks: a review from a co-design perspective. *Adv Mater* (2023) 35(37):e2204944. doi: 10.1002/adma.202204944

17. Mannocci P, Farronato M, Lepri N, Cattaneo L, Glukhov A, Sun Z, et al. In-memory computing with emerging memory devices: status and outlook. *APL Mach Learn* (2023) 1(1):010902. doi: 10.1063/5.0136403

18. Chih Y-D, Lee P-H, Fujiwara H, Shih Y-C, Lee C-F, Naous R, et al. 16.4 an 89TOPS/W and 16.3TOPS/mm2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications. In: Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC); 2021 Feb 13–22; San Francisco, CA, USA. New York, NY: Association for Computing Machinery (2021). 252–4. doi: 10.1109/ISSCC42613.2021.9365766

19. Lichtsteiner P, Posch C, Delbruck T. A 128×128–120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J Solid-State Circuits* (2008) 43(2):566–76. doi: 10.1109/JSSC.2007.914337

20. Son B, Suh Y, Kim S, Jung H, Kim J-S, Shin C, et al. 4.1 A 640×480 dynamic vision sensor with a 9μm pixel and 300Meps address-event representation. In: 2017 IEEE International Solid-State Circuits Conference (ISSCC); 2017 Feb 5–9; San Francisco, CA, USA. New York, NY: Association for Computing Machinery (2017). 66–7. doi: 10.1109/ISSCC.2017.7870263

21. Gallego G, Delbrück T, Orchard G, Bartolozzi C, Taba B, Censi A, et al. Event-based vision: a survey. *IEEE Trans Pattern Anal Mach Intell* (2020) 44(1):154–80.

22. Boroujerdian B, Genc H, Krishnan S, Cui W, Faust A, Reddi V. Mavbench: micro aerial vehicle benchmarking. In: Proceedings of the 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2018). Fukuoka, Japan. 20-24 Ocotber 2018. New York, NY: IEEE (2018) 894–907. doi: 10.1109/MICRO.2018.00077

23. Gibaut W, Pereira L, Grassiotto F, Osorio A, Gadioli E, Munoz A, et al. Neurosymbolic AI and its taxonomy: a survey. arXiv [preprint] (2023). doi: 10.48550/arXiv.2305.08876

24. Wan Z, Liu CK, Yang H, Li C, You H, Fu Y, et al. Towards cognitive AI systems: a survey and prospective on neuro-symbolic AI. arXiv [preprint]. (2024). doi: 10.48550/arXiv.2401.01040

25. Wan Z, Chandramoorthy N, Swaminathan K, Chen PY, Reddi VJ, Raychowdhury A. BERRY: bit error robustness for energy-efficient reinforcement learning-based autonomous systems. In: 2023 60th ACM/IEEE Design Automation Conference (DAC); 2023 July 9–13; San Francisco, CA, USA. New York, NY: Association for Computing Machinery (2023). 1–6. doi: 10.1109/DAC56929.2023.10247999

26. Wan Z, Swaminathan K, Chen PY, Chandramoorthy N, Raychowdhury A. Analyzing and improving resilience and robustness of autonomous systems. In: Proceedings of the 41st IEEE/ACM International Conference on Computer-aided Design; 2022 Dec 22; San Diego, CA, USA. New York, NY: Association for Computing Machinery (2022). 1–9. doi: 10.1145/3508352.3561111

27. Saxena U, Roy K. McQueen: mixed precision quantization of early exit networks [abstract 511]. In: The 34th British Machine Vision Conference Proceedings. (BMVC) (2023). Available at: https://proceedings.bmvc2023.org/511/

28. Saxena U, Roy K. Partial-sum quantization for near ADC-less compute-in-memory accelerators. In: 2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED).; 2023 Aug 07–08; Vienna, Austria. New York, NY: IEEE (2023). 1–6. doi: 10.1109/ISLPED58423.2023.10244291

29. Saxena U, Sharify S, Roy K, Wang X. ResQ: mixed-precision quantization of large language models with low-rank residuals. In: Proceedings of the ICLR 2025 Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models (SCOPE). Open Review (2025). Available at: https://openreview.net/forum?id=sROOmwTHyW

30. Frantar E, Alistarh D. Optimal brain compression: a framework for accurate post-training quantization and pruning [article 323]. Advances in Neural Information Processing Systems 35 (NeurIPS 2022). New York, NY: Curran Associates, Inc. (2022). 4475–88. Available at: https://papers.nips.cc/paper_files/paper/2022/hash/1caf09c9f4e6b0150b06a07e77f2710c-Abstract-Conference.html

31. Cheng H, Zhang M, Shi JQ. A survey on deep neural network pruning: taxonomy, comparison, analysis, and recommendations. *IEEE Trans Pattern Anal Mach Intell* (2024) 46(12):10558–78. doi: 10.1109/TPAMI.2024.3447085

32. Bai H, Li Y. Structured sparsity in the NVIDIA ampere architecture and applications in search engines [online]. NVIDIA Developer (2023). Available at: https://developer.nvidia.com/blog/structured-sparsity-in-the-nvidia-ampere-architecture-and-applications-in-search-engines/

33. Zbikowski R. Fly like a fly [micro-air vehicle]. *IEEE Spectr* (2005) 42(11):46–51. doi: 10.1109/MSPEC.2005.1526905

34. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. New York, NY: IEEE (2016). 770–8. doi: 10.1109/CVPR.2016.90

35. Medsker LR, Jain LC. Recurrent neural networks. *Des Appl* (2001) 5(64–67):2.

36. Rathi N, Roy K. DIET-SNN: a low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *Proceedings of the IEEE Trans Neural Netw Learn Syst* (2023) 34(6):3174–82. doi: 10.1109/TNNLS.2021.3111897

37. Rathi N, Srinivasan G, Panda P, Roy K. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In: Proceedings of the ICLR 2020. The Eighth International Conference on Learning Representations. Open Review (2020). Available at: https://openreview.net/forum?id=B1xSperKvH&noteId=B1xSperKvH

38. Neftci EO, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks. *IEEE Signal Process Mag* (2019) 36(6):51–3. doi: 10.1109/MSP.2019.2931595

39. Sengupta A, Ye Y, Wang R, Liu C, Roy K. Going deeper in spiking neural networks: VGG and residual architectures. *Front Neurosci* (2019) 13:95. doi: 10.3389/fnins.2019.00095

40. Kosta AK, Roy K. Adaptive-SpikeNet: event-based optical flow estimation using spiking neural networks with learnable neuronal dynamics. In: Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA). London, United Kingdom. 29 May - 2 June 2023. New York, NY: IEEE (2023). 6021–7. doi: 10.1109/ICRA48891.2023.10160551

41. Lee C, Sarwar SS, Panda P, Srinivasan G, Roy K. Enabling spike-based backpropagation for training deep neural network architectures. *Front Neurosci* (2020) 14:119. doi: 10.3389/fnins.2020.00119

42. Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. In: Navab N, Hornegger J, Wells W, Frangi A, editors. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III. Berlin: Springer (2015). 234–41. doi: 10.1007/978-3-319-24574-4_28

43. Zhu AZ, Yuan L, Chaney K, Daniilidis K. EV-FlowNet: self-supervised optical flow estimation for event-based cameras. In: Kress-Gazit H, Srinivasa S, Howard S, Atanasov N, editors. Proceedings of the Robotics: Science and Systems XIV.; 2018 Jun. Pittsburgh, Pennsylvania. Robotics: Science and Systems online proceedings (2018). doi: 10.15607/RSS.2018.XIV.062

44. Zhu AZ, Yuan L, Chaney K, Daniilidis K. Unsupervised event-based learning of optical flow, depth, and egomotion. In: Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. New York, NY: IEEE (2019). 989–97. doi: 10.1109/CVPR.2019.00108

45. Lee C, Kosta AK, Zhu AZ, Chaney K, Daniilidis K, Roy K. Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks. In: Vedaldi A, Bischof H, Brox T, Frahm JM, editors. Computer Vision – ECCV 2020 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX. Berlin: Springer (2020). 366–82. doi: 10.1007/978-3-030-58526-6_22

46. Lee C, Kosta AK, Roy K. Fusion-FlowNet: energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures. In: Proceedings of the 2022 International Conference on Robotics and Automation (ICRA 2022) (ICRA); 2022 May 23–27; Philadelphia, PA, USA. New York, NY: IEEE (2022). 6504–10. doi: 10.1109/ICRA46639.2022.9811821

47. Negi S, Sharma D, Kosta AK, Roy K. Best of both worlds: hybrid SNN-ANN architecture for event-based optical flow estimation. In: Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2024 Oct 14–18; Abu Dhabi, United Arab Emirates. New York, NY: IEEE (2023). 2696–703. doi: 10.1109/IROS58592.2024.10802844

48. Nagaraj M, Liyanagedera CM, Roy K. Dotie-detecting objects through temporal isolation of events using a spiking architecture. In: Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA 2023) (ICRA); 2023 May 29–2023 Jun 02; London, United Kingdom. New York, NY: IEEE (2023). 4858–64. doi: 10.1109/ICRA48891.2023.10161164

49. Das Biswas S, Kosta A, Liyanagedera C, Apolinario M, Roy K. HALSIE: hybrid approach to learning segmentation by simultaneously exploiting image and event modalities. In: Proceedings of the 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV); 2024 Jan 03–08; Waikoloa, HI, USA. New York, NY: IEEE (2024). 5952–62. doi: 10.1109/WACV57701.2024.00586

50. Zoph B, Vasudevan V, Shlens J, Le QV. Learning transferable architectures for scalable image recognition. In: Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2018 Jun 18–23; Salt Lake City, UT, USA. New York, NY: IEEE (2018). 8697–710. doi: 10.1109/CVPR.2018.00907

51. Sabater A, Montesano L, Murillo AC. Event transformer. A sparse-aware solution for efficient event data processing. In: Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2022 Jun 19–20; New Orleans, LA, USA. New York, NY: IEEE (2022). 2677–86. doi: 10.1109/CVPRW56347.2022.00301

52. Gehrig M, Scaramuzza D. Recurrent vision transformers for object detection with event cameras. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2023 Jun 17–24; Vancouver, BC, Canada. New York, NY: IEEE (2023). 13884–93. doi: 10.1109/CVPR52729.2023.01334

53. Elliott DG, Snelgrove WM, Stumm M. Computational RAM: a memory-SIMD hybrid and its application to DSP. In: Proceedings of the 1992 IEEE Custom Integrated Circuits Conference; 1992 May 03–06; Boston, MA, USA. New York, NY: IEEE (1992). 30.6.1–4. doi: 10.1109/CICC.1992.591879

54. Gokhale M, Holmes B, Iobst K. Processing in memory: the Terasys massively parallel PIM array. *Computer* (1995) 28(4):23–31. doi: 10.1109/2.375174

55. Wulf WA, McKee SA. Hitting the memory wall: implications of the obvious. *ACM SIGARCH Comput Archit News* (1995) 23(1):20–4. doi: 10.1145/216585.216588

56. Horowitz M. 1.1 computing's energy problem (and what we can do about it). In: Proceedings of the 2014 IEEE International Solid-state Circuits Conference Digest of Technical Papers (ISSCC); 2014 Feb 09–13; San Francisco, CA, USA. New York, NY: IEEE (2014). doi: 10.1109/ISSCC.2014.6757323

57. Chakraborty I, Mustafa FA, Kim DE, Ankit A, Roy K. Geniex: a generalized approach to emulating non-ideality in memristive Xbars using neural networks. In: Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC); 2020 Jul 20–24; San Francisco, CA, USA. New York, NY: IEEE (2020). 1–6. doi: 10.1109/DAC18072.2020.9218688

58. Chakraborty I, Ali M, Ankit A, Jain S, Roy S, Sridharan S, et al. Resistive crossbars as approximate hardware building blocks for machine learning: opportunities and challenges. *Proc IEEE* (2020) 108(12):2276–310. doi: 10.1109/JPROC.2020.3003007

59. Ankit A, El Hajj IE, Chalamalasetti SR, Ndu G, Foltin M, Williams RS, et al. PUMA: a programmable ultra-efficient memristor-based accelerator for machine learning inference. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems; 2019 Apr 04; Providence, RI, USA. New York, NY: Association for Computing Machinery (2019). 715–31. doi: 10.1145/3297858.3304049

60. Jaiswal A, Chakraborty I, Agrawal A, Roy K. 8T SRAM cell as a multibit dot-product engine for beyond von Neumann computing. *IEEE Trans Very Large Scale Integr Syst* (2019) 27(11):2556–67. doi: 10.1109/TVLSI.2019.2929245

61. Valavi H, Ramadge PJ, Nestler E, Verma N. A 64-tile 2.4-MB in-memory-computing CNN accelerator employing charge-domain compute. *IEEE J Solid-State Circuits* (2019) 54(6):1789–99. doi: 10.1109/JSSC.2019.2899730

62. Wang J, Wang X, Eckert C, Subramaniyan A, Das R, Blaauw D, et al. A 28-nm compute sram with bit-serial logic/arithmetic operations for programmable in-memory vector computing. *IEEE J Solid-State Circuits* (2020) 55(1):76–86. doi: 10.1109/JSSC.2019.2939682

63. Crafton B, Spetalnick S, Yoon JH, Raychowdhury A. Statistical optimization of compute in-memory performance under device variation. In: Proceedings of the 2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED); 2021 Jul 26–28; Boston, MA, USA. New York, NY: IEEE (2021). 1–6. doi: 10.1109/ISLPED52811.2021.9502484

64. Crafton B, Wan Z, Spetalnick S, Yoon JH, Wu W, Tokunaga C, et al. Improving compute in-memory ECC reliability with successive correction. In: Proceedings of the 59th ACM/IEEE Design Automation Conference; 2022 Aug 23; San Francisco, California. New York, NY: Association for Computing Machinery (2022). 745–50. doi: 10.1145/3489517.3530526

65. Khan AI, Keshavarzi A, Datta S. The future of ferroelectric field-effect transistor technology. *Nat Electron* (2020) 3(10):588–97. doi: 10.1038/s41928-020-00492-7

66. Bae G, Bae D-I, Kang M, Hwang SM, Kim SS, Seo B, et al. 3nm GAA technology featuring multi-bridge-channel FET for low power and high performance applications. In: Proceedings of the 2018 International Electron Devices Meeting (IEDM 2018). San Francisco, CA, USA. 1-5 December 2018. New York, NY: IEEE (2018). 28.7.1–28.7.4. doi: 10.1109/IEDM.2018.8614629

67. Jain P, Arslan U, Sekhar M, Lin BC, Wei L, Sahu T, et al. 13.2 A 3.6 Mb 10.1 Mb/mm 2 embedded non-volatile ReRAM macro in 22nm FinFET technology with adaptive forming/set/reset schemes yielding down to 0.5 V with sensing time of 5ns at 0.7 V. In: Proceedings of the 2019 IEEE International Solid-State Circuits Conference (ISSCC); 2019 Feb 17–21; San Francisco, CA, USA. New York, NY: IEEE (2019). 212–4. doi: 10.1109/ISSCC.2019.8662393

68. Shih YC, Lee CF, Chang YA, Lee PH, Lin HJ, Chen YL, et al. A reflow-capable, embedded 8Mb STT-MRAM macro with 9ns read access time in 16nm FinFET logic CMOS process. In: Proceedings of the 2020 IEEE International Electron Devices Meeting (IEDM); 2020 Dec 12–18; San Francisco, CA, USA. New York, NY: IEEE (2020). 11.4.1–11.4.4. doi: 10.1109/IEDM13553.2020.9372115

69. Min D, Park J, Weber O, Wacquant F, Villaret A, Vandenbossche E, et al. 18nm FDSOI technology platform embedding PCM & innovative continuous-active construct enhancing performance for leading-edge MCU applications. In: Proceedings of the 2021 IEEE International Electron Devices Meeting (IEDM); 2021 Dec 11–16; San Francisco, CA, USA. New York, NY: IEEE (2021). 13.1.1–13.1.4. doi: 10.1109/IEDM19574.2021.9720542

70. Dünkel S, Trentzsch M, Richter R, Moll P, Fuchs C, Gehring O, et al. A FeFET based super-low-power ultra-fast embedded NVM technology for 22nm FDSOI and beyond. In: Proceedings of the 2017 IEEE International Electron Devices Meeting (IEDM); 2017 Dec 02-06; San Francisco, CA, USA. New York, NY: IEEE (2017). doi: 10.1109/IEDM.2017.8268425

71. Pentecost L, Hankin A, Donato M, Hempstead M, Wei G-Y, Brooks D. NVMExplorer: a framework for cross-stack comparisons of embedded non-volatile memories. In: Proceedings of the 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA); 2022 Apr 02–06; Seoul, Korea. New York, NY: IEEE (2021). 938–56. doi: 10.1109/HPCA53966.2022.00073

72. Song L, Zhuo Y, Qian X, Li H, Chen Y. GraphR: accelerating graph processing using ReRAM. In: Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA); 2018 Feb 24–28; Vienna, Austria. New York, NY: IEEE (2018). 531–43. doi: 10.1109/HPCA.2018.00052

73. Dutta S, Schafer C, Gomez J, Ni K, Joshi S, Datta S. Supervised learning in all FeFET-based spiking neural network: opportunities and challenges. *Front Neurosci* (2020) 14:634. doi: 10.3389/fnins.2020.00634

74. Yang X, Yan B, Li H, Chen Y. ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration. In: Proceedings of the 39th International Conference on Computer-Aided Design; 2020 Dec 17; Virtual Event, USA. New York, NY: Association for Computing Machinery (2020). 1–9. doi: 10.1145/3400302.3415640

75. Li W, Manley M, Read J, Kaul A, Bakir MS, Yu S. H3DAtten: heterogeneous 3-D integrated hybrid analog and digital compute-in-memory accelerator for vision transformer self-attention. *IEEE Trans Very Large Scale Integr (VLSI) Syst* (2023) 31(10):1592–602. doi: 10.1109/TVLSI.2023.3299509

76. Langenegger J, Karunaratne G, Hersche M, Benini L, Sebastian A, Rahimi A. In-memory factorization of holographic perceptual representations. *Nat Nanotechnol* (2023) 18(5):479–85. doi: 10.1038/s41565-023-01357-8

77. Wan Z, Liu CK, Ibrahim M, Yang H, Spetalnick S, Krishna T, et al. H3DFact: heterogeneous 3D integrated CIM for factorization with holographic perceptual representations. In: Proceedings of the 2024 Design, automation & test in Europe

conference & exhibition (DATE); 2024 Mar 25–27; Valencia, Spain. New York, NY: IEEE (2024). 1–6. doi: 10.23919/DATE58400.2024.10546582

78. Chen PY, Yu S. Compact modeling of RRAM devices and its applications in 1T1R and 1S1R array design. *IEEE Trans Electron Devices* (2015) 62(12):4022–8. doi: 10.1109/TED.2015.2492421

79. Chen WH, Li KX, Lin WY, Hsu KH, Li PY, Yang C-H, et al. A 65nm 1mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors. In: Proceedings of the 2018 IEEE International Solid-State Circuits Conference (ISSCC); 2018 Feb 11–15; San Francisco, CA, USA. New York, NY: IEEE (2018). 494–6. doi: 10.1109/ISSCC.2018.8310400

80. Xue CX, Chen WH, Liu JS, Li JF, Lin WY, Wei-En Lin JHW, et al. 24.1 a 1mb multibit ReRAM computing-in-memory macro with 14.6 ns parallel MAC computing time for CNN based ai edge processors. In: Proceedings of the 2019 IEEE International Solid-State Circuits Conference (ISSCC); 2019 Feb 17–21; San Francisco, CA, USA. New York, NY: IEEE (2019). 388–90. doi: 10.1109/ISSCC.2019.8662395

81. Xue CX, Huang TY, Liu JS, Chang TW, Kao HY, Wang JH, et al. 15.4 a 22nm 2mb ReRAM compute-in-memory macro with 121–28TOPS/W for multibit MAC computing for tiny AI edge devices. In: Proceedings of the 2020 IEEE International Solid-State Circuits Conference (ISSCC); 2020 Feb 16–20; San Francisco, CA, USA. New York, NY: IEEE (2020). 244–6. doi: 10.1109/ISSCC19947.2020.9063078

82. Liu Q, Gao B, Yao P, Wu D, Chen J, Pang Y, et al. 33.2 a fully integrated analog ReRAM based 78.4 TOPS/W compute-in-memory chip with fully parallel MAC computing. In: Proceedings of the 2020 IEEE International Solid-State Circuits Conference (ISSCC); 2020 Feb 16–20; San Francisco, CA, USA. New York, NY: IEEE (2020). 500–2. doi: 10.1109/ISSCC19947.2020.9062953

83. Yoon JH, Chang M, Khwa WS, Chih YD, Chang MF, Raychowdhury A. 29.1 a 40nm 64kb 56.67 TOPS/W read-disturb-tolerant compute-in-memory/digital RRAM macro with active-feedback-based read and *in-situ* write verification. In: Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC); 2021 Feb 13–22; San Francisco, CA, USA. New York, NY: IEEE (2021). 404–6. doi: 10.1109/ISSCC42613.2021.9365926

84. Xue CX, Hung J-M, Kao HY, Huang YH, Huang SP, Chang FC, et al. 16.1 a 22nm 4MB 8b-precision ReRAM computing-in-memory macro with 11.91 to 195.7 TOPS/W for tiny AI edge devices. In: Proceedings of the 2021 IEEE International Solid-State Circuits Conference (ISSCC); 2021 Feb 13–22; San Francisco, CA, USA. New York, NY: IEEE (2021). 245–7. doi: 10.1109/ISSCC42613.2021.9365769

85. Hung J-M, Wen TH, Huang YH, Huang SP, Chang FC, Su CI, et al. 8-b precision 8-mb ReRAM compute-in-memory macro using direct-current-free time-domain readout scheme for AI edge devices. *IEEE J Solid-State Circuits* (2023) 58(1):303–15. doi: 10.1109/JSSC.2022.3200515

86. Spetalnick SD, Chang M, Crafton B, Khwa WS, Chih YD, Chang MF, et al. A 40nm 64KB 26.56 TOPS/W 2.37 Mb/mm2 RRAM binary/compute-in-memory macro with 4.23x improvement in density and >75% use of sensing dynamic range. In: Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC); 2022 Feb 20–26; San Francisco, CA, USA. New York, NY: IEEE (2022). 1–3. doi: 10.1109/ISSCC42614.2022.9731725

87. Correll JM, Jie L, Song S, Lee S, Zhu J, Tang W, et al. An 8-bit 20.7 TOPS/W multi-level cell ReRAM-based compute engine. In: Proceedings of the 2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits); 2022 Jun 12–17; Honolulu, HI, USA. New York, NY: IEEE (2022). 264–5. doi: 10.1109/VLSITechnologyandCir46769.2022.9830490

88. Spetalnick SD, Chang M, Konno S, Crafton B, Lele AS, Khwa WS, et al. A 2.38 MCells/mm2 9.81–350 TOPS/W RRAM compute-in-memory macro in 40nm CMOS with hybrid offset/IOFF cancellation and ICELL RBLSL drop mitigation. In: Proceedings of the 2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits); 2023 Jun 11–16; Kyoto, Japan. New York, NY: IEEE (2023). 1–2. doi: 10.23919/VLSITechnologyandCir57934.2023.10185424

89. Huang WH, Wen TH, Hung J-M, Khwa WS, Lo YC, Jhang CJ, et al. A nonvolatile al-edge processor with 4MB SLC-MLC hybrid-mode ReRAM compute-in-memory macro and 51.4-251TOPS/W. In: Proceedings of the 2023 IEEE International Solid-State Circuits Conference (ISSCC); 2023 Feb 19–23; San Francisco, CA, USA. New York, NY: IEEE (2023). 15–7. doi: 10.1109/ISSCC42615.2023.10067610

90. Kim S, Kim S, Um S, Kim S, Kim K, Yoo H-J. Neuro-CIM: ADC-less neuromorphic computing-in-memory processor with operation gating/stopping and digital–analog networks. *IEEE J Solid-State Circuits* (2023) 58(10):2931–45. doi: 10.1109/JSSC.2023.3273238

91. Sharma T, Wang C, Agrawal A, Roy K. Enabling robust SOT-MTJ crossbars for machine learning using sparsity-aware device-circuit co-design. In: Proceedings of the 2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED); 2021 Jul 26–28; Boston, MA, USA. New York, NY: IEEE (2021). 1–6. doi: 10.1109/ISLPED52811.2021.9502492

92. Chang M, Spetalnick SD, Crafton B, Khwa WS, Chih YD, Chang MF, et al. A 40nm 60.64 TOPS/W ECC-capable compute-in-memory/digital 2.25 MB/768KB RRAM/SRAM system with embedded cortex M3 microprocessor for edge recommendation systems. In: Proceedings of the 2022 IEEE International Solid-State Circuits Conference (ISSCC); 2022 Feb 20–26; San Francisco, CA, USA. New York, NY: IEEE (2022). 1–3. doi: 10.1109/ISSCC42614.2022.9731679

93. Chang M, Lele AS, Spetalnick SD, Crafton B, Konno S, Wan Z, et al. A 73.53 TOPS/W 14.74 TOPS heterogeneous RRAM in-memory and SRAM near-memory SoC for hybrid frame and event-based target tracking. In: Proceedings of the 2023 IEEE International Solid-State Circuits Conference (ISSCC); 2023 Feb 19–23; San Francisco, CA, USA. New York, NY: IEEE (2023). 426–8. doi: 10.1109/ISSCC42615.2023.10067544

94. Lele AS, Chang M, Spetalnick SD, Crafton B, Konno S, Wan Z, et al. A heterogeneous RRAM in-memory and SRAM near-memory SoC for fused frame and event-based target identification and tracking. *IEEE J Solid-State Circuits* (2024) 59(1):52–64. doi: 10.1109/JSSC.2023.3297411

95. Singh S, Sarma A, Jao N, Pattnaik A, Lu S, Yang K, et al. Nebula: a neuromorphic spin-based ultra-low power architecture for SNNS and ANNS. In: Proceedings of the 47th Annual International Symposium on Computer Architecture (ISCA); 2020 May 30–2020 Jun 03. New York, NY: IEEE (2020). 363–76. doi: 10.1109/ISCA45697.2020.00039

96. Chen Y, Lu L, Kim B, Kim TT-H. Reconfigurable 2T2R ReRAM architecture for versatile data storage and computing in-memory. *IEEE Trans Very Large Scale Integr (VLSI) Syst* (2020) 28(12):2636–49. doi: 10.1109/TVLSI.2020.3028848

97. Shou S, Liu CK, Yun S, Wan Z, Ni K, Imani M, et al. See MCAM: scalable multi-bit FeFET content addressable memories for energy efficient associative search. In: Proceedings of the 42nd IEEE/ACM International Conference on Computer-aided Design; 2023 Oct 28–2023 Nov 02; San Francisco, CA, USA. New York, NY: IEEE (2023). 1–9. doi: 10.1109/ICCAD57390.2023.10323738

98. Wan W, Kubendran R, Eryilmaz SB, Zhang W, Liao Y, Wu D, et al. 33.1 a 74 TMACS/W CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and *in-situ* transposable weights for probabilistic graphical models. In: Proceedings of the 2020 IEEE International Solid-State Circuits Conference-(ISSCC); 2020 Feb 16–20; San Francisco, CA, USA. New York, NY: IEEE (2020). 498–500. doi: 10.1109/ISSCC19947.2020.9062979

99. Shin D, Lee J, Lee J, Yoo H-J. 14.2 DNPU: an 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks. In: Proceedings of the 2017 IEEE International Solid-State Circuits Conference (ISSCC); 2017 Feb 05–09; San Francisco, CA, USA. New York, NY: IEEE (2017). 240–1. doi: 10.1109/ISSCC.2017.7870350

100. Zahedi M, Mayahinia M, Lebdeh MA, Wong S, Hamdioui S. Efficient organization of digital periphery to support integer datatype for memristor-based CIM. In: Proceedings of the 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI); 2020 Jul 06–08; Limassol, Cyprus. New York, NY: IEEE (2020). 216–21. doi: 10.1109/ISVLSI49217.2020.00047

101. Liu CK, Chen H, Imani M, Ni K, Kazemi A, Laguna AF, et al. Cosime: FeFET based associative memory for in-memory cosine similarity search. In: Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design; 2022 Dec 22; San Diego, California. New York, NY: Association for Computing Machinery (2022). 1–9. doi: 10.1145/3508352.3549412

102. Zhang Q, An H, Fan Z, Wang Z, Li Z, Wang G, et al. A 22nm 3.5 TOPS/W flexible micro-robotic vision soc with 2MB eMRAM for fully-on-chip intelligence. In: Proceedings of the 2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits); 2022 Jun 12–17; Honolulu, HI, USA. New York, NY: IEEE (2022). 72–3. doi: 10.1109/VLSITechnologyandCir46769.2022.9830340

103. Rossi D, Conti F, Eggiman M, Mach S, Di Mauro A, Guermandi M, et al. 4.4 a 1.3 TOPS/W@ 32GOPS fully integrated 10-core SoC for IoT end-nodes with 1.7 μW cognitive wake-up from MRAM-based state-retentive sleep mode. In: Proceedings of the 2021 IEEE International Solid-State Circuits Conference (ISSCC); 2021 Feb 13–22; San Francisco, CA, USA. New York, NY: IEEE (2021). 60–2. doi: 10.1109/ISSCC42613.2021.9365939

104. Kang M, Keel MS, Shanbhag NR, Eilert S, Curewitz K. An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM. In: Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2014 May 04–09; Florence, Italy. New York, NY: IEEE (2014). 8326–30. doi: 10.1109/ICASSP.2014.6855225

105. Zhang J, Wang Z, Verma N. A machine-learning classifier implemented in a standard 6T SRAM array. In: Proceedings of the 2016 IEEE symposium on VLSI circuits (VLSI-circuits); 2016 Jun 15–17; Honolulu, HI. New York, NY: IEEE (2016). 1–2. doi: 10.1109/VLSIC.2016.7573556

106. Ali M, Jaiswal A, Kodge S, Agrawal A, Chakraborty I, Roy K. . IMAC: in-memory multi-bit multiplication and accumulation in 6T SRAM array. *IEEE Trans Circuits Syst I* (2020) 67(8):2521–31. doi: 10.1109/TCSI.2020.2981901

107. Mori H, Zhao WC, Cheng-En Lee CFL, Hsu YH, Chuang CK, Hashizume T, et al. A 4nm 6163-TOPS/W/b 4790–TOPS/mm2/b SRAM based digital-computing-in-memory macro supporting bit-width flexibility and simultaneous MAC and weight update. In: Proceedings of the 2023 IEEE International Solid-State Circuits Conference (ISSCC); 2023 Feb 19–23; San Francisco, CA, USA. New York, NY: IEEE (2023). 132–4. doi: 10.1109/ISSCC42615.2023.10067555

108. Si X, Chen JJ, Tu YN, Huang WH, Wang JH, Chiu YC, et al. A twin-8T SRAM computation-in-memory unit-macro for multibit CNN-based AI edge processors. *IEEE J Solid-State Circuits* (2020) 55(1):189–202. doi: 10.1109/JSSC.2019.2952773

109. Biswas A, Chandrakasan AP. Conv-RAM: an energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications. In: Proceedings of the 2018 IEEE International Solid-State Circuits Conference-(ISSCC); 2018 Feb 11–15; San Francisco, CA, USA. New York, NY: IEEE (2018). 488–90. doi: 10.1109/ISSCC.2018.8310397

110. Fujiwara H, Mori H, Zhao WC, Chuang MC, Naous R, Chuang CK, et al. A 5-nm 254-TOPS/W 221-TOPS/mm 2 fully-digital computing-in-memory macro supporting wide-range dynamic-voltage- frequency scaling and simultaneous MAC and write operations. In: Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC); 2022 Feb 20–26; San Francisco, CA, USA. New York, NY: IEEE (2022). 1–3. doi: 10.1109/ISSCC42614.2022.9731754

111. Agrawal A, Jaiswal A, Roy D, Han B, Srinivasan G, Ankit A, et al. Xcel-RAM: accelerating binary neural networks in high-throughput SRAM compute arrays. *IEEE Trans Circuits Syst I* (2019) 66(8):3064–76. doi: 10.1109/TCSI.2019.2907488

112. Srivastava P, Kang M, Gonugondla SK, Lim S, Choi J, Adve V, et al. Promise: an end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms. In: Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA); 2018 Jun 01–06; Los Angeles, CA, USA. New York, NY: IEEE (2018) ACM. 43–56. doi: 10.1109/ISCA.2018.00015

113. Wu PC, Su JW, Hong LY, Ren JS, Chien C-H, Chen HY, et al. A 22nm 832Kb hybrid-domain floating-point SRAM in-memory-compute macro with 16.2-70.2 TFLOPS/W for high-accuracy AI-edge devices. In: Proceedings of the 2023 IEEE International Solid-State Circuits Conference (ISSCC); 2023 Feb 19–23; San Francisco, CA, USA. New York, NY: IEEE (2023). 126–8. doi: 10.1109/ISSCC42615.2023.10067527

114. Ali M, Chakraborty I, Choudhary S, Chang M, Kim DE, Raychowdhury A, et al. A 65 nm 1.4-6.7 TOPS/W adaptive-SNR sparsity-aware CIM core with load balancing support for DL workloads. In: Proceedings of the 2023 IEEE Custom Integrated Circuits Conference (CICC); 2023 Apr 23–26; San Antonio, TX, USA. New York, NY: IEEE (2023). 1–2. doi: 10.1109/CICC57935.2023.10121243

115. Huang S, Ankit A, Silveira P, Antunes R, Chalamalasetti SR, El Hajj I, et al. Mixed precision quantization for ReRAM-based DNN inference accelerators. In: Proceedings of the 26th Asia and South Pacific Design Automation Conference; 2021 Jan 29; Tokyo, Japan. New York, NY: Association for Computing Machinery (2021). 372–7. doi: 10.1145/3394885.3431554

116. Kim Y, Kim H, Kim J-J. Extreme partial-sum quantization for analog computing-in-memory neural network accelerators. *J Emerg Technol Comput Syst* (2022) 18(4):1–19. doi: 10.1145/3528104

117. Kim H, Kim Y, Ryu S, Kim J-J. Algorithm/hardware co-design for in-memory neural network computing with minimal peripheral circuit overhead. In: Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC); 2020 Jul 20–24; San Francisco, CA, USA. New York, NY: IEEE (2020). 1–6. doi: 10.1109/DAC18072.2020.9218657

118. Azamat A, Asim F, Lee J. Quarry: quantization-based ADC reduction for ReRAM-based deep neural network accelerators. In: Proceedings of the 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD); 2021 Nov 01–04; Munich, Germany. New York, NY: IEEE (2021). 1–7. doi: 10.1109/ICCAD51958.2021.9643502

119. Saxena U, Chakraborty I, Roy K. Towards ADC-less compute-in-memory accelerators for energy efficient deep learning. In: Proceedings of the 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE); 2022 Mar 14–23; Antwerp, Belgium. New York, NY: IEEE (2022). 624–7. doi: 10.23919/DATE54114.2022.9774573

120. Negi S, Saxena U, Sharma D, Roy K. HCIM: ADC-less hybrid analog-digital compute in memory accelerator for deep learning workloads. In: Proceedings of the 30th Asia and South Pacific Design Automation Conference; 2025 Mar 04; Tokyo, Japan. New York, NY: Association for Computing Machinery (2025). 648–55. doi: 10.1145/3658617.3697572

121. Esser SK, McKinstry JL, Bablani D, Appuswamy R, Modha DS. Learned step size quantization. Proceedings of the ICLR 2020. The Eighth International Conference on Learning Representations. Open Review (2020). Available at: https://openreview.net/forum?id=rkgO66VKDS

122. Apolinario MPE, Kosta AK, Saxena U, Roy K. Hardware/software co-design with adc-less in-memory computing hardware for spiking neural networks. *IEEE Trans Emerg Top Comput* (2024) 12(1):35–47. doi: 10.1109/TETC.2023.3316121

123. Seshadri V, Lee D, Mullins T, Hassan H, Boroumand A, Kim J, et al. Ambit: in-memory accelerator for bulk bitwise operations using commodity DRAM technology. In: Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO); 2017 Oct 14; Cambridge, Massachusetts. New York, NY: Association for Computing Machinery (2017). 273–87. doi: 10.1145/3123939.3124544

124. Li S, Niu D, Malladi KT, Zheng H, Brennan B, Xie Y. DRISA: a DRAM-based reconfigurable *in-situ* accelerator. In: Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO); 2017 Oct 14; Cambridge, Massachusetts. New York, NY: Association for Computing Machinery (2017). 288–301. doi: 10.1145/3123939.3123977

125. Deng Q, Jiang L, Zhang Y, Zhang M, Yang J. DrACC: a DRAM based accelerator for accurate CNN inference. In: Proceedings of the 55th ACM/ESDA/IEEE Design Automation Conference (DAC); San Francisco, California. New York, NY: Association for Computing Machinery (2018). 1–6. doi: 10.1109/DAC.2018.8465866

126. Xin X, Zhang Y, Yang J. ELP2IM: efficient and low power bitwise operation processing in DRAM. In: IEEE International Symposium on High Performance Computer Architecture (HPCA); 2020 Feb 22–26; San Diego, CA, USA. New York, NY: IEEE (2020). 303–14. doi: 10.1109/HPCA47549.2020.00033

127. Hajinazar N, Oliveira GF, Gregorio S, Ferreira JD, Ghiasi NM, Patel M, et al. SIMDRAM: a framework for bit-serial SIMD processing using DRAM. In: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2021); 2021 Apr 17; Virtual, USA. New York, NY: Association for Computing Machinery (2021). 329–45. doi: 10.1145/3445814.3446749

128. Ali MF, Jaiswal A, Roy K. In-memory low-cost bit-serial addition using commodity dram technology. *IEEE Trans Circuits Syst I* (2020) 67(1):155–65. doi: 10.1109/TCSI.2019.2945617

129. Roy S, Ali M, Raghunathan A. PIM-DRAM: accelerating machine learning workloads using processing in commodity DRAM. *IEEE J Emerg Sel Top Circuits Syst* (2021) 11(4):701–10. doi: 10.1109/JETCAS.2021.3127517

130. He M, Song C, Kim I, Jeong C, Kim S, Park I, et al. Newton: a DRAM-maker's accelerator-in-memory (AiM) architecture for machine learning. In: Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). 2020 Oct 17–21; Athens, Greece. New York, NY: IEEE (2020). 372–85. doi: 10.1109/MICRO50266.2020.00040

131. Ke L, Gupta U, Cho BY, Brooks D, Chandra V, Diril U, et al. RecNMP: accelerating personalized recommendation with near-memory processing. In: Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA); 2020 May 30–2020 June 03; Valencia, Spain. New York, NY: IEEE (2020). 790–803. doi: 10.1109/ISCA45697.2020.00070

132. Jeddeloh J, Keeth B. Hybrid memory cube new DRAM architecture increases density and performance. In: Proceedings of the 2012 Symposium on VLSI Technology (VLSIT); 2012 Jun 12–14; Honolulu, HI, USA. New York, NY: IEEE (2012). 87–8. doi: 10.1109/VLSIT.2012.6242474

133. Kim D, Kung J, Chai S, Yalamanchili S, Mukhopadhyay S. Neurocube: a programmable digital neuromorphic architecture with high-density 3D memory. *ACM SIGARCH Comput Archit News* (2016) 44(3):380–92. doi: 10.1109/ISCA.2016.41

134. Gao M, Pu J, Yang X, Horowitz M, Kozyrakis C. Tetris: scalable and efficient neural network acceleration with 3d memory. *ACM SIGARCH Comput Archit News* (2017) 45(1):751–64. doi: 10.1145/3093337.3037702

135. Lee S, Kang SH, Lee J, Kim H, Lee E, Seo S, et al. Hardware architecture and software stack for PIM based on commercial DRAM technology: industrial product. In: Proceedings of the 2021 48th Annual International Symposium on Computer Architecture (ISCA); 2021 Jun 14–19; Valencia, Spain. New York, NY: IEEE (2021). 43–56. doi: 10.1109/ISCA52012.2021.00013

136. Xiang Y, Huang P, Han R, Li C, Wang K, Liu X, et al. Efficient and robust spike-driven deep convolutional neural networks based on nor flash computing array. *IEEE Trans Electron Devices* (2020) 67(6):2329–35. doi: 10.1109/TED.2020.2987439

137. Choi WH, Chiu PF, Ma W, Hemink G, Hoang TT, Lueker-Boden M, et al. An in-flash binary neural network accelerator with SLC NAND flash array. In: Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS); 2020 Oct 12–14; Seville, Spain. New York, NY: IEEE (2020). 1–5. doi: 10.1109/ISCAS45731.2020.9180920

138. Kang M, Kim H, Shin H, Sim J, Kim K, Kim LS. S-flash: a NAND flash-based deep neural network accelerator exploiting bit-level sparsity. *IEEE Trans Comput* (2021) 71(6):1-;. doi: 10.1109/TC.2021.3082003

139. Gao C, Xin X, Lu Y, Zhang Y, Yang J, Shu J. Parabit: processing parallel bitwise operations in NAND flash memory based SSDs. In: Proceedings of the Micro-54th Annual IEEE/ACM International Symposium on Microarchitecture; 2021 Oct 17; Virtual Event, Greece. New York, NY: Association for Computing Machinery (2021). 59–70. doi: 10.1145/3466752.348007

140. Lee H, Kim M, Min D, Kim J, Back J, Yoo H, et al. 3D-FPIM: an extreme energy-efficient DNN acceleration system using 3D NAND flash-based *in-situ* PIM unit. In: Proceedings of the 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO); 2022 Oct 01–05; Chicago, IL, USA. New York, NY: IEEE (2022). 1359–76. doi: 10.1109/MICRO56248.2022.00093

141. Harrison J, Kubaska T, Story S, Tang PTP. The computation of transcendental functions on the IA-64 architecture. *Intel Technol J Q* (1999) 4:1–7. Available at: https://www.cl.cam.ac.uk/~jrh13/papers/itj.pdf

142. Lee D, Fong X, Roy K. R-MRAM: a ROM-embedded STT MRAM cache. *IEEE Electron Dev Lett* (2013) 34(10):1256–8. doi: 10.1109/LED.2013.2279137

143. Lee D, Roy K. Area efficient ROM-embedded SRAM cache. *IEEE Trans Very Large Scale Integr (VLSI) Syst* (2012) 21(9):1583–95. doi: 10.1109/TVLSI.2012.2217514

144. Dutta T. Energy efficient hardware for neural network applications [Master's thesis]. West Lafayette, IN: Purdue University School of Electrical and Computer Engineering (2023). doi: 10.25394/PGS.22779695.v1

145. Kim DE, Sharma T, Roy K. Hardware-software co-design for accelerating transformer inference leveraging compute-in-memory. *IEEE Trans Circuits Syst Artif Intell* (Early Access). (2025). doi: 10.1109/TCASAI.2025.3601975

146. Akopyan F, Sawada J, Cassidy A, Alvarez-Icaza R, Arthur J, Merolla P, et al. Truenorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. *IEEE Trans Comput-Aid Des Integr Circuits Syst* (2015) 34(10):1537–57. doi: 10.1109/TCAD.2015.2474396

147. Davies M, Srinivasa N, Lin T-H, Chinya G, Cao Y, Choday SH, et al. Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* (2018) 38(1):82–99. doi: 10.1109/MM.2018.112130359

148. Orchard G, Frady EP, Rubin DBD, Sanborn S, Shrestha SB, Sommer FT, et al. Efficient neuromorphic signal processing with Loihi 2. In: Proceedings of the 2021 IEEE Workshop on Signal Processing Systems (SiPS); 2021 Oct 19–21; Coimbra, Portugal. New York, NY: IEEE (2021). 254–9. doi: 10.1109/SiPS52927.2021.00053

149. Benjamin BV, Gao P, McQuinn E, Choudhary S, Chandrasekaran AR, Bussat JM, et al. Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc IEEE* (2014) 102(5):699–716. doi: 10.1109/JPROC.2014.2313565

150. Schemmel J, Briiderle D, Griibl A, Hock M, Meier K, Millner S. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In: Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS); 2010 May 30–2010 Jun 2; Paris, France. New York, NY: IEEE (2010). 1947–50. doi: 10.1109/ISCAS.2010.5536970

151. Furber SB, Galluppi F, Temple S, Plana LA. The spinnaker project. *Proc IEEE* (2014) 102(5):652–65. doi: 10.1109/JPROC.2014.2304638

152. Huo D, Zhang J, Dai X, Zhang J, Qian C, Tang KT, et al. ANP-G: A 28nm 1.04 pJ/SOP sub-mm2 spiking and back-propagation hybrid neural network asynchronous olfactory processor enabling few-shot class-incremental on-chip learning. In: Proceedings of the 2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits); 2023 Jun 11–16; Kyoto, Japan. New York, NY: IEEE (2023). 1–2. doi: 10.23919/VLSITechnologyandCir57934.2023.10185410

153. Frenkel C, Indiveri G. Reckon: a 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales. In: Proceedings of the 2022 IEEE International Solid-State Circuits Conference (ISSCC); 2022 Feb 20–26; San Francisco, CA, USA. New York, NY: IEEE (2022). 1–3. doi: 10.1109/ISSCC42614.2022.9731734

154. Zhang J, Huo D, Zhang J, Qian C, Liu Q, Pan L, et al. 22.6 ANP-I: a 28nm 1.5 pJ/SOP asynchronous spiking neural network processor enabling sub-o. 1 μj/sample on-chip learning for edge-AI applications. In: Proceedings of the 2023 IEEE International Solid-State Circuits Conference (ISSCC); 2023 Feb 19–23; San Francisco, CA, USA. New York, NY: IEEE (2023). 21–3.

155. Sharma D, Negi S, Dutta T, Agrawal A, Roy K. A 65 nm 5 TOPS/W digital CIM accelerator with reconfigurable precision and temporal pipelining for Spiking Neural Networks. In: Proceedings of the 2025 IEEE European Solid-State Electronics Research Conference (ESSERC). 8–11 September 2025. Munich (Germany). New York, NY: IEEE (2025). 5–8. doi: 10.1109/ESSERC66193.2025.11214115

156. Agrawal A, Ali M, Koo M, Rathi N, Jaiswal A, Roy K. Impulse: a 65-nm digital compute-in-memory macro with fused weights and membrane potential for spike-based sequential learning tasks. *IEEE Solid-State Circuits Lett* (2021) 4:137–40. doi: 10.1109/LSSC.2021.3092727

157. Narayanan S, Taht K, Balasubramonian R, Giacomin E, Gaillardon PE. Spinalflow: an architecture and dataflow tailored for spiking neural networks. In: Proceedings of the 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA); 2020 May 30–2020 Jun 3; Valencia, Spain; New York, NY: IEEE (2020). 349–62. doi: 10.1109/ISCA45697.2020.00038

158. Sharma D, Ankit A, Roy K. Identifying efficient dataflows for spiking neural networks. In: Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design. New York, NY: Association for Computing Machinery (2022). 1–6. doi: 10.1145/3531437.3539704

159. Yik J, Van den Berghe K, den Blanken D, Bouhadjar Y, Fabre M, Hueber P, et al. The neurobench framework for benchmarking neuromorphic computing algorithms and systems. *Nat Commun* (2025) 16:1545. doi: 10.1038/s41467-025-56739-4

160. Srinivasan G, Lee C, Sengupta A, Panda P, Sarwar SS, Roy K. Training deep spiking neural networks for energy-efficient neuromorphic computing. In: Proceedings of the 2020 ICASSP 2020–2020 ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2020 May 4–8; Barcelona, Spain. New York, NY: IEEE (2020). 8549–53. doi: 10.1109/ICASSP40776.2020.9053914

161. Sengupta A, Parsa M, Han B, Roy K. Probabilistic deep spiking neural systems enabled by magnetic tunnel junction. *IEEE Trans Electron Devices* (2016) 63(7):2963–70. doi: 10.1109/TED.2016.2568762

162. Sengupta A, Panda P, Wijesinghe P, Kim Y, Roy K. Magnetic tunnel junction mimics stochastic cortical spiking neurons. *Sci Rep* (2016) 6(1):30039. doi: 10.1038/srep30039

163. Roy D, Chakraborty I, Roy K. Scaling deep spiking neural networks with binary stochastic activations. In: Proceedings of the 2019 IEEE International Conference on Cognitive Computing (ICCC); 2019 July 8–13; Milan, Italy. New York, NY: IEEE (2019). 50–8. doi: 10.1109/ICCC.2019.00020

164. Bi GQ, Poo MM. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J Neurosci* (1998) 18(24):10464–72. doi: 10.1523/JNEUROSCI.18-24-10464.1998

165. Raiko T, Berglund M, Alain G, Dinh L. Techniques for learning binary stochastic feedforward neural networks. In: Proceedings of the ICLR 2015. The Third International Conference on Learning Representations. (2015). Available at: 10.48550/arXiv.1406.2989

166. Jaiswal A, Fong X, Roy K. Comprehensive scaling analysis of current induced switching in magnetic memories based on in-plane and perpendicular anisotropies. *IEEE J Emerg Sel Top Circuits Syst* (2016) 6(2):120–33. doi: 10.1109/JETCAS.2016.2547698

167. Sun W, Gao B, Chi M, Xia Q, Yang JJ, Qian H, et al. Understanding memristive switching via in situ characterization and device modeling. *Nat Commun* (2019) 10(1):3453. doi: 10.1038/s41467-019-11411-6

168. Tuma T, Pantazi A, Le Gallo M, Sebastian A, Eleftheriou E. Stochastic phase-change neurons. *Nat Nanotechnol* (2016) 11(8):693–9. doi: 10.1038/nnano.2016.70

169. Camsari KY, Sutton BM, Datta S. P-bits for probabilistic spin logic. *Appl Phys Rev* (2019) 6(1). doi: 10.1063/1.5055860

170. Liyanagedera CM, Sengupta A, Jaiswal A, Roy K. Stochastic spiking neural networks enabled by magnetic tunnel junctions: from nontelegraphic to telegraphic switching regimes. *Phys Rev Appl* (2017) 8(6):064017. doi: 10.1103/PhysRevApplied.8.064017

171. Kharya P, Alvi A. Using DeepSpeed and Megatron to train Megatron-Turing NLG 530B, the world's largest and most powerful generative language model [online]. (2021). Available at: https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/

172. Sharma T, Ali M, Chakraborty I, Roy K. What, when, where to compute-in-memory for efficient matrix multiplication during machine learning inference. *IEEE Trans Emerg Top Comput* (2025) 13(3):1215–29. doi: 10.1109/TETC.2025.3574508