Merged Logic and Memory Fabrics for Accelerating Machine Learning Workloads

Brian Crafton, Samuel Spetalnick, Yan Fang, and Arijit Raychowdhury
Georgia Institute of Technology

Editor's notes:
Designing hardware accelerators for machine learning (ML) applications is a well-researched problem. This article presents a tutorial regarding new computing architectures, circuits techniques, and multiple promising device technologies for in-memory computing targeting ML workloads.
—Partha Pratim Pande, Washington State University

Today's computing systems and emerging workloads are heavily dependent on the capacity and latency of memory banks, thanks to the increasing performance gap between main memory and logic. Decades of research and the majority of on-chip area in modern integrated circuits (ICs) has been dedicated to creating complex memory hierarchies to negate this growing performance gap. Although this design strategy works well for general-purpose computing, recent trends in data analytics and artificial intelligence have further exacerbated the long-standing memory bottleneck. Rather than fast single threaded performance and unknown data-access patterns, these applications require massively parallel computation and fixed, known data-access patterns. As a result, the clever caching hierarchy that takes advantage of spatial and temporal data reuse is overwhelmed by the vast amount of data required by new applications. Because these elaborate caching schemes have been rendered practically useless owing to the embarrassingly parallel nature of emerging applications, traditional processors have failed to provide either the performance or energy-efficiency that are demanded by these workloads.

Consequently, we have seen a plethora of high-quality software packages, such as TensorFlow [1] and PyTorch [2], and hardware packages, such as Google's tensor processing unit (TPU) [3] and Nvidia's Volta, that vastly outperform previous top of the line commercial general-purpose hardware. Unfortunately, it is inevitable that these improvements will again slow down and the hardware solutions will be limited in performance by the memory bottleneck. To make matters worse, the cost of moving data has become more expensive than operating on it [4], [5]. So not only has memory become the fundamental bottleneck of computing, but both reading and transporting the data throughout the growing size of modern ICs has become more expensive than the operation we seek to perform.

This has given rise to various areas of research to mitigate the memory bottleneck, and tremendous effort is being driven from the device to the architectural levels. At the top of the computing hierarchy,

Digital Object Identifier 10.1109/MDAT.2020.3016587
Date of publication: 14 August 2020; date of current version: 10 March 2021.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

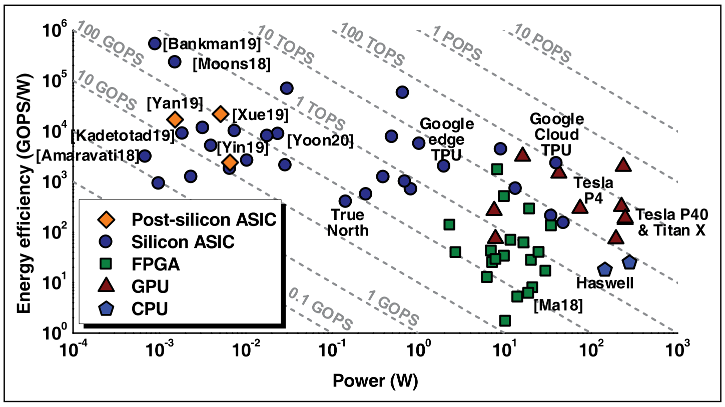

Figure 1. Recent advances in machine learning hardware spans over a large power-performance design space. (Source: Dr. Jong-Hyeok Yoon, Georgia Tech.)

computer architects and compiler designers are attempting to use the same CMOS technology, but use advanced dataflow patterns and mapping strategies so that the data movement is minimized and data reuse is maximized. Several recent demonstrations in machine learning hardware have made tremendous advances in both performance and energy-efficiency, and this is illustrated in Figure 1. Original works [6], [7], created accelerators to match the architectures of emerging deep neural networks (DNNs). Later works [5], [8], [9] showed the primary opportunity for improving the performance of these accelerators is maximizing data reuse (or minimizing total data transport). The initial demonstration of this technique was Eyeriss [5] in 2016, but since then new strategies for both reinforcement learning (RL) [10] and recurrent neural networks (RNNs) [11] have also been demonstrated in silicon. Furthermore designs using mixed signal compute [12], [13] have also been demonstrated using low and variable precision. Although these designs feature data reuse and reduced power, they make application specific design choices which reduce their generality in a wide spectrum of applications. In Figure 2a and b, we show architectural differences between the TPU [3] and Eyeriss [5]. By placing small caches inside of the processing elements (PEs), Eyeriss is able to reuse feature maps and filter weights

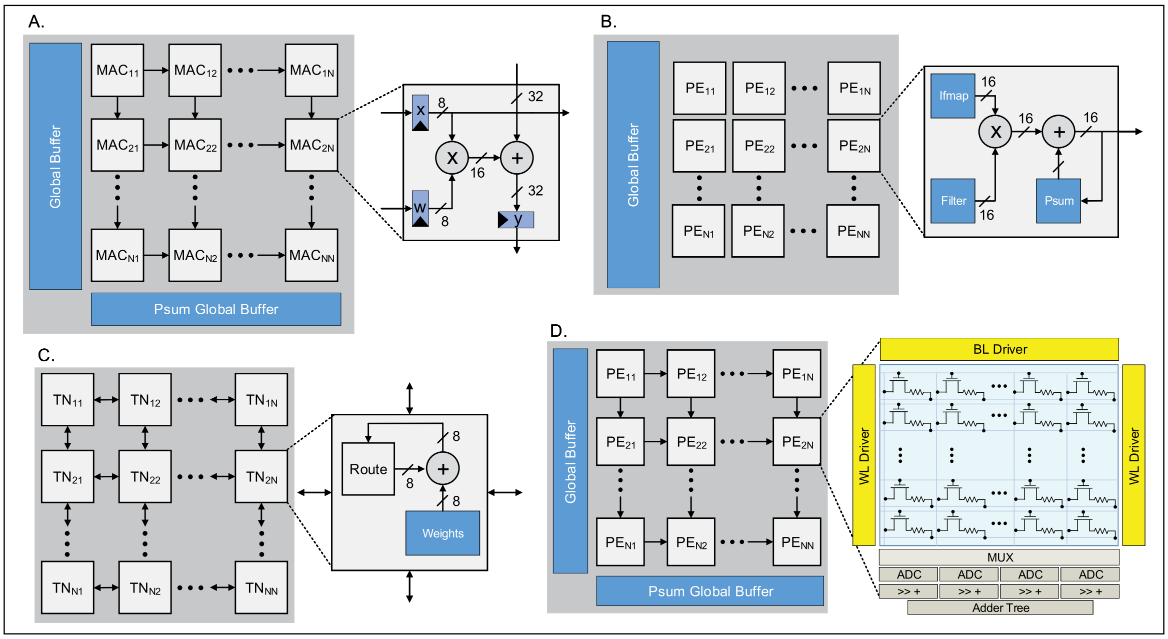

Figure 2. Four architectures used by machine learning accelerators. The memory used to store weights and input data is colored in blue. (a) Systolic array used by the TPU [3]. (b) Systolic array with local cache for filters and feature maps to maximize data reuse used by Chen et al. [5]. (c) SNN architecture with all weights stored local to neuro-synaptic core used by Merolla et al. [19] and Davies et al. [20]. Each core stores weights for many neurons, computing is shared among all neurons mapped to the core. (d) In-memory architecture using RRAM. Computing and memory are combined so there is no memory transfer. Analog-to-digital converters (ADCs) are still shared among inputs and outputs to the array.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

rather than reading and transporting them for each output neuron. More recent works [14]–[17] are attempting to exploit sparsity and compression in both weights and activations of large networks. This is particularly important in emerging models for natural language processing [18] where huge numbers of parameters are used.

In a radically different approach spiking neural network (SNN) chips like [19] and [20] attempt to only read and modify weights using their respective neuron. As a result, large amounts of static random access memory (SRAM) and low precision weights are required to keep all weights on chip without having to fetch them from the main memory. This is demonstrated in Figure 2c, where no global buffer is used and the only transfer of data is directly between cores. Because weights are only accessed by a single core, compute units are not reused across the various layers. This is not ideal from a performance perspective because it means latency will be much higher given that only a fraction of the compute units are used for a given layer. The throughput can potentially be recovered if all the layers are pipe-lined. However, the biggest advantage of this dataflow architecture is its ability to reduce global movement of data. This design strategy is commonly referred to as near memory computing, where the objective is to physically place compute and memory together rather than keep them separated by a high performance memory controller. While this is a useful technique and we can expect significant power savings, it will still face some of the fundamental technological limitations of CMOS.

It is critical to understand the role of embedded nonvolatile memory (eNVM) on future computing platforms. Volatile memory solutions for on-die integration are typically charge-based: SRAM and embedded-DRAM (eDRAM). Both these technologies consume zero standby power—in terms of leakage (SRAM) or refresh (eDRAM) power. With the growing need for larger and larger on-die memory for machine learning applications, the total standby power continues to be a significant limitation in the energy efficiency of SRAM- and eDRAM-based systems. Hence, eNVM is expected to alleviate this challenge. Furthermore, the eNVM solutions that are currently being pursued include noncharge-based physical states—such as ion movement or spin polarization. These physical phenomenon are often scalable to dimensions that are smaller than what SRAM or eDRAM can enable. Consequently, eNVM technologies continue to gain popularity as both a research vehicle as well as a solution at the end of the scaling roadmap.

Fortunately, for more than a decade there has been a steady increasing effort in design, fabrication, and manufacturing of novel memory technologies that are logic process and voltage compatible, while providing high density as well as target read and write performance. These new devices have exciting new properties that have been long absent in traditional charge based memory technologies. The four such technologies that we discuss in this article share the following properties [21]:

- They are all eNVM solutions. They can be completely powered down without loss of data, and hence consume virtually no leakage power.
- All these technologies are process and (somewhat) voltage compatible with CMOS logic processes, although more advances need to be made on both fronts.
- All these technologies store information through change of resistance. This enables us to perform compute in-memory (CIM) on the bitline (BL) with breakthrough improvements in throughput and energy-efficiency.

These properties have the potential to realize the long awaited benefits of in-memory computing. In addition to these properties, new eNVM technologies feature high density and often multilevel cells (MLCs) where more than a single bit can be stored per cell. In comparison, modern 6T SRAM can be as large as $150F^2$ while only offering a single bit of differential storage. In-memory computing uses physical properties of the devices to do computation without a dedicated compute unit. Using Ohm's law, where a voltage applied across a device's conductance results in a current, and Kirchhoff's current law (KCL) we can sum along the columns of our memory crossbar to perform matrix multiplication in O(1). These memory crossbars can then abstracted into cores as shown in Figure 2d, where system level approaches similar to systolic arrays [3] or DNN accelerators [5] can be used.

If successful, in-memory computing promises to solve many of the engineering challenges that the modern memory hierarchy faces with regards to data transport. In recent years, new emerging devices have made huge milestones on their way to commercial viability. High density macros have been fabricated [22], and used in the implementation of highly

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

efficient neural networks [23]–[25]. However, these new emerging memories face many challenges of their own preventing them from widespread commercial use. In this article, we will examine some of these challenges and provide an overview of the recent developments both from technology as well as circuits and systems perspectives.

System needs for memory-centric neural network workloads

The demand for in-memory computing is ever growing as the state-of-the-art designs for computer vision, natural language processing, and RL continue to have larger models with more computation and performance that require months of training time on clusters of custom accelerators. At the same time advances in microrobotics require smaller area footprints and power budgets. These different workloads require the same thing: energy-efficient computing. As these applications begin to outpace the computers we have to perform these tasks, we will need to look at fundamentally different approaches of computing.

At the start of the current surge in deep learning, AlexNet [26] was trained and implemented using a pair of GTX 580 graphics processing units (GPUs) and custom CUDA code. Soon after larger and larger models requiring more hardware became the new state of the art, as VGG [27] and ResNet [28] required multiple GPUs to train in a reasonable amount of time. Today there is an abundance of deep learning frameworks and hardware designed for machine learning readily available. These tools and hardware have propelled deep learning research, making it far more practical to design and test large deep learning models. Despite these efforts, state of the art models in natural language processing, computer vision, and RL require massive amounts of memory and computation that can no longer be run on consumer grade hardware. For example, OpenAI has used RL to beat professional players at Atari, GO, and most recently StarCraft 2 [29]. Jouppi et al. [3] claim that training an agent requires 44 days of training with 32 third-generation TPUs. Training such a model is extremely expensive and impossible for anyone without access to enormous compute power. While inference for this model can be run on a typical GPU, newer models in NLP [30] and computer vision [31] require 4 billion and 1 billion parameters, respectively.

At the same time, the interest in machine learning in mobile and edge platforms is constantly increasing and hardware accelerators and software frameworks are being developed specifically for these platforms. While great progress is being made to run small models on resource and power constrained hardware, CMOS limitations and the von Neumann bottleneck will limit the capability of edge hardware.

Overview of emerging nonvolatile memory

There are many properties to evaluate when considering new emerging devices for in-memory computing. Of course we expect these devices to be nonvolatile with good retention as well as having a resistive state that can be used for in-memory computing. Besides these qualities we also must consider read and write times, read and write energy, read and write voltage, endurance, area, and the number of distinguishable states per cell. We present an exhaustive overview of these metrics for in-memory computing in Table 1. We focus our attention to the four leading candidates for in-memory computing and compare them to the current standard used in commercial products: SRAM, DRAM, and Flash. In their current state, none of the emerging devices display all desired characteristics. In a different review article, Yu and Chen [21] and Yu [32] identify the ideal characteristics for these devices.

We group these properties into four major areas: 1) density; 2) read performance; 3) write performance; and 4) reliability. The importance of the device properties is very application-specific. For example, as we discuss later this in work, the write performance and write endurance of these devices are less important if the devices are used in a network only performing inference. However, in a network that is being trained, high write energy can be the primary source of energy consumption, and it can be a significant system challenge.

- *Density*: The density of these NVM technologies is the product of the number of cells per unit area and the number distinguishable states per cell. The density of eNVM is on the order of $10F^2$, whereas SRAM is over $150F^2$. 3D integration of eNVM on the back end of line (BEOL) process is under investigation; where the density can be pushed even further. However, some of the constraints in BEOL processing, most importantly the low temperature requirements, continue to be challenging.
- *Read performance*: Read performance for CIM constitutes memory access, memory transport,

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

Table 1. Device characteristics of mainstream and emerging memory technologies.

Memory	Ref	Density		Read			Write			Reliability	
		Area	Bit	Voltage	Time	Energy	Voltage	Time	Energy	Retention	Endurance
SRAM	[21]	150 F^2	1	1	1ns	1fJ	1	1ns	1fJ	·	10^{16}
DRAM	[21]	6 F^2	1	1	10ns	10fJ	1	10ns	10fJ	64ms	10^{16}
Flash	[21]	10 F^2	2	10	50ns	10pJ	10	1ms	1nJ	10y	10^5
MRAM	[21]	50 F^2	1	1.5	10ns	1pJ	1.5	10ns	100fJ	10y	10^{15}
	[33]	75 F^2	1	1.2	2.8ns	0.7pJ	1.8	20ns	4.5pJ	·	·
	[34]	$0.358\mu m^2$	1	1.2	30ns	14.6pJ	1.2	30ns	17.4pJ	·	·
	[35]	·	1	1.2	3.3ns	0.3pJ	1.2	3ns	0.6pJ	·	·
PCM	[21]	30 F^2	2	<1	10ns	1pJ	3	50ns	10pJ	10y	10^6
	[36]	·	2	<1	<10ns	0.1pJ	3	100ns	10pJ	10y	10^5
	[37]	50 F^2	2	<1	<10ns	0.5pJ	3.3	300ns	30pJ	10y	10^8
RRAM	[21]	12 F^2	2	<1	10ns	1pJ	3	10ns	10pJ	10y	10^6
	[38]	$100nm^2$	1	<1	10ns	10fJ	1.5	10ns	100fJ	10y	10^7
	[39]	·	1	<0.5	5ns	1pJ	<1	10us	1nJ	·	·
	[40]	·	2.3	0.2	23ns	1.76pJ	3	50ns	10.1pJ	10y	·
FeFET-RAM	[21]	40 F^2	2	3	10ns	1fJ	3	10ns	100fJ	10y	10^6
	[41]	·	1	<4.5	<25ns	·	·	<500ns	·	10y	10^5
	[42]	·	1	0.68	3ns	0.28pJ	0.4	0.55ns	4.82pJ	·	·
	[43]	$2.7\mu m^2$	1	1.64	·	15.5pJ	0.8	0.55ns	15.0pJ	·	·

All parameters taken from demonstrations in original research works.When a parameter is not reported or cannot be computed the table index is left blank.

and multiply-and-accumulate (MAC) operations. It is even more important when only inference is being performed as all power dissipation will come from reads.

- *Write performance*: As we will discuss later, write performance is more important when training crossbar arrays since they will be updated very frequently and most of the times the write energy for eNVM devices is much higher than read energy. It is quite intuitive to understand that memory solutions that are inherently nonvolatile will require higher write energy to change states. Often, the devices require high write voltage as well (2–5 V) and they are harder to integrate on the logic process. However, compared to eFLASH where write voltages are often 20 V and higher, the current generation of eNVM require significantly lower write voltages and demonstrate scaling paths for voltage and process compatibility with logic.
- *Reliability*: Most devices have retention on par with commercial flash processes. However, the endurance of these devices varies greatly. As we will discuss later, endurance is important when implementing training since the devices will be updated frequently.

Resistive random access memory

Resistive random access memory (RRAM) (often called ReRAM) is a filamentary device that switches between a high resistance state (HRS) and low resistance state (LRS) based on the direction of current applied across the two terminals. The HRS and LRS in RRAM are achieved by forming and destroying a filament inside the insulator material of the device. By creating and destroying this filament we can lower and raise the resistance of the device by orders of magnitude. The transition from HRS to LRS is called the *set* process where the device allows more current to flow emulating a digital "1." The transition from LRS to HRS is called the *reset* process where the device is less conductive and results in less current across the terminals. Since a read and write operation both apply voltage on the two terminals, the read voltage must be much lower to not alter the state of the device and perform a destructive read. In the 1T1R (1 transistor and 1 resistor) structure, the read voltage is controlled by using a small voltage on the gate of the transistor.

Although there are different types of RRAM, the most successful is metal-oxide RRAM [44]. It has been used in the implementation of a commercial RRAM macro [45] as well as in a trainable neural network [25]. The alternative, conductive bridge RAM (CBRAM), offers a higher HRS to LRS ratio, but has worse endurance and retention. For this article, we will focus on metal-oxide RRAM. The device structure of metal-oxide RRAM simple, comprising a top metal electrode, a bottom metal electrode, and a transition metal oxide (TMO) layer in-between as shown in Figure 3. RRAM initially starts in a pristine state and most devices must undergo formation prior to being used as intended. During formation, an initial large

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

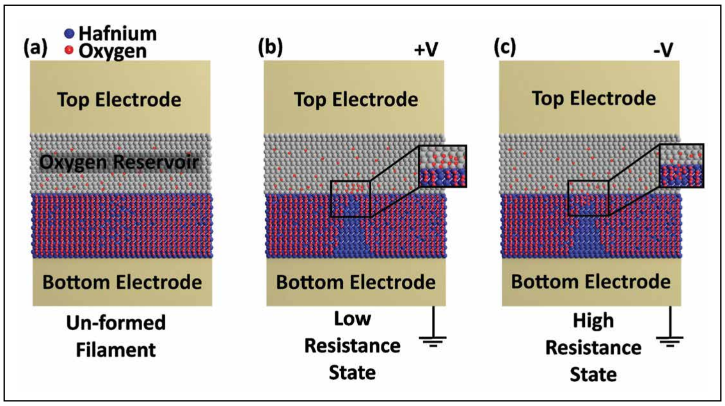

Figure 3. (a) Fabricated device. (b) RRAM after forming and device in the LRS. The conducting filament is formed and will be used to toggle between the LRS and the HRS. (c) RRAM after being reset into the HRS. The tip of the filament is reoxidized which increases the resistance of the cell. (Source: Matthew West, Georgia Tech.)

voltage is required to create an electric field capable of knocking oxygen atoms out of the insulator's lattice and creating vacancies that make up the conductive filament leading to the LRS. The formation process only needs to be done once taking the device from its initial pristine state which has a resistance larger than the HRS of the device post formation.

After formation, it is assumed that the conductive filament is created with sufficient oxygen vacancies in the insulator such that the large formation voltage will not be required again. From the formed LRS state, the device can switch back to an HRS state during the reset process by recombining the oxygen atoms with the vacancies making up the conductive filament. Once the filament is ruptured its resistance will increase, however, the distance of the ruptured filament can cause cycle-to-cycle and device-to-device variation because of the inconsistent location where the filament is ruptured. The way the device is reset depends on the materials used as the insulator. The device can be reset as a unipolar or bipolar switch. A unipolar device is reset based on the magnitude of the pulse applied across the device. A bipolar device is reset based on the polarity of the applied voltage. Most device fabricated at large scale use the HfOx RRAM which is a bipolar switching device. However, there are advantages to using a unipolar device since it only requires a diode as a selector rather than a transistor.

Although used primarily as a binary device, some work has been shown to demonstrate multi-bit or even analog state in RRAM. As many as five states have been demonstrated using HfOx RRAM [46]. If binary encoding is used for multiple bits, then additional CMOS circuitry is needed for add and shift logic. However, achieving multiple states is difficult due to the abrupt switching behavior, low on/off ratio, and device-to-device and cycle-to-cycle variation. Some techniques have been explored to achieve multiple states in the device. Different pulse and programming schemes can be used to better control the exponential behavior of set resistance states [47]. Another technique [48] has demonstrated analog control of the RRAM using the slower reset behavior.

Phase change random access memory

Modern phase-change memory (PCM) devices enjoy relative maturity and have been explored for several decades. Early work by Ovshinsky [49] demonstrated the ability of phase-change materials to store data, and a subsequent discovery by Yamada et al. [50] demonstrated a class of materials the stored state of which could be overwritten many times and switched quickly enough for storage devices based on these materials to be competitive with the then-dominant memory technologies. Phase change materials have since become the critical component in optical storage media, with an additional research interest in these materials for use in new types of electronic storage devices having been revitalized [51].

PCM consists of memory devices that take advantage of the ability of certain materials to repeatably transition between a crystalline phase and an amorphous phase. For a material to be useful in typical PCM applications this transition should be accompanied by a marked change in at least one measurable quantity. Memory systems constructed from these materials typically leverage either a large contrast in reflectivity, as in the case of optical storage, or in resistivity, as in the case of the electrically operated phase-change random access memory (PCRAM), where *contrast* here refers to a change in the measured quantity as observed in the crystalline material versus that observed in the amorphous material [51]–[54]. This section will proceed to discuss resistive PCRAM and will pay particular attention to the $Ge_2Sb_2Te_5$ (GST) chalcogenide material that finds

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

extensive use in PCRAM, e.g., [36], [55], and [56]. A discussion of material and device properties in the context of memory applications will be followed by a brief survey of quantifiable PCM parameters.

PCRAM devices are made up of a layer of glass chalcogenide material, hereafter assumed to be GST, sandwiched between two electrodes, as shown in Figure 4. These electrodes may be used to write or read the cell state. Crystalline GST is more conductive than amorphous GST, and the conversion of the volume of GST within the cell between these phases causes a resistance change that is typically at least one or two decades [53], [57]. State switching is accomplished via joule heating by passing current through the cell, while readout is typically done by placing a small voltage across the device and measuring current [58]. This is shown in Figure 4. The two phase-transition write processes, the crystallization-driven SET toward lower cell resistance and the melt/quench RESET toward higher resistance, will now be discussed in more detail.

Crystallization of amorphous GST is a temperature-dependent process [59], [60]. Crystal growth velocity first increases exponentially with temperature until a relatively hot transition point at which the relationship between crystal growth and temperature slows down, with growth velocity eventually reaching a peak and then decreasing with further increases in temperature. Therefore, Joule heating to an intermediate temperature (≈400 °C) leads to rapid growth in the volume of crystalline material in a given cell and tends to reduce the net resistance between the two contacts, giving rise to a SET operation that consists of a prolonged current pulse with limited magnitude. Heating past the rapid crystallization temperature eventually results in melting of the GST material at ≈600 °C. Thus, a RESET operation involves a large-magnitude current pulse followed by a quick quench, to solidify the material in an amorphous state, which is accomplished by a sharp trailing edge. The relatively high-temperature transition point at which the GST crystal growth rate ceases to increase exponentially helps to resolve the dilemma that arises from the conflicting requirements for a PCM material, i.e., that it must crystallize quickly when desired but must crystallize only very slowly, and in the ideal case not at all, when not intentionally heated.

Without an additional mechanism, the voltage required to produce sufficient Joule heating to achieve rapid crystallization from the high-resistance cell state would be problematically large. However, PCM materials exhibit a threshold-switching behavior defined by a dramatic increase in current flow through the amorphous-state material under a sufficiently strong electric field [49], [61]. This switching is a high-field effect distinct from the previously mentioned phase-mediated memory switching. As threshold switching is induced by field strength the voltage at which switching occurs is nonstationary and depends at least on the thickness of the amorphous layer [62]. Notable is that this effect has a non-negligible temporal component, exhibiting both a delay time, between application of the switching voltage and the large current increase, and a recovery time, between the cessation of voltage application and the corresponding increase in resistivity, both on the order of tens of nanoseconds [63]–[65]. When biased subthreshold, the amorphous resistance still depends on field strength and cells are in an approximately linear IV region only for applied voltages of up to about 300 mV, above which the IV dependence becomes exponential [66].

After cessation of Joule heating an additional transient effect leads to a slower power-law increase in off-state resistance which is accompanied by a similar increase in threshold voltage [63], [67]. This so-called resistance drift may be explained by the time-dependent structural relaxation of the

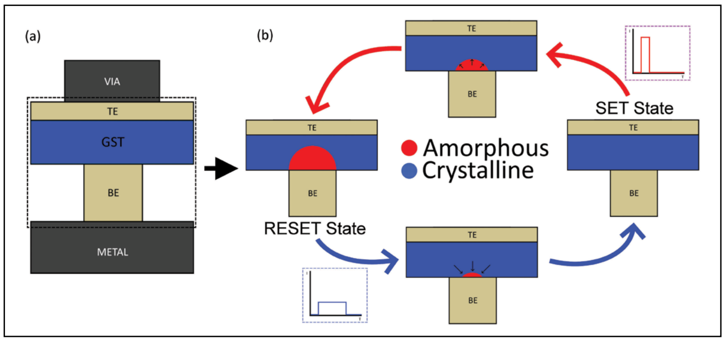

Figure 4. (a) Typical "mushroom cell" PCM layout, showing top electrode (TE), GST, and bottom electrode (BE). Proportions follow TEM images of a recent device demonstrated in [36]. (b) State-transition diagram showing the PCM cell in the RESET state, with the BE plugged by high-resistivity amorphous material, and in the SET state, with the volume of GST material switched to the conductive crystalline phase. Partial-set, bottom-middle, is typical of MLC while partial-reset, top-middle, is less feasible.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

amorphous-state material [68]. A more complete elaboration on the time and temperature dependence of the amorphous phase resistance may be found in [69] or [70]. In a non-CIM application with two-state (binary) PCM cells this drift is less important as it tends to broaden the interstate gap. An associated property of GST-based devices is the presence of a $1/f$ noise current which is nearly two orders of magnitude greater in the amorphous state as compared to the crystalline state [71].

Having laid out the key physical properties of PCM, we proceed to discuss how these properties practically impact PCM devices. Crystallization occurs over time, during which the cell is held at an intermediate temperature. Recall that the melt and quench procedure can be very fast, and in fact the falling edge must be very fast to prevent recrystallization. This means the SET operation traditionally requires hundreds of nanoseconds and is much slower than the RESET operation which can occur in tens of nanoseconds. This long SET operation also means the write time dwarfs the read time. Furthermore, the required Joule heating means that the write operations are energetically costly. This is again in contrast to the read operation, which must occur at just a few hundreds of millivolts to avoid threshold switching and remain in the linear operating region of the cell.

Recompense for the difficulty of writes is that the written states are nominally of high quality with a substantial on/off ratio of at least 100× in measured devices [36]. A consequence of this is that it becomes feasible to program cells to intermediate states between the fully on and fully off conditions, leading to increased storage density through MLCs [58], [72]. Both MLCs and CIM (via current summing across multiple cells) imply a nonbinary range of output states and thus depend on precise control of cell resistance, which is not trivial for two reasons. First, there are many degrees of separation between the input variable during programming (i.e., current or voltage) and the output (which is finally measured as resistance). Second, there are the previously discussed resistance drift and $1/f$ noise in the amorphous material, which become problematic for applications that depend on the discernibility of several intermediate states despite being mostly benign with respect to the binary memory application.

The first challenge may be approached by using feedback-based write algorithms. In these schemes, a SET is accomplished with a train of write pulses, where the properties (such as amplitude, width, and trailing edge length) of each subsequent write pulse are modulated based on a resistance measurement that is made after the previous pulse [58], [72]–[74]. Feedback cannot be taken during the write pulse itself due to threshold switching, and there must be some delay after the write, prior to the low-voltage read, to ensure that normal resistance has resumed. The time for each cycle of a write algorithm is therefore determined by the delay and recovery time of threshold switching on top of the minimum pulse time to heat the material and cause some phase transition.

The second challenge, which may be summarized as the poor resistive accuracy of the amorphous material, does not present application-agnostic circuit-level solutions. As mentioned, the effect of this poor accuracy on purely binary memory is expected to be minimal, and indeed more in-depth noise analysis shows that the $1/f$ noise component has only a slight impact on RESET-state readout relative to other sources of error [75]. In the context of MLC, these problems may be understood as components of a broader issue, namely the decoupling of the measured quantity (cell resistance) from the underlying stored state (relative volumes of crystalline and amorphous material). This has motivated the development of alternative cell-state metrics that, relative to the low-field resistivity metric, more directly audit the amount of amorphous material in a cell by measuring with higher subthreshold fields [76], [77]. Cell readout under these schemes implies a voltage bias that varies with cell state meaning they cannot directly be used in current-accumulative CIM applications.

In light of technological accuracy limitations, applications that are inherently error-tolerant or that rely on CIM only for low-precision components of a computing task are ideal candidates for acceleration via CIM with PCM. Recently presented examples of such robust applications include compressed sensing [78] (in which the linear algebraic operations associated with both compression and recovery of signals may be accelerated with approximate in-memory compute with PCM) and the solving of linear equations using deliberate mixed-precision techniques [79]. Note that in the first example the error due to the inaccuracy of the PCM is accepted, since some error in the reconstructed signal (an image) is tolerable, while in the second example a high-precision computing unit complements the CIM system in an iterative refinement scheme so that the resultant error

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

is only that of the high-precision unit. An alternative approach is to attempt to mirror biological synapses by leveraging the analog (i.e., continuous) nature of the increases and decreases in resistance that can be accomplished in a PCM cell [80]. Continuous crystallization is also exploited in [81]. A more basic approach is to select a lower precision version of the computing task and hide any remaining error behind the built-in robustness of that task, as is the case with binary neural networks (BNNs) as suggested in [36].

Spin torque transfer magnetic random access memory

A more mature technology for resistive memories is the spin transfer torque-based RAM. The Spin torque transfer magnetic random access memory (STT-MRAM) or MRAM bitcell consists of one access transistor and one magnetic tunnel junction (MTJ) where a single bit of information is stored. An MTJ is formed with two ferromagnetic CoFeB-based layers and one insulating layer (MgO) in between [82]. One ferromagnetic layer is called a fixed layer because its magnetic moment is fixed to one direction. The other ferromagnetic layer is called a free layer since the direction of magnetic moment can be changed based on the direction of current flowing across the MTJ.

Figure 5 describes how the direction of magnetic moment in the free layer changes based on the current across the MTJ. Figure 5 shows how the direction of magnetic moment in the free layer changes from (a) antiparallel to parallel and (b) parallel to antiparallel direction compared to the direction of magnetic moment in a fixed layer. Since the fixed layer acts as a spin polarizer, the spin polarized electrons that pass the fixed layer exerts the torque on the magnetic moment in the free layer and causes a flip in the direction of the magnetic moment in the fixed layer as shown in Figure 5a. When the current flows from the fixed layer to the free layer as shown in Figure 5b, the electrons with opposite spin are reflected back from the fixed layer and exerts a torque that changes the direction of the magnetic moment of the free layer to an antiparallel direction with respect to the magnetic moment in the fixed layer. The alignment of the magnetic moment in the fixed and free layers determine the resistance across the MTJ. When the magnetic moments in the two layers are antiparallel to each other, the resistance across MTJ is high.

A low resistance is achieved when both the magnetic moments are parallel to each other. The high/

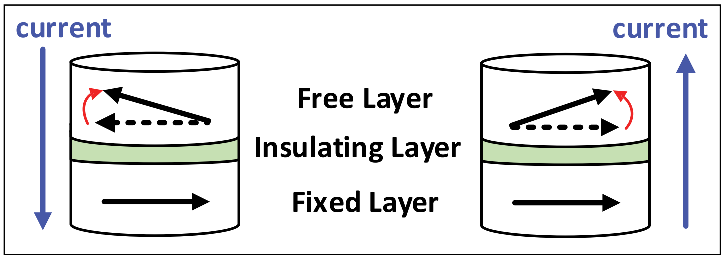

Figure 5. Direction of magnetic moment in free layer changes from (a) antiparallel to parallel (b) parallel to antiparallel to the direction of magnetic moment of fixed layer. The arrow in the free/fixed layer indicates the direction of magnetic moment.

low resistance is mapped to 1/0. The bias conditions applied for the write and read operations are shown in Figure 6. As shown in Figure 6a, the write operation is bidirectional. In the case of writing a 1, the BL and the source line are set to supply voltage (VDD) and ground (GND) and the write current flows from the fixed layer to the free layer of the MTJ. The biasing condition for writing a 0 is the opposite and is shown in Figure 6a. In the case of read operations, the wordline is asserted to VREAD, and the BL and the source line are set to VDD and GND, respectively. This causes a weak current to flow across the MTJ and the resistance state is sensed using either a constant current scheme or a BL discharge scheme [83]. As the technology continues to mature and move from research to manufacturing, larger arrays and systems are being demonstrated.

Ferroelectric field effect transistor-based random access memory

Ferroelectric FET (FeFET) is a nanoelectronic device composed of a traditional MOSFET or FinFET and an additional special layer of ferroelectric (FE) material, which is integrated into the stack of gate

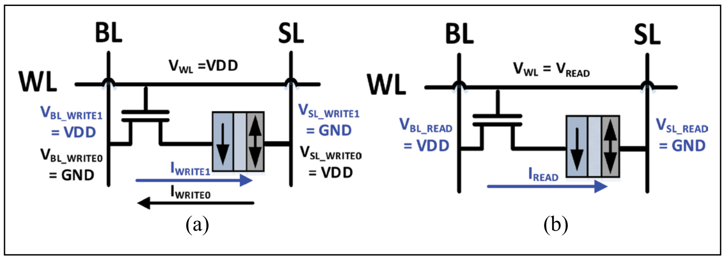

Figure 6. STT-MRAM cell schematic of (a) write and (b) read operation.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

terminal [84]. Figure 7a illustrates the 3D structure of a FeFET built on FinFET. The FE materials are usually lead zirconium titanate (PZT) [85] or hafnium zirconium oxide (HZO) [86]. The latter one is compatible with CMOS process while the former one does not. Note that the spontaneous polarization of the FE layer is reversible under a certain electric field applied in the correct direction. The polarization depends on the current electric field and its history, resulted in a hysteresis loop. Such a feature of FE layer induces a FeFET to switch "on" at a high voltage and "off" at a low applied gate voltage (Figure 7b and c). It is nontrivial to clarify the definition of "FeFET" here since the FeFET is not the only device that stack FE layer on the gate of a FET and such a semiconductor structure can also operate in different operating modes. For example, a negative capacitance FET (NCFET) with the same structure operates in "steer switching mode" and usually aims at eliminating hysteresis for faster switching [87]. Here, we emphasize that an FeFET exhibits the property of hysteresis that can be utilized in different applications. FeFET has been explored in multiple application levels of computing systems, such as components of traditional analog circuits [87], digital circuits [88]–[90], nonvolatile memories [42], [91], and FPGA [92], [93].

Recently, FeFET has been considered as an emerging device used as a memory structure (FeFET RAM) particularly for machine learning hardware and neuromorphic computing. It was applied to the accelerator architectures of both digital and analog neural networks [94], [95]. In these works, FeFETs are utilized to design synaptic crossbar arrays such as RRAM crossbar structure. RRAM-based architectures are troubled with high write energy and sneak paths. The three-terminal structure of FeFET provides better write and read power consumption [94]. Write operations consume power only when FE layer capacitance VG gets charged, which is much lower than the crossbar of RRAMs. For the accelerator of a binary convolutional neural network (CNN), the cell of an FeFET crossbar array consists of two FeFETs and two access transistors. It performs the XNOR operation between the input bit and the weight bit stored in the two FeFETs. Such a design of FeFET-based crossbar exhibit power reduction of both read and write operations when compared to the same designs based on RRAM and CMOS [94]. In addition to acceleration of BNN, FeFET-based nonvolatile analog memories are a promising solution to the future accelerator of DNN with analog weights. An FeFET synaptic weight is capable of achieving multibit operation by leveraging the partial polarization switching dynamics of multidomain FE $Hf_xZr_{1-x}O_2$ thin films [95].

Furthermore, FeFET has been recently explored in SNNs, another neuromorphic computing paradigm that is more biomimetic than DNN. In these scenarios, FeFETs are adopted in the circuits of bioinspired neuron model, instead of synapses. A spiking neuron with excitatory and inhibitory interfaces can be implemented with a relaxation oscillator based on FeFET [96]. The proposed circuits employ the hysteresis of an FeFET and a traditional NMOS transistor to periodically charge and discharge a load capacitor and generate spikes of voltage. Such a two-transistor compact design of silicon neuron is capable of modeling multiple neural dynamics that have been observed in cortical and thalamic neurons when fed with excitatory and inhibitory synaptic inputs [97]. These various neural dynamics are demonstrated to be useful in a FeFET-based SNN that solves optimization problems [98]. Due to the flexible neural dynamics provided by FeFET-based spiking neuron it also has the potential to become a promising circuits design for other applications in neuromorphic engineering, such as neural interfaces and biohybrid neural circuits [99], [100].

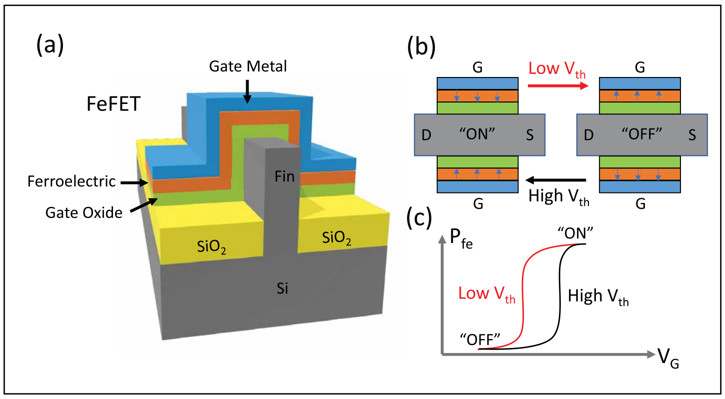

Figure 7. (a) 3D structure of FeFET, (b) "on" and "off" states of FeFET with different threshold voltages, corresponding to the polarity states of FE layer shown in (c), and (c) hysteresis in terms of polarity and gate voltage.

Overview of in-memory computing

Matrix multiplication in eNVM crossbar arrays

Modern deep learning techniques such as CNNs and RNNs contain a workload of almost entirely matrix multiplication ($\vec{y} = W\vec{x}$) [101]. In traditional

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

von Neumann machines, both the feature data $(\vec{x})$ and matrix weights (W) are transported from the main memory to the compute units, where the MAC operations are performed, and after which, the results $(\vec{y})$ are transported and written back into main memory. For this procedure, prior work has shown that the energy cost of reading and transporting data from memory to logic greatly outweighs the cost of the MAC [4], [5], thus motivating in-memory computing.

In-memory computing seeks to perform the MAC operation in a crossbar structure using Ohm's law and the nonvolatile conductance state provided by ReRAM or other emerging memories. Using this technique, each weight of the matrix (W_{ij}) is programmed as a conductance to a cell in the crossbar and each value of the vector $(\vec{x}_i)$ is converted to voltage and applied to the rows of the memory crossbar. By Ohm's law, the current through each cell is proportional to the product of the programmed conductance (W_{ij}) and applied voltage $(\vec{x}_i)$. By KCL, the resulting currents summed along the columns of the crossbar are proportional to the product of the matrix and vector, $(\vec{y})$. Under this procedure, the only data transport required for matrix multiplication is the feature vector $(\vec{x})$ and result $(\vec{y})$. Therefore, in-memory computing is positioned in such a way that it eliminates the majority of data transfer and thus the energy cost of DNNs.

Analog versus digital in-memory computing

The motivation for eNVM has come from multiple directions, and as a result, two different models for the memories have emerged: 1) digital and 2) analog. These memories were originally intended to be used as a digital, low latency, and energy-efficient alternative to existing nonvolatile memory in the traditional memory hierarchy. Later on, HP labs created a metal-oxide RRAM and claimed it had memristive properties [102]. They intended to use this device as an analog memory that could serve as a synapse or weight in neural networks.

There are several tradeoffs between the two models. The main tradeoff is performance versus feasibility. On the one hand, the analog model has numerous performance advantages over the digital model. In the ideal case, the analog model can store any positive, continuous value between its on and off resistance states. Such a device could be used to store high precision values without needing several devices to represent a single weight. Given that traditional neural networks contain both positive and negative weights, two cells are required to represent a single weight since the conductance state of eNVM cannot be signed. The value of the weight is represented as the difference of the conductance of two cells as shown in Figure 8. Naturally, this enables very high density, but also has implications for both power and performance.

However, due to device-to-device variance, cycle-to-cycle variance, and limited on–off resistance ratio using the eNVM as an analog memory has proven difficult. Hence most implementations today focus on digital implementations using write–verify circuitry. Recent work [36] has demonstrated PCRAM devices with a resistance standard deviation of 3.5% and on-to-off ratio of $>10^2$ using this technique. With such a device, it is plausible to enable MLC. An important distinction between MLC and analog memory is that while an MLC has multiple states, it still requires ADC to distinguish the states.

Shafiee et al. [103] encode weights in a offset format so that negative weights can be represented. Given that the devices can perform only multiply and accumulation in the crossbar, there is no way to naturally represent a negative value. Therefore, a bias is applied to all weights before being written to the crossbar, and then subtracted after the rows are read. Since many rows are turned on and accumulated along the same column, the bias is subtracted from the final result for each row that was turned on. For example: in a signed 8-bit 2's complement format, -128 would be written to the crossbar as 0. After being read out during compute-in memory, the bias of 128 would be subtracted.

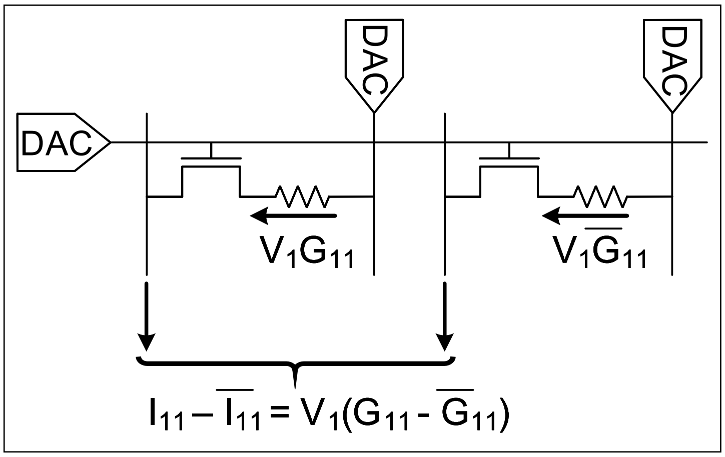
DAC
DAC
DAC
V_1G_{11}
$V_1\overline{G}_{11}$
$I_{11} - \overline{I}_{11} = V_1(G_{11} - \overline{G}_{11})$

Figure 8. 1T1R circuit with a differential pair for signed weights.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

Peripherals for in-memory computing

CIM architectures promise to achieve certain advantages over traditional architectures in efficiency and throughput by replacing digital addition with an analog-domain alternative. This implies that the binary bitcell readouts typical of traditional memory macros become multiple-bit readouts in CIM arrays, and the sense-amps typical of traditional memory accordingly become ADCs in CIM architectures. Furthermore, certain technology candidates for CIM require specific bias points for read and/or unusually large power for write. Meeting these added requirements without sacrificing the performance benefits promised by CIM will require novel peripheral circuitry. This section will provide an introduction to the readout peripheral design space for CIM by detailing a representative set of requirements and proposing how they may be satisfied. We focus on readout circuitry up to the ADC, omitting a detailed discussion of both write circuitry and ADC composition since the former is intimately dependent on choice of memory technology and the latter represents a massive variety of design choices.

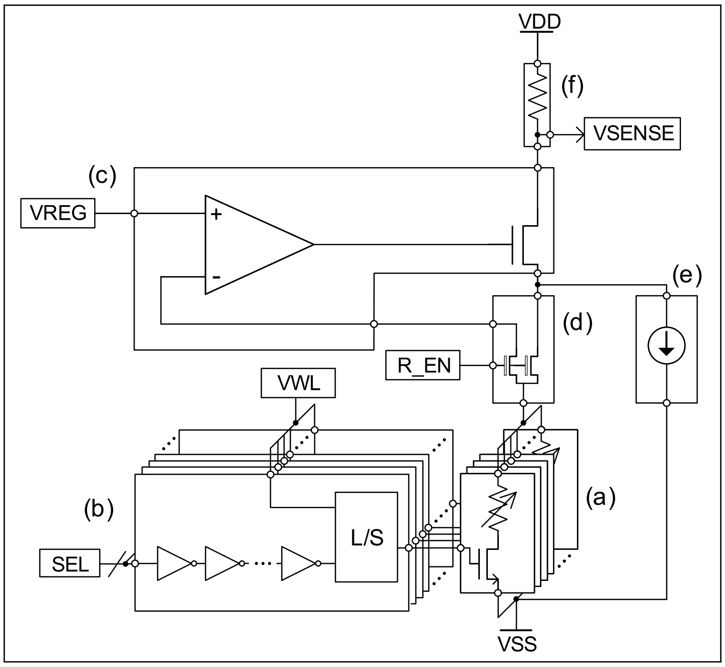

Figure 9. Readout peripherals for a resistive CIM array shown as a block diagram of individual functional units. Examples of the functional makeup of each unit are shown inside each block, although the actual implementations will vary based on system requirements. (a) Column (that is, a group sharing a BL) of resistive memory cells with access devices. (b) Wordline driver with level-shifting. (c) Biasing/BL-driving circuit formatted as a voltage regulator. (d) High-V_t pass transistors that may be necessary to protect the BL driver from write pulses. (e) Bias current used to improve the stability of the voltage regulator in (c) and linearize the current sense device used to implement. (f) Current-sensing effective resistance. Note that some designs will use the gate of the regulator's pass device shown in (c) as the measurement point for a voltage representing memory cell current.

The functional components involved in readout for CIM are shown in Figure 9, a new figure which expands on and has been inspired by the work of Close et al. [104], Kwon et al. [105], and Athmanathan et al. [106]. We propose that the readout process may be thought of as consisting of three stages.

- A *selection* stage during which the cells are activated via a change in wordline state.
- A *biasing* stage during which the selected cells are forced into the correct operating region, so that they produce some readable analog value.
- A *conversion* stage during which the analog value is captured into a digital reading.

The selection stage is accomplished via a buffer chain subject to two functional requirements: it must be sized to drive the wordline capacitance within a required time-frame, and may additionally be required to provide voltage conversion or protection between the core logic synchronizing the computation and the gate voltage of the cell selector devices. The degrees of freedom for wordline driver implementation include the choice of voltage level-shifting scheme, [107]–[112], and the choice of stage-over-stage sizing in the buffer chain [113]. The tradeoff space is speed, typically accomplished with a many-stage buffer and converter design (closer to *e*-scaling), against power and area. Broadly, these design constraints for CIM wordline drivers match those of the well-explored wordline driver design spaces for traditional memory technologies, including DRAM [108], SRAM [107], [109]–[111], and Flash [112]. The wordline driver is modeled by component (b) in Figure 9.

As shown in Table 1, typical implementations of Flash, STT-MRAM, PCM, RRAM, and FeFET-RAM require high voltage and potentially significant current during write. This encourages a low selector device channel resistance which may be accomplished by driving the gates of these selector devices far above typical core logic voltages. For the buffer to survive these

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

operating conditions at minimum the output devices must be able to handle these higher voltages. Typically, this will mean the minimum channel length of these transistors must be increased, which will in turn significantly increase the gate area, and capacitance, of transistors sized to rapidly charge the wordline [112]. Alternatives include cascoded output devices that each only see a portion of the high voltage or a series of moderate voltage step-ups. Any of these will result in increased area and energy usage. Some other technological requirements, e.g., for an especially low gate voltage during read to avoid read-disturb, do not risk buffer transistor breakdown and are therefore significantly easier to accommodate.

Having selected cells, the next task is to bias the cells into an appropriate operating region so that their stored state, typically resistance, may be measured. This is shown in Figure 9c. While other biasing schemes are possible, the most straightforward and typical technique is to apply a specific voltage across the selected cells, converting cell resistance to current [105]. If the state of each cell is expressed as a conductance, the resultant current due to this voltage will be proportional to the sum of the cell states. Constant-voltage biasing therefore directly implements current summing. The circuitry involved in applying a constant-voltage bias across memory cells is similar to that involved in typical voltage regulator designs, with a few additional requirements: the biasing circuit must be able to tolerate high voltage at the BL (if required for write), tolerate widely variable load resistance, and sense the output current.

The first of these additional requirements parallels the voltage-tolerance requirement of the wordline driver. However, the BL driver does not swing across the full output voltage range but instead holds voltage nearly fixed while scaling current delivery to match the load. Additionally, the parasitic BL capacitance, which typically looks into the memory cell (modeled as a variable resistor) may be lower than the wordline capacitance, which typically looks into the gate of a fairly high-current transistor. These factors combine to mean that adding series channel resistance via a protection pass transistor can tolerate high voltage on the BL and can be turned off during write, can be less deleterious to biasing speed and efficiency. This pass device may be inserted between the output of the voltage regulator and the BL. Additionally, if significant cell current may be expected to pass from the regulator to the BL through the pass device, a separate low-current "voltage-sense" pass device may be used for the voltage feedback component of BL voltage regulation. This structure is shown in Figure 9d.

The widely variable load resistance results from a design goal of resistive memory. Resistive memory cells should exhibit a high on/off ratio, meaning that their stored state should have a high dynamic range to maximally separate each nominal state and reduce error. This has the potential to create two issues. Practically, the voltage regulator that biases the cells will likely incorporate negative feedback to establish a precise BL voltage, and may therefore require compensation. This compensation may occur at the output for high performance, and the location of this output pole in frequency will be sensitive to the small-signal impedance at the output of the regulator. In order-of-magnitude terms, as an example, if ten binary memory cells each swing 100× from off-state to on-state, the load impedance seen at the output of the regulator will swing 1000×, i.e., 60 dB. Compensating for the highest-impedance extreme will therefore be likely to hinder performance at the lowest-impedance extreme, which is exactly where a quick response is required as the load will be demanding the most current when most cells are on-state. Conversely, under-compensating to achieve a quick response for the largest load-dump may cause stability issues when few cells are on. The other issue is simply that reliably sensing current across several decades can be challenging.

Both of these issues may be addressed by introducing a current-bias circuit, component (e) in Figure 9, to establish a minimum output current above what would be set by the memory device alone [105]. This bias current can be set to place the expected current range that will be read across a linear region of the current-sensing device, if needed. The bias circuit may be expected to decrease output impedance first of all simply by increasing the minimum current through the output device of the voltage regulator. Additionally, the bias circuit may itself be designed to have a low small-signal input impedance to directly limit output impedance of the regulator.

The remaining challenge to discuss in the context of the bias circuit is current sensing, component (f) of Figure 9. The canonical technique for sensing current is to add a low-value effective resistance in series with the load then amplify the voltage drop across this effective resistance. Typical pure-CMOS

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

implementations have accomplished current sensing via the control-side of a current mirror or cascoded current mirror, that is, with a diode-connected transistor [104]–[106]. In particular, if this current-sense transistor is cascoded (or source-degenerated), the voltage headroom required by the sense device added to that required by the output device of the voltage regulator can be significant. In power-sensitive designs, this may preclude more efficient schemes that would first reduce the core voltage using a capacitive or inductive converter before finally setting the BL bias with a linear regulator. Alternatively, adding several series devices between the regulator's output transistor and the rail may reduce the effective transconductance of this transistor, requiring its size to be increased to recover gain and improve regulation. These factors may motivate a simpler scheme where the gate voltage of this output transistor is itself used as the current-sense metric.

The final step, analog to digital conversion, is application-specific. The maximum useful resolution is established by the minimum of the precision of the memory devices (inversely proportional to their variance) and the expected number of distinct output states to be measured. It may be that the current handling of core devices in a technology limits the maximum number of on-state devices (at a given minimum bias voltage) to a small number so that only this small number of distinct states needs to be measurable, or alternatively it may be that the variance of the current conducted by the memory devices limits the number of devices that may be measured before variance may be expected to cause unacceptably frequent errors.

Circuit-level ADC limitations are due to power and area restrictions. Area usage is a critical factor in CIM because memory arrays can be very dense while precise peripherals can be bulky. In a typical design, peripherals will be pitch-matched along both axes. While wordline drive circuitry can be very simple, the BL components will quickly exceed the pitch of a single column of memory cells. This limits the ADC and BL driver complexity that is suitable for a scalable design. While we will not fully explore the available options for ADCs, consider that the special requirements for ADCs for CIM are that they have a high throughput at a relatively low bit depth (3- to 6-bit designs are typical) and that their complexity and efficiency is managed so that throughput may be increased by tiling many of them adjacent to the array.

Low precision peripheral designs

As we will discuss in the "CIM-based accelerator architecture" section, it is possible to reduce the complexity of peripheral circuits by reducing the bit precision used in both the input features and weights of the network. While this does come at a cost in application performance, several works have shown that this is minimal for less complex applications. As a result, these works can achieve significantly higher throughput and lower energy per operation [13]. However, it should be noted that for larger and more complex problems, higher precision is often required to achieve respectable results.

One such approach is to use a BNN where only 1-bit words and activations are needed [114], [115]. Originally used in FPGAs to take advantage of their configurability and offset the area and power overhead, BNNs have also been shown to work with RRAM in both simulation [116] and experiment [23]. When using a binary activation function, a 1-bit word-line driver and sense amplifier can be used rather than an ADC [117]. This design forgoes using multiple states in the device and avoids the additional circuit overhead. To activate many rows at a time, these networks can change the threshold of their sense amplifier to implement a binary activation function which quantizes all activations to 0 and 1. Although this implementation is appealing from a circuit perspective it suffers greatly when scaling the data sets to more complex problems [115]. Rastegari et al. [115] showed 9% performance degradation on ImageNet when going from full precision weights to binary weights. In addition to accuracy, BNNs struggle with training, as they are trained using full precision weights only to be quantized during inference.

SNNs have also been used to bypass the peripheral circuit overhead of ADCs [24]. Since SNNs only output spikes rather than conventional multibit digital values, an analog circuit can be used to integrate the current through PCM memory and fire a spike when it has passed a threshold. Though this technique is similar to BNNs, it makes use of the time domain by integrating over time and slowly reducing the state value of the neuron over time similar to the behavior of leaky-integrate and fire neuron model. Like the BNN, however, SNNs have issues with both

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

accuracy and training preventing them from seeing large-scale use.

In Figure 10, we show both ends of the complexity spectrum ranging from DACs and ADCs to word-line drivers and sense amplifiers. In the past, many optimistic designs used multilevel inputs, high precision designs, and MLCs. However, such designs have proved to be difficult given cell-to-cell variance and select transistor source degradation. This gave rise to simpler designs using just word line drivers and sense amplifiers with single state cells, but as we discussed these designs have had difficulty in scaling to larger problems. As a result, most recent CIM designs utilize word-line drivers and ADCs [103]. This configuration is popular because it allows multiple rows to be read at the same time, and avoids the source degradation problem that occurs when using multilevel inputs.

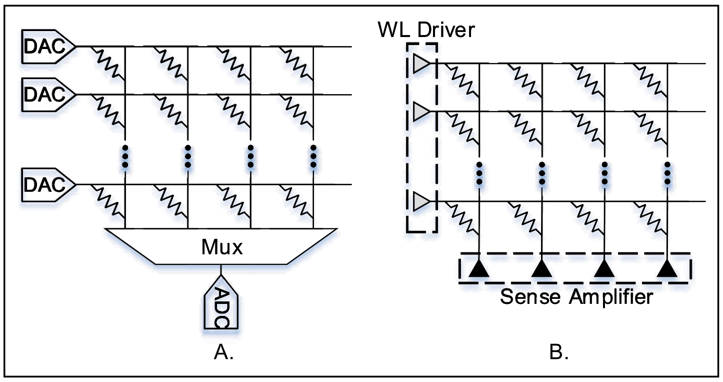

Figure 10. Two different peripheral circuit configurations for neural networks for compute in memory. (a) DACs and ADCs are used for multibit activations and reading higher precision. (b) 1-bit wordline drivers and sense amplifiers can be used to implement BNN and minimize area and power.

Zero skipping

There are two common techniques for performing CIM. The first technique, which we refer to as *baseline*, is simply to read as many rows as the ADC precision allows (e.g., for a 3-bit ADC, we read eight rows simultaneously). The next technique is commonly called zero skipping [118]–[120], where only rows with "1"s are read. Zero-skipping performs faster than the baseline technique because for most cases it will process more total rows per cycle. In Figure 11, we provide an example case for zero-skipping where eight total rows are read using a 2-bit ADC. Baseline (11A) requires two cycles since it targets four consecutive rows at a time. Zero-skipping (11B) is able to finish all eight rows in a single cycle because we only consider the "1"s in the input vector.

There are few reasons not to perform zero skipping, unless there is limited input data bandwidth or the eNVM has high variance and accumulated too many errors. A recent work [120] has exploited this technique along with compression to achieve upward of 10× performance improvement. They illustrate that in most DNNs used today, activation sparsity is well above 50%. In fact, the larger the size of the neural network model the higher activation sparsity is observed.

Impact of variance in in-memory computing

At the device level, the fundamental performance bottleneck is a function of the device-to-device variance and the on-to-off ratio of each cell. These two properties define the number of distinguishable states that can be accurately read from a column of the crossbar. If more states are read than can be accurately distinguished, then errors in the operation will occur following the distribution of the device-to-device variance. Since these errors compromise the

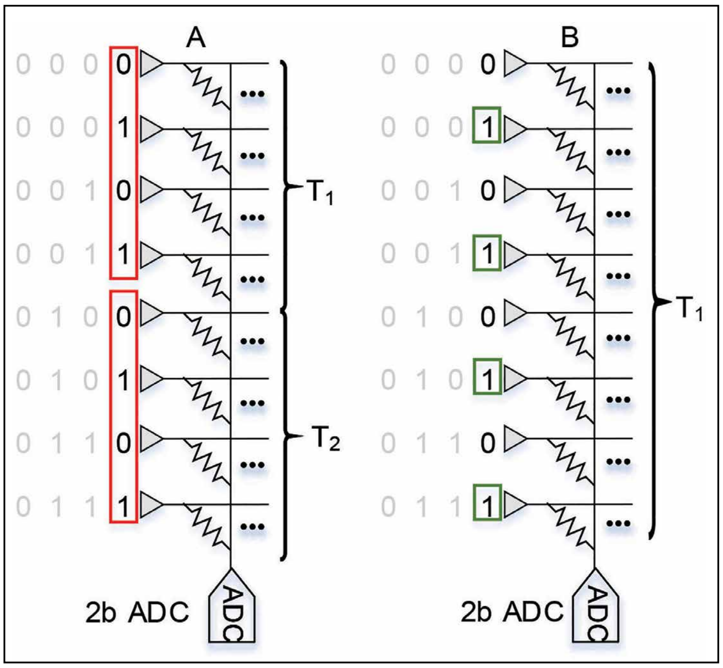

Figure 11. Simplified breakdown of ADC reads in baseline and zero-skipping with 2-bit ADC precision. (a) Baseline targets four consecutive rows at a time since the 2-bit ADCs are capable of distinguishing four states. (b) Zero skipping targets the next four rows where the word-line is enabled. This way we can read more rows and not overflow our ADC.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

accuracy of the operation, the performance can no longer be compared to that of a bit-accurate CMOS implementation. Interestingly, in many data-intensive applications, particularly in neural networks, limited amounts of variance can be tolerated.

Modern eNVM technologies, such as PCRAM and RRAM, suffer considerably from device-to-device variance. State-of-the-art devices [36] have been demonstrated with a resistance standard deviation of 3.5%, on-to-off ratio of $>10^2$, logic process and voltage compatibility, and high density. That said, analyses focused on solely device variability have revealed that the intrinsic and cell-to-cell variability can result in standard deviation that range from 5% to 50% depending on the write effort and the final stored resistance state [121], [122]. While lower nominal variability has been achieved in limited experimental research, real-world factors such as device drift and degeneration along with limited write-energy budget mean that well-controlled variance is rarely guaranteed and often practically impossible. The latter implies a precision/power tradeoff design-space that encourages algorithmic solutions to the variance issue.

In Figure 12, we demonstrate how, given enough variance, a CIM operation will result in an error. We show three cases: 5%, 10%, and 20% variance in the resistance of the memory state. This figure depicts the resultant variance expressed as the cumulative distribution of the computing error, when seven on-state cells are being read (the maximum allowed by a 3-bit ADC) and when 15 on-state cells are read (the maximum for a 4-bit ADC). Recent work [123] has shown several techniques on how the impact of device level variance can be reduced at the system level by controlling the operating speed based on the variance of the devices and importance of the particular operation. In 8-bit matrix multiplication, an error when computing the product of the most significant bits is 2^{14} times more costly than an error for the least significant bits. Therefore, when operating on the most significant bit the number of rows being read should be reduced to minimize error at the system level.

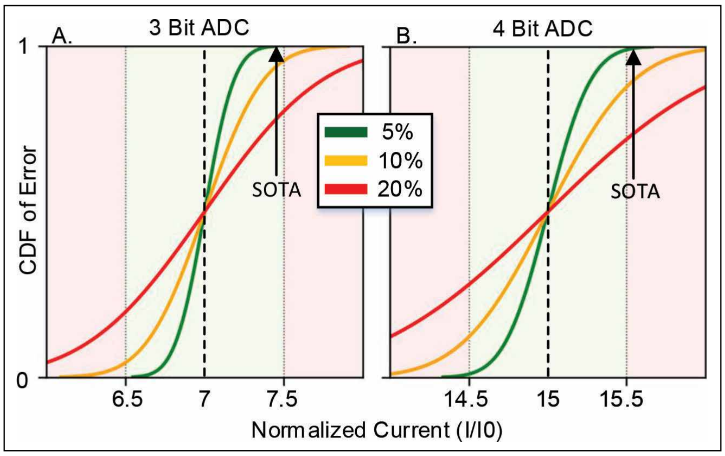

Figure 12. CDF of error in calculating MAC as a function of normalized cell current under parametric variation.

Implementation of neural networks

Vector operations in DNN accelerators

So far, we have discussed the various memory technologies and circuit design techniques to implement CIM engines. Despite matrix multiplication being the bulk of the workload, there are other essential operations to implement in a DNN accelerator. Fortunately, these operations represent a small portion of the workload since they operate directly on input and output vectors rather than performing matrix transformations on the data. There are several examples of this in CNNs, LSTMs [124], and transformer networks [125]. In all cases, these element-wise vector operations are performed post-matrix multiplication and thus have at least twice the bit-precision of the matrix operations.

In a CNN, the most common vector-wise operations are element-wise addition and comparison for bias and ReLU [126]. ReLU is particularly popular given that it typically yields the highest accuracy and also has the simplest operation. Other popular activation functions like sigmoid and tanh require exponential functions and division which require more complex CMOS logic. In low precision logic these can be implemented as lookup tables, however, in higher precision accelerators expensive ALUs are required. Many CNN accelerators [11] create special cores for processing these nonlinear activation functions and applying biases. A similar approach can be used for CIM accelerators, where eNVM arrays process the matrix operations and these vector operations are performed in special logic units.

For training deep CNNs, batch normalization [127] has become an integral component of all very DNNs. Prior to batch normalization and the use of residual [28] and dense [128] connections, very DNNs suffered from the vanishing and exploding gradient problems. Batch normalization normalizes the outputs of

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

intermediate layers and applies a bias and scaling parameter to each output. This adds additional complexity to the necessary ALU operations required because performing multiplication and division are more challenging than addition and comparison. However, this can be avoided in all inference-only accelerators since the normalization, bias, and scaling parameters can be folded together [129].

Another popular neural network architecture for extracting patterns in sequences and time is the RNN. The most popular form of the RNN is a long short-term memory (LSTM). eNVM is especially interesting for implementing LSTMs since they are constructed from large fully connected layers and thus have far more parameters and greater dependence on memory bandwidth. Recent work [130] has demonstrated an RRAM-based RNN accelerator. They create a special function unit (SFU) for the many vector operations of RNNs to complement the PE used for matrix multiplication. Although CMOS logic is used to implement the SFUs, the number of vector operations is far fewer than matrix operations, and therefore does not negate the performance benefit from using RRAM.

Weight and activation quantization

Neural networks are most commonly trained in high precision 32-bit floating point operations. Existing infrastructure in BLAS [131] and GPUs enabled sufficient performance. However, new research trends seek to use lower precision weights and activations to achieve the same results. Naturally, this has big implications in terms of memory capacity, data transport, and computation as the complexity of all three scales with the precision of the operands. Given the performance advantages, popular tools like TensorFlow and PyTorch have created packages dedicated to quantized arithmetic.

Originally, a “binarized” neural network [132] was shown to yield good results for small data sets. This network was trained with full precision weights, but during inference these weights could be stored as just 0 and 1. Later work has shown that this technique did not scale well to more challenging data sets [129], [133], [134], but lower precision weights and activations (8-bit) could still be used on large data sets like ImageNet [135]. There are different types of quantization, namely post training quantization and training aware quantization [129]. Post training quantization uses statistics from the weights and activations distributions to quantize a pretrained model. Training aware quantization trains a quantized model by quantizing full precision weights during the forward pass of backpropagation, and proceeding to train the model as if it were not quantized.

Most works in traditional CMOS accelerators [3] focus on 8-bit integer weights and activations for inference since it yields a nice balance between performance and accuracy. In [129] and [134], a small accuracy degradation can be observed when transitioning from full precision to 8-bit arithmetic. As the precision is lowered beyond 8 bits, a much larger accuracy degradation occurs. For training higher precision is often used [136], however, recent work has shown that it is possible to use 8-bit precision to train networks as well [136]–[138].

CIM array level simulators

eNVM have many new properties that make it difficult to determine their utility in creating machine learning accelerators. Of course it is better to have lower read energy and latency, but understanding how important these properties are at the system level is challenging. This problem is particularly important in device engineering so research effort is allocated to the most important device properties. For example, in the design of a system with low ADC precision (2-bit) device variance and on-to-off ratio have a much lower impact on accuracy and performance than a system with high precision ADCs. It is these types of scenarios that simulators seek to evaluate to both guide device level research and provide accurate power, performance, and area estimations for CIM system level designs.

The design and implementation of these simulators is a research challenge in itself. Typical circuit simulations like SPICE are notoriously slow for even small designs. At the same time, modeling all these properties while evaluating the memories in a higher level system is not only computationally demanding, but rarely explored until recently. Large neural networks are already extremely computationally demanding although they only perform matrix multiplications with optimized BLAS libraries on dedicated hardware. Simulating the same networks with circuit and device level components is even more computationally challenging. Several works [139]–[142] have made promising initial progress on this problem.

An early work, NVSim [139], was not developed specifically for DNNs. Since they avoid the computational complexity of a DNN accelerator, they are able to focus on lower level details that allow for a more

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

accurate comparison with flash for server workloads. More recent works [140]–[142] have extended the idea, but for evaluating DNNs. These simulators use object-oriented models of circuit components and devices to accurately model power, variance, delay, and other properties that must be modeled to give circuit level approximations.

Recent works, such as NeuroSim [140], explore the simulation tradeoff space choosing the important characteristics to model. Given the notoriously slow simulation time of accurate SPICE models, tradeoffs have to be made if a realistic DNN will be simulated. However, accurate estimations for area and power can be achieved by abstracting temporal variation and considering fixed energy and delay parameters for all array level components and global interconnects. RxNN [142] gives special attention to nonidealities of emerging devices and peripheral circuits as these will be fundamental performance bottlenecks in any realistic design. In both simulation tools, all devices from this article can be evaluated with both optimistic and pessimistic parameters. Power and performance breakdowns can be gathered across circuit level components to identify the largest sources of power to motivate and guide future research effort.

Network on chip for CIM accelerators

Although CIM seeks to minimize data transport, it is nonetheless an important aspect of implementing a CIM-based DNN accelerator. Fortunately, CIM-based architectures are similar to CMOS- and SRAM-based designs and thus make use existing research progress in network on chip (NoC), routers, and interconnects. Despite these similarities, there are some differences that require special attention. Given that CIM accelerators offer the advantage of high density and low power, low overhead NoC designs, such as those mentioned in [143], are of special interest. Furthermore, since CIM accelerators typically run at lower clock frequency and struggle with computational efficiency [103], special routing policies [144] will likely be required if eNVM CIM systems are to be used in real-time embedded systems.

Recent work [145] has proposed a new data flow based on prior work in CMOS-based CNN accelerators [5], [9]. This work implements a new weight mapping strategy for minimizing data movement in CIM DNN accelerators. As we discuss further in the "CIM-based accelerator architecture" section, eNVM PEs are not typically reprogrammed and hence each compute unit is not capable of performing each operation in a DNN. Therefore, the mapping of weights to compute units is a new challenge that was avoided in CMOS- and SRAM-based designs. Using this data flow and mapping strategy, the authors demonstrate improvements in power, performance, and memory efficiency.

Since all of the eNVM candidates are both process and voltage compatible with CMOS, we expect future research in interconnects to be coupled with CIM research. Emerging technology such as optical interconnects [146] or wireless NoCs [147] can be directly applied to CIM to further reduce the cost of data transport. Although these technologies need further research and development before mainstream adoption, they promise to reduce power and improve throughput over traditional interconnects, particularly for global routing layers. The combination of CIM and new interconnect technologies has the potential to drastically reduce power in DNN accelerators by both reducing the cost of data transport and eliminating a large fraction of the total data movement required.

CIM-based accelerator architecture

In previous sections, we describe the different components that go into creating a standalone CIM PE. By encapsulating the array, ADCs, and shift and add logic, a matrix multiplication engine can be created. Using these arrays as building blocks, prior work has implemented CNNs where a group of arrays implement a larger matrix multiplication. In Figure 13, we illustrate this idea, showing how a group of arrays is tiled together to form a PE and then used in a larger systolic array that can perform matrix operations. In most designs so far [103], [145], [148], the most common crossbar choice is 128×128 where 8 bit weights and activations are used. Therefore, each array can perform a 128×16 dot product on its own requiring only the 128-byte input vector. These arrays can easily be pieced together to form larger matrices. For example, a 1024×1024 matrix can be formed by a 8×64 grid of 128×128 arrays.

This has important implications as modern DNNs such as CNNs and RNNs contain a workload of almost entirely matrix multiplication [101]. Despite performing more complex operations, the core operations of CNNs and RNNs are converted into matrix multiplication. In Figure 14, we further depict how these arrays can be pieced together to form a larger matrix that forms the kernel of a CNN. In this example, both input feature maps and filters are vectorized

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

with the filters forming the columns of a matrix. The vectorized feature maps are input to the crossbar to perform matrix multiplication, where the results are output feature maps for this layer in a CNN.

Given the high density of these PEs, hundreds or thousands of them can be tiled in the same area used by modern ICs. Although similar in concept, CIM-based DNN accelerators have numerous differences from traditional CMOS-based designs that introduce challenges in maximizing performance. First off, a CIM-based PE has fixed weights that cannot be reprogrammed due to the high energy cost of writing eNVM. Traditional CMOS-based PEs are generalized compute units that can operate on any input data, since they do not contain fixed weights. Thus, the fundamental issue in CIM-based accelerators is array utilization [149]. Several works have addressed this issue introducing ideas such as weight duplication, weight partitioning, and layer pipe-lining.

Layer pipelining

In traditional CMOS accelerators, PEs can be used for all layers of the network. CIM arrays are not as flexible because the weights should not be reprogrammed between layers because of high write energy. Thus, each array in the CIM accelerator should be used for a single layer. It is possible to split layers across these PEs for larger arrays; however, for typical 128 × 128 arrays it is impossible to split many layers across a single array effectively [145].

To combat this issue, pipe-lining the layers of the DNN has been proposed [103], [150]. In these works images are pipelined through the network to keep all arrays fed. Although this compromises single example latency, the total throughput of the array is the same as if all arrays were used for a single layer. However, pipe-lining faces performance issues when some layers perform faster than others. In this case, the maximum throughput of the network is constrained to the slowest layer, hence it is important that arrays are allocated uniformly so that performance is optimized.

Weight duplication

Weight duplication [103], [150] is used to maximize throughput in large-scale CIM accelerators where the amount of on-chip memory exceeds the number of weights in the model. In [145], 24,960 arrays are used for a total on-chip memory capacity of nearly 104 MB (2b cells) for weights alone, while only using an area of 250 mm^2. In comparison, the TPU [3] which occupies over 300 mm^2 only has 24 MB of on chip SRAM that is used for both weights and activations. Using this enormous on-chip memory capacity, they not only fit ResNet [28] but duplicate shallow layers up to 32× so that all arrays are fully utilized.

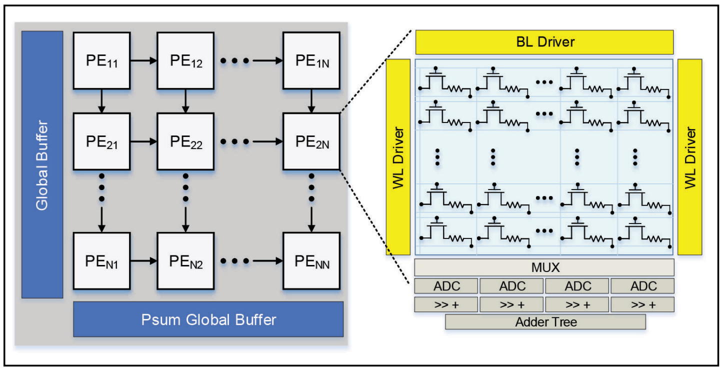

Figure 13. Typical in-memory computing array architecture featuring dual word line drivers, ADCs, shift and add units, and an adder tree.

When weights are duplicated, the input data are divided equally among each group duplicate array so they can process in parallel. In a convolutional layer, the input patches can be sent to each duplicate in an alternating fashion as the accelerator strides across the image. We further illustrate this idea in Figure 15. In this example, we show two duplicates each receiving alternating input patches. Each duplicate performs the convolution and sends the resulting output feature map to its designated location in main memory.

Weight mapping and partitioning

Traditional digital accelerators [14] utilize arrays of PEs to perform MAC operations and accumulate

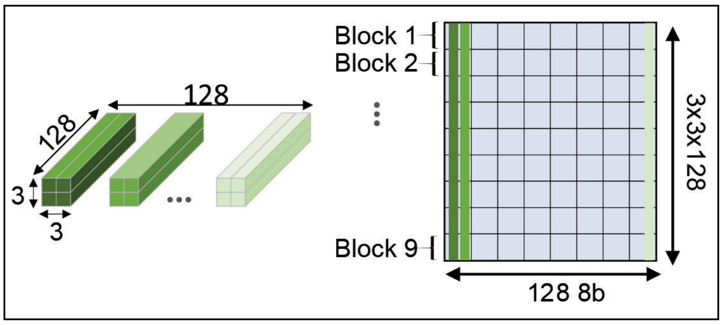

Figure 14. Typical 3 × 3 × 128 × 128 filter used in layer 10 from ResNet18 converted into a matrix and mapped to 128 × 128 eNVM arrays. This filter requires 72 128 × 128 arrays to store in a 9 × 8 grid.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

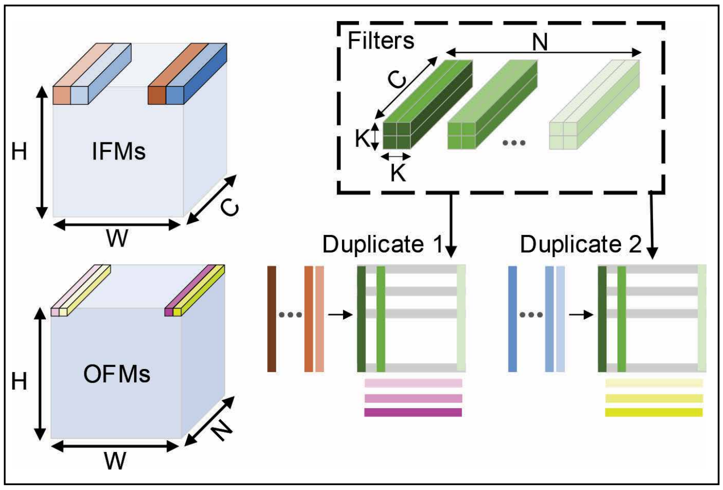

Figure 15. Convolutional layer mapped to a CIM array. Both input feature maps and filters are vectorized with the filters forming the columns of a matrix. The vectorized feature maps are input to the crossbar to perform matrix multiplication, where the results are output feature maps for this layer in a CNN.

partial sums. For each operation, input and weight data are read in through cache or main memory. Due to the low density SRAM caches that are used, data reuse is key to minimize the number of main memory accesses performed. As a result, many data flow architectures have been proposed that maximize both spatial and temporal reuse and consequently, minimize data transport. Despite these efforts, data transport still consumes a large fraction of the total energy consumption.

In-memory computing offers an elegant solution to this problem with high density, nonvolatile memory that performs the MAC operation using the physical properties of the cell and crossbar. The challenge in-memory computing faces is not with weight transport, but rather with weight placement. To maximize throughput, weights must be distributed in a way that allows each CIM PE to be operating at all times to maximize throughput. One example of an optimal weight mapping and data flow was demonstrated in [145]. Using redundant weights and clever mapping strategies, they maximize throughput of a large-scale CIM accelerator.

D. Weight rotation

Another interesting technique to improve array utilization is weight rotation [151]. Given that ADCs make up most of the power consumption and area for small form factor eNVM PEs. Thus, rather than pipe-lining a CNN to make full use of the arrays, [151] redefine the challenge to making full use of the ADCs. To do this, every PE is allocated an array programmed with a portion of the weights from each layer. Then all arrays in a PE are connected to shared ADCs through multiplexers. In this way, arrays can be rotated based on the current layer and maximize use of the ADCs in each PE. This technique results in a small area footprint and large on-chip capacity, however, maximum throughput is reduced.

E. CIM-based architecture level designs

Numerous exhaustive CIM designs have been done with excellent power and performance evaluations. These works [103], [145], [148] consider both computational and energy efficiency measured as TOP/mm^2 and TOP/W, respectively. The key metric for evaluating DNN accelerators has been TOP/W, however, for CIM-based designs computational efficiency (TOP/mm^2) is particularly important. This is because current eNVM technology struggles to compete with CMOS-based designs in computational efficiency. CMOS designs have far less on-chip memory and can use this additional area for many low area PEs, increasing the computational efficiency. However, the lack of on-chip memory allows eNVM CIM designs to be far more energy-efficient.

In Table 2, we compare several large-scale designs in terms of storage, area, normalized TOP/mm^2, and normalized TOP/W. The different designs used different parameters which greatly impact performance. For this comparison, we scaled all designs to 8-bit inputs and 8-bit weights with 2 bits per NVM cell. In each of these designs the primary bottleneck in both performance and energy efficiency is the ADC. The ADC accounts for the majority of power and area, but their performance set the bottleneck for each design.

Table 2. Comparison of architectural designs.

Design	Storage (MB)	Area (mm^2)	TOPs/mm^2	TOPs/W
ISAAC	66.1	85.4	1.92	2.51
PipeLayer	82.63	82.6	2.97	0.29
AtomLayer	83.89	6.9	1.90	2.73
Yang et al.	66.1	-	2.21	4.19
Peng et al.	28.3	54.9	1.91	4.76

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

ISAAC [103] and PRIME [150] were the original exhaustive architecture-level simulations for CIM DNN accelerators. Both made large strides in mapping the design DNNs to CIM arrays utilizing layer pipe-lining and weight duplication. PipeLayer [148] maximized computational efficiency for training DNNs using a clever strategy in computing the gradients. AtomLayer maximized on-chip memory and minimized area using weight rotation achieving 83.69 MB in just 6.9 mm^2. Sparse ReRAM engine [120] took advantage of sparsity in both the feature inputs and the weights to greatly improve the computational and energy efficiency of ISAAC. Peng et al. [145] created a new weight mapping strategy and data flow to minimize data transport throughout the chip. This strategy is comparable to CMOS-based designs like those mentioned in [5] and [152].

Training CIM neural networks

Implementation of neural networks feature two distinct challenges: 1) inference and 2) training. Inference consists of just the forward pass or evaluation of the neural network model that we have discussed so far. Training is a more complex procedure that updates the weights in the network to optimize a loss function and improve performance. To train a network, we must perform inference and two more additional steps: 1) gradient calculation and 2) weight update.

After inference is performed on data samples and the network has made a prediction, the error is computed using the prediction and the label to compute the gradients for all weights in the network. These gradients represent the error of each weight with respect to the global loss function. Traditionally, backpropagation [153] is used to compute these gradients. Backpropagation computes the partial error of each weight in the network via the chain rule. Once acquired, these gradients are then applied to the network iteratively in the weight update phase.

The gradient calculation and weight update components of training present challenges to CIM that has been mostly avoided thus far. In particular, the additional complexity of transposable memory and the large number of write operations has motivated the use of off-chip learning. In the following sections, we outline challenges and current solutions for each of these steps.

Forward propagation (inference)

The forward pass computation of the neural network required to make a prediction is the inference phase. This features matrix multiplication, convolution, and the element-wise vector functions we discussed in the "Implementation of neural networks" section. Often times, models have already been trained offline and are ready to be deployed. For these applications our NVM will be programmed once and then not modified during the duration of the application. Under this assumption, we need perform only the forward pass of the neural network and do not consider the many complexities that come with training such as the cost of write energy or transposable memory. Mathematically, the feed forward computation can be written as

$$y_1 = W_1 \cdot x, a_1 = f\left(y_1\right) \tag{1}$$

$$y_2 = W_2 \cdot a_1, a_2 = f\left(y_2\right) \tag{2}$$

$$y_n = W_n \cdot a_{n-1}, a_n = f\left(y_n\right) \tag{3}$$

where x is the feature vector and W_i is the weight matrix connecting layer $i - 1$ to layer i ($y_0 = x$). The dot product of x and W_i yields y_i, and applying the nonlinear activation function f results in the activation at neuron i, a_i.

Backpropagation

To train state-of-the-art DNNs on chip, we rely on backpropagation or some variant which computes gradients of the weights with respect to a loss function. Backpropagation computes this gradient at each layer, starting from the last layer and propagating backward one layer at a time. This technique efficiently reuses computation, since the gradients of layers early in the network are functions of gradients from layers deeper in the network. The error at the last layer of the network, n, is the classification error e. BP computes the error at each hidden layer l, δa_l, by transposing the weight matrices W and multiplying by the gradient of the activation function. These layerwise computations for BP can be written as

$$\delta a_n = e \tag{4}$$

$$\delta a_2 = W_2^T \cdot \delta a_3 \odot f'\left(a_2\right) \tag{5}$$

$$\delta a_1 = W_1^T \cdot \delta a_2 \odot f'\left(a_1\right) \tag{6}$$

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

where ⊙ is the element-wise multiplication operator. In (5) and (6), the transpose of matrices W_2 and W_1 are used to compute the error at the hidden layers of the network.

Therefore, to compute the error at each of the hidden layers in our network, we use the transposed weights in the previous layer to backpropagate the error to the current layer. In CIM crossbars, this is a significant problem that can be solved by two methods.

- Read conductance values from the crossbar, and perform the multiplication in CMOS peripherals.
- Transposed matrix multiplication in the crossbar.

Reading the weights out requires that we spend significant time reading each weight one at a time, or a row at a time described by Li et al. [25]. Once complete, we must then perform the matrix multiplication in CMOS. This method limits the advantages of using in-memory computing because we revert to a von Neumann computing architecture in the backward direction.

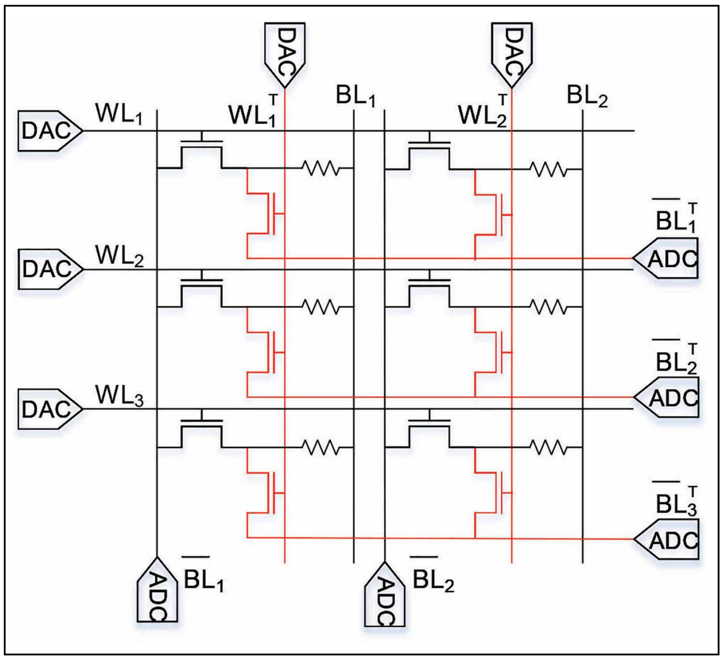

Figure 16. 2T1R circuit for transposed matrix multiplication. The transpose logic is shown in red. The 2T1R configuration requires additional select transistor, wordline, BL, and ADC.

Using additional circuitry, it is possible to transpose the weight matrix so that the backward pass can be computed in memory [154], [155]. Transposing the weight matrix requires significant circuit overhead which increases area, power, and design complexity. The 2T1R synapse circuit is shown in a crossbar array in Figure 16. The circuit elements highlighted in red represent the portion of the circuit for the transposed read and matrix multiplication. This additional circuity comes at a significant overhead in area and design complexity. Additional select transistors, wordline drivers, BL drivers, and ADCs are required to read the transposed data. Such a design greatly reduces inference computational efficiency since this overhead is not needed in the forward direction.

Network update

Once we have computed the gradients using backpropagation, we must apply the gradients to the network. The error with respect to each weight is computed as the outer product of input values and error: $\vec{a} \cdot \vec{e}$. This results in a matrix, δW, that we must apply to our crossbar. To update an eNVM cell, we must apply a positive voltage to "set" the device (lower resistance) or negative voltage to "reset" (increase resistance).

There are different ways to apply the error update to the crossbar. For digital CIM, the most common approach is to compute δW in CMOS and then write each row one at a time. Like computing the backward pass, this limits the advantages of using CIM because we need to move the two vectors $\vec{a}$ and $\vec{e}$ to CMOS logic and then perform $N\,M$ MAC operations. After which, we have to transport a matrix with the dimensions of the size of vectors $\vec{a}$ and $\vec{e}$ back from logic to the array. This technique not only struggles from data movement, but also in storing the gradient of the matrix in SRAM. While we have high density eNVM on chip to store weight matrices, there will not be sufficient SRAM to store a large gradient. Another technique [156] showed that the eNVM arrays can be reprogrammed with the error gradients so that weight gradients could be computed with CIM. However, this method requires significant data transfer and high power writing of eNVM.

For analog CIM, there exists an extremely efficient method for performing the weight update that has been proposed in [24], [157], and [158]. To update the eNVM network we perform outer product of $\vec{a}$

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

and $\vec{e}$ with time and the results are integrated by the eNVM. If we apply pulses proportional to the input values on the word lines, and pulses proportional to the errors on the columns then the time overlap of the two is proportional to the product. In Figure 17, we demonstrate an example where WL is asserted for 40% of the period and $\overline{BL}$ is asserted for 80% of its period. The resulting voltage across the cell is roughly 32% of the period. It is necessary that the period of $\overline{BL}$ is several times shorter than WL, otherwise a min function is performed rather than multiplication.

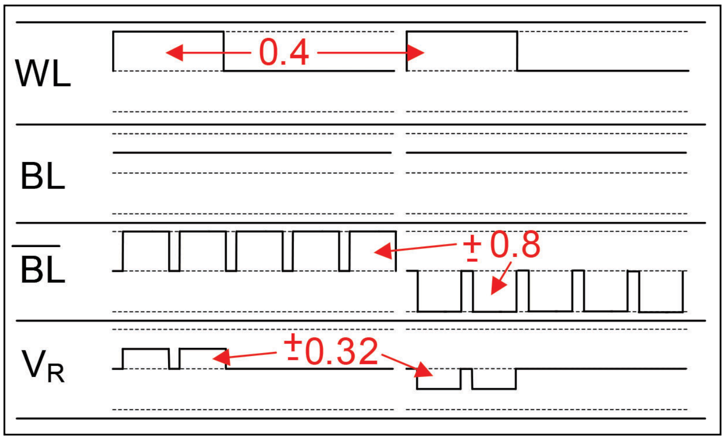

Figure 17. Waveform showing pulsed-based multiplication of positive and negative voltages applied to eNVM array. *Wordline* (*WL*) and $\overline{\textit{Bitline}}$ ($\overline{\textit{BL}}$) are the two inputs. The result of multiplication is V_R.

Alternatives to backpropagation

Given the difficulty of training eNVM on chip, most works train the model off chip and simply perform inference on the eNVM CIM accelerator. The difficulty of training on chip comes primarily from the gradient calculation phase in training. In both CMOS- and eNVM-based designs, gradient calculation using backpropagation requires excessive data transport and doubles the number of computations required by a training example. For eNVM-based designs, backpropagation also requires additional hardware for training that reduces inference performance per unit area.

An interesting alternative to backpropagation inspired from biological perspective was proposed in [159]. Feedback alignment (FA) uses fixed random feedback weights to propagate the errors back through the layers of a DNN rather than using the actual network weights to compute the partial error. Consequently, the weights in the shallow layers of the network no longer need information of the weights of all the deeper layers. Building on top of this, Nøkland [160] proposes direct FA (DFA), where it was shown that the feedback to shallow layers need not be propagated through all the layers. DFA showed that instead, a random matrix can be used to compute the error at each layer. Such a matrix can be randomly initialized with size N C, where N is the number of hidden neurons at a hidden layer, and C is the number of classes (or outputs) in the network. This matrix is used to create a linear projection of the error for each hidden neuron.

Decoupling the forward and backward weights makes DFA a suitable algorithm for training networks that rely on in-memory computing. The advantage of in-memory computing is that in the forward pass, the weights do not need to be read out and brought to compute. Similarly, with DFA we no longer need to read weights out during the backward pass. Instead a random feedback matrix can be used to compute the error and update the network independent of the forward weights. Using these techniques, several works [158], [161]–[164] have achieved lower hardware complexity and improved performance.

Compensating for large write latency of eNVM

All the eNVM technologies that we have discussed so far promise high density, short read latency, no leakage power and voltage/process scalability with scaled processor logic. However, all these technologies require high write energy and typically have higher write latency. For edge devices where only inference is performed, the write latency is not critical. But in systems with real-time performance requirement, such as real-time RL in drones, robots, and UAVs, it is challenging to maintain high speed while being able to write into the eNVM. To address this challenge, Yoon et al. [165] recently proposed a method of transfer learning with real-time RL over a small portion of the model weights. In a typical application of drones, it has been shown that the initial model weight can be trained off-line using a simulation environment and the weights of the shallow layers are transferred to the eNVM. Next the drone is deployed in a real environment and the last few layers of the network, which are stored on an on-die SRAM, are updated in real-time. It has been shown that such a system performs at almost identical performance

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

with more than 70% decrease in system energy and latency compared to end-to-end learning [165].

Discussions and outlook

From enabling intelligent microrobotics to training massive neural networks, both the applications and progress of research in artificial intelligence are bounded by the tools we have to design with. CMOS has carried us so far, but ultimately we must find alternatives that better suit the massively parallel and data intensive needs to artificial intelligence and machine learning. In-memory computing is a competitive design choice, where there is minimal overhead in transporting data. Increased research effort from both the industry and the academia as well as consistent breakthroughs in device quality make emerging nonvolatile memories a promising technological advancement. ■

Acknowledgments

This work was supported by the U.S. Department of Defense's Multidisciplinary University Research Initiatives (MURI) Program under Grant FOA: N00014-16-R-FO05, the Applications and Systems Driven Center for Energy-Efficient Integrated Nano-Technologies (ASCENT); in part by the Center of Brain Inspired Computing (C-BRIC) though the Joint University Microelectronics Program sponsored by DARPA; and in part by the Semiconductor Research Corporation.

References

[1] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[2] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

[3] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.

[4] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 2014, pp. 10–14.

[5] Y.-H. Chen et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[6] T. Chen et al., "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Comput. Archit. News*, vol. 49, no. 4, pp. 269–284, Apr. 2014.

[7] Y. Chen et al., "DaDianNao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchit.*, Dec. 2014, pp. 609–622.

[8] H. Kwon, A. Samajdar, and T. Krishna, "MAERI: Enabling flexible dataflow mapping over DNN accelerators via reconfigurable interconnects," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 461–475, Nov. 2018.

[9] H. Kwon, A. Samajdar, and T. Krishna, "A communication-centric approach for designing flexible DNN accelerators," *IEEE Micro*, vol. 38, no. 6, pp. 25–35, Nov. 2018.

[10] C. Kim et al., "A 2.1TFLOPS/W mobile deep RL accelerator with transposable PE array and experience compression," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 136–138.

[11] D. Shin et al., "14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 240–241.

[12] N. Cao, M. Chang, and A. Raychowdhury, "14.1 a 65 nm 1.1-to-9.1TOPS/W hybrid-digital-mixed-signal computing platform for accelerating model-based and model-free swarm robotics," in *IEEE Int. Solid- State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 222–224.

[13] D. Bankman et al., "An always-on 3.8 μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, Jan. 2019.

[14] Y.-H. Chenet al., "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, 2019.

[15] P. N. Whatmough et al., "14.3 a 28 nm SoC with a 1.2 GHz 568 nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 242–243.

[16] J. Lee et al., "7.7 LNPU: A 25.3TFLOPS/W sparse deep-neural-network learning processor with fine-grained mixed precision of FP8-FP16," in *IEEE Int. Solid- State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 142–144.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

[17] F. Karimzadeh et al., “Hardware-aware pruning of DNNs using LFSR-generated pseudo-random indices,” 2019, *arXiv:1911.04468*. [Online]. Available: http://arxiv.org/abs/1911.04468

[18] E. Qin et al., “SIGMA: A sparse and irregular GEMM accelerator with flexible interconnects for DNN training,” in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2020, pp. 58–70.

[19] P. A. Merolla et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[20] M. Davies et al., “Loihi: A neuromorphicmanycore processor with on-chip learning,” *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[21] S. Yu and P.-Y. Chen, “Emerging memory technologies: Recent trends and prospects,” *IEEE Solid State Circuits Mag.*, vol. 8, no. 2, pp. 43–56, Feb. 2016.

[22] Y. J. Song et al., “Demonstration of highly manufacturable STT-MRAM embedded in 28 nm logic,” in *IEDM Tech. Dig.*, Dec. 2018, pp. 18–22.

[23] S. Yu et al., “Binary neural network with 16 mb RRAM macro chip for classification and online training,” in *IEDM Tech. Dig.*, Dec. 2016, pp. 16–22.

[24] S. Kim et al., “NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous *in-situ* learning,” in *IEDM Tech. Dig.*, Dec. 2015, pp. 17–21.

[25] C. Li et al., “Efficient and self-adaptive *in-situ* learning in multilayer memristor neural networks,” *Nature Commun.*, vol. 9, no. 1, p. 2385, Dec. 2018.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton “ImageNet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[27] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[28] K. He et al., “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[29] O. Vinyals et al., “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, pp. 350–354, Oct. 2019.

[30] N. Shazeer et al., “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” 2017, *arXiv:1701.06538*. [Online]. Available: http://arxiv.org/abs/1701.06538

[31] H. Touvron et al., “Fixing the train-test resolution discrepancy,” 2019, *arXiv:1906.06423*. [Online]. Available: http://arxiv.org/abs/1906.06423

[32] S. Yu, “Neuro-inspired computing with emerging nonvolatilememorys,” *Proc. IEEE*, vol. 106, no. 2, pp. 260–285, Feb. 2018.

[33] Q. Dong et al., “A 1 Mb 28 nm STT-MRAM with 2.8 ns read access time at 1.2 V VDD using single-cap offset-cancelled sense amplifier and *in-situ* self-write-termination,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 480–482.

[34] K. Tsuchida et al., “A 64 Mb MRAM with clamped-reference and adequate-reference schemes,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2010, pp. 258–259.

[35] H. Noguchi et al., “7.5 a 3.3 ns-access-time 71.2 μW/MHz 1 Mb embedded STT-MRAM using physically eliminated read-disturb scheme and normally-off memory architecture,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2015, pp. 1–3.

[36] J. Y. Wu et al., “A 40 nm low-power logic compatible phase change memory technology,” in *IEDM Tech. Dig.*, Dec. 2018, pp. 27–36.

[37] Z. T. Song et al., “High endurance phase change memory chip implemented based on carbon-doped $Ge_2Sb_2Te_5$ in 40 nm node for embedded application,” in *IEDM Tech. Dig.*, Dec. 2018, pp. 27–35.

[38] B. Govoreanu et al., “10×10nm^{2}Hf/HfO *pp. 31–36, IEEE, 2011.* crossbar resistive RAM with excellent performance, reliability and low-energy operation,” in *IEDM Tech. Dig.*, pp. 31–36, IEEE, 2011.

[39] P. Jain et al., “13.2 a 3.6Mb 10.1Mb/mm2 embedded non-volatile ReRAM macro in 22 nm FinFET technology with adaptive forming/set/reset schemes yielding down to 0.5 V with sensing time of 5ns at 0.7 V,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 212–214.

[40] T. F. Wu et al., “14.3 a 43pJ/cycle non-volatile microcontroller with 4.7μs shutdown/wake-up integrating 2.3-bit/Cell resistive RAM and resilience techniques,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 226–228.

[41] M. Trentzsch et al., “A 28 nm HKMG super low power embedded NVM technology based on ferroelectric FETs,” in *IEDM Tech. Dig.*, Dec. 2016, pp. 11–15.

[42] S. George et al., “Nonvolatile memory design based on ferroelectric FETs,” in *Proc. 53rd Annu. Design Autom. Conf. (DAC)*, 2016, pp. 1–6.

[43] M. Qazi et al., “A low-voltage 1 Mb FRAM in 0.13 μm CMOS featuring time-to-digital sensing for expanded

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

operating margin," *IEEE J. Solid-State Circuits*, vol. 47, no. 1, pp. 141–150, 2011.

[44] H.-S. P. Wong et al., "Metal-oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012, doi: 10.1109/JPROC.2012.\break2190369.

[45] S.-S. Sheu et al., "A 4 Mb embedded SLC resistive-RAM macro with 7.2 ns read-write random-access time and 160ns MLC-access capability," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2011, pp. 200–202.

[46] H. Y. Lee et al., "Low power and high speed bipolar switching with a thin reactive ti buffer layer in robust HfO_2 based RRAM," in *IEDM Tech. Dig.*, Dec. 2008, pp. 1–4.

[47] S. Yu, Y. Wu, and H.-S.-P.Wong, "Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory," *Appl. Phys. Lett.*, vol. 98, no. 10, Mar. 2011, Art.no. 103514.

[48] S. Yu et al., "A neuromorphic visual system using RRAM synaptic devices with sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling," in *IEDM Tech. Dig.*, Dec. 2012, pp. 10–14.

[49] S. R. Ovshinsky, "Reversible electrical switching phenomena in disordered structures," *Phys. Rev. Lett.*, vol. 21, no. 20, p. 1450, Nov. 1968.

[50] N. Yamada et al., "High speed overwritable phase change optical disk material," *Jpn. J. Appl. Phys.*, vol. 26, no. S4, p. 61, Jan. 1987.

[51] M. Wuttig and N. Yamada, "Phase-change materials for rewriteable data storage," *Nature Mater.*, vol. 6, no. 11, p. 824, 2007.

[52] S. Raoux, "Phase change materials," *Annu. Rev. Mater. Res.*, vol. 39, pp. 25–48, Aug. 2009.

[53] S. Lai, "Current status of the phase change memory and its future," in *IEDM Tech. Dig.*, 2003, pp. 10–11.

[54] A. Sebastian, M. Le Gallo, and E. Eleftheriou, "Computational phase-change memory: Beyond von Neumann computing," *J. Phys. D, Appl. Phys.*, vol. 52, no. 44, Oct. 2019, Art. no. 443002.

[55] S. Ambrogio et al., "Unsupervised learning by spike timing dependent plasticity in phase change memory (PCM) synapses," *Frontiers Neurosci.*, vol. 10, p. 56, Mar. 2016.

[56] M. He et al., "Ultra-low program current and multilevel phase change memory for high-density storage achieved by a low-current SET pre-operation," *IEEE Electron Device Lett.*, vol. 40, no. 10, pp. 1595–1598, Oct. 2019.

[57] A. R. A. L. Lacaita, "Electrothermal and phase-change dynamics in chalcogenide-based memories," in *IEDM Tech. Dig.*, 2004, pp. 911–914.

[58] F. Bedeschi et al., "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, Jan. 2009.

[59] J. Orava et al., "Characterization of supercooled liquid $Ge_2Sb_2Te_5$ and its crystallization by ultrafast-heating calorimetry," *Nature Mater.*, vol. 11, no. 4, p. 279, 2012.

[60] A. Sebastian, M. Le Gallo, and D. Krebs, "Crystal growth within a phase change memory cell," *Nature Commun.*, vol. 5, no. 1, p. 4314, Sep. 2014.

[61] D. Adler et al., "Threshold switching in chalcogenide-glass thin films," *J. Appl. Phys.*, vol. 51, no. 6, pp. 3289–3309, 1980.

[62] D. Ielmini et al., "Analysis of phase distribution in phase-change nonvolatile memories," *IEEE Electron Device Lett.*, vol. 25, no. 7, pp. 507–509, Jul. 2004.

[63] D. Ielmini, A. L. Lacaita, and D. Mantegazza, "Recovery and drift dynamics of resistance and threshold voltages in phase-change memories," *IEEE Trans. Electron Devices*, vol. 54, no. 2, pp. 308–315, Feb. 2007.

[64] S. Lavizzari et al., "Transient effects of delay, switching and recovery in phase change memory (PCM) devices," in *IEDM Tech. Dig.*, Dec. 2008, pp. 1–4.

[65] M. Le Gallo et al., "Evidence for thermally assisted threshold switching behavior in nanoscale phase-change memory cells," *J. Appl. Phys.*, vol. 119, no. 2, Jan. 2016, Art.no. 025704.

[66] D. Ielmini and Y. Zhang, "Analytical model for subthreshold conduction and threshold switching in chalcogenide-based memory devices," *J. Appl. Phys.*, vol. 102, no. 5, Sep. 2007, Art. no. 054517.

[67] A. Pirovano et al., "Low-field amorphous state resistance and threshold voltage drift in chalcogenide materials," *IEEE Trans. Electron Devices*, vol. 51, no. 5, pp. 714–719, May 2004.

[68] M. Boniardi and D. Ielmini, "Physical origin of the resistance drift exponent in amorphous phase change materials," *Appl. Phys. Lett.*, vol. 98, no. 24, Jun. 2011, Art.no. 243506.

[69] M. Le Gallo et al., "The complete time/temperature dependence of I-V drift in PCM devices," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Apr. 2016, p. MY-1.

[70] M. Le Gallo et al., "Collective structural relaxation in phase-change memory devices," *Adv. Electron. Mater.*, vol. 4, no. 9, Sep. 2018, Art. no. 1700627.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

[71] P. Fantini et al., “Experimental investigation of transport properties in chalcogenide materials through 1/f noise measurements,” *Appl. Phys. Lett.*, vol. 88, no. 26, Jun. 2006, Art.no. 263506.

[72] T. Nirschl et al., “Write strategies for 2 and 4-bit multi-level phase-change memory,” in *IEDM Tech. Dig.*, Dec. 2007, pp. 461–464.

[73] A. Pantazi et al., “Multilevel phase change memory modeling and experimental characterization,” in *Proc. EPCOS*, 2009, pp. 1–8.

[74] N. Papandreou et al., “Programming algorithms for multilevel phase-change memory,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2011, pp. 329–332.

[75] G. BettiBeneventi, M. Ferro, and P. Fantini, “1/f noise in 45-nm RESET-state phase-change memory devices: Characterization, impact on memory readout operation, and scaling perspectives,” *IEEE Electron Device Lett.*, vol. 33, no. 11, pp. 1559–1561, Nov. 2012.

[76] A. Sebastian et al., “Non-resistance-based cell-state metric for phase-change memory,” *J. Appl. Phys.*, vol. 110, no. 8, Oct. 2011, Art. no. 084505.

[77] M. Stanisavljevic et al., “Phase-change memory: Feasibility of reliable multilevel-cell storage and retention at elevated temperatures,” in *Proc. IEEE Int. Rel. Phys. Symp.*, Apr. 2015.

[78] M. Le Gallo et al., “Compressed sensing with approximate message passing using in-memory computing,” *IEEE Trans. Electron Devices*, vol. 65, no. 10, pp. 4304–4312, Oct. 2018.

[79] M. Le Gallo et al., “Mixed-precision in-memory computing,” *Nature Electron.*, vol. 1, no. 4, p. 246, 2018.

[80] D. Kuzum et al., “Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing,” *Nano Lett.*, vol. 12, no. 5, pp. 2179–2186, 2011.

[81] A. Sebastian et al., “Temporal correlation detection using computational phase-change memory,” *Nature Commun.*, vol. 8, no. 1, p. 1115, Dec. 2017.

[82] O. Golonzka et al., “MRAM as embedded non-volatile memory solution for 22FFL FinFET technology,” in *IEDM Tech. Dig.*, Dec. 2018, pp. 18.1.1–18.1.4.

[83] A. Chintaluri et al., “Analysis of defects and variations in embedded spin transfer torque (STT) MRAM arrays,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 3, pp. 319–329, Sep. 2016.

[84] A. Aziz et al., “Computing with ferroelectric FETs: Devices, models, systems, and applications,” in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2018, pp. 1289–1298.

[85] J. Chen, M. P. Harmer, and D. M. Smyth, “Compositional control of ferroelectric fatigue in perovskite ferroelectric ceramics and thin films,” *J. Appl. Phys.*, vol. 76, no. 9, pp. 5394–5398, Nov. 1994.

[86] M. H. Lee et al., “Prospects for ferroelectric $HfZrO_x$ FETs with experimentally CET=0.98 nm, SS_{for}=42 mV/dec, SS_{rev}=28 mV/dec, switch-off 0.2 V, and hysteresis-free strategies,” in *IEDM Tech. Dig.*, Dec. 2015, pp. 22–25.

[87] S. Salahuddin and S. Datta, “Use of negative capacitance to provide voltage amplification for low power nanoscale devices,” *Nano Lett.*, vol. 8, no. 2, pp. 405–410, Feb. 2008.

[88] S. George et al., “Device circuit co design of FEFET based logic for low voltage processors,” in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 649–654.

[89] X. Yin et al., “Exploiting ferroelectric FETs for low-power non-volatile logic-in-memory circuits,” in *Proc. 35th Int. Conf. Comput.-Aided Design*, Nov. 2016, p. 121.

[90] D. Wang et al., “Ferroelectric transistor based non-volatile flip-flop,” in *Proc. Int. Symp. Low Power Electron. Design* (*ISLPED*), 2016, pp. 10–15.

[91] A. Sharma and K. Roy, “1T non-volatile memory design using sub-10nm ferroelectric FETs,” *IEEE Electron Device Lett.*, vol. 39, no. 3, pp. 359–362, Mar. 2018.

[92] X. Chen et al., “Power and area efficient FPGA building blocks based on ferroelectric FETs,” *IEEE Trans. Circuits Syst. I*, *Reg. Papers*, vol. 66, no. 5, pp. 1780–1793, May 2019.

[93] X. Chen, M. Niemier, and X. S. Hu, “Nonvolatile lookup table design based on ferroelectric field-effect transistors,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–5.

[94] X. Chen et al., “Design and optimization of FeFET-based crossbars for binary convolution neural networks,” in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2018, pp. 1205–1210.

[95] M. Jerry et al., “A ferroelectric field effect transistor based synaptic weight cell,” *J. Phys. D, Appl. Phys.*, vol. 51, no. 43, Oct. 2018, Art. no. 434001.

[96] Z. Wang and A. I. Khan, “Ferroelectric relaxation oscillators and spiking neurons,” *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 5, no. 2, pp. 151–157, Dec. 2019.

[97] Y. Fanget al., “Neuro-mimetic dynamics of a ferroelectric FET-based spiking neuron,” *IEEE Electron Device Lett.*, vol. 40, no. 7, pp. 1213–1216, Jul. 2019.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

[98] Y. Fang et al., "A swarm optimization solver based on ferroelectric spiking neural networks," *Frontiers Neurosci.*, vol. 13, p. 855, Aug. 2019.

[99] J. Wang et al., "Assimilation of biophysical neuronal dynamics in neuromorphic VLSI," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 6, pp. 1258–1270, Dec. 2017.

[100] E. Donati et al., "Deriving optimal silicon neuron circuit specifications using data assimilation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[101] V. Sze et al., "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[102] D. B. Strukov et al., "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

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

[104] G. Close et al., "A 512 Mb phase-change memory (PCM) in 90 nm CMOS achieving 2b/cell," in *Symp. VLSI Circuits Dig. Tech. Papers*, pp. 202–203, IEEE, 2011.

[105] J.-W. Kwon et al., "A two-step 5b logarithmic ADC with minimum step-size of 0.1% full-scale for MLC phase-change memory readout," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2014, pp. 1–4.

[106] A. Athmanathan et al., "A 6-bit drift-resilient readout scheme for multi-level phase-change memory," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2014, pp. 137–140.

[107] W. R. Reohr et al., "High voltage word line driver," U.S. Patent 8 120 968, Feb. 21, 2012.

[108] G. B. Bronner, S. H. Dhong, and W. Hwang, "Word line driver circuit for dynamic random access memories," U.S. Patent 5 253 202, Oct. 12 1993.

[109] I. Arsovski et al., "Word-line level shift circuit," U.S. Patent 8 218 378, Jul. 10, 2012.

[110] S. R. Cottier et al., "Level shifter for boosting wordline voltage and memory cell performance," U.S. Patent 7 710 796, May 4, 2010.

[111] O. Hirabayashi et al., "A process-variation-tolerant dual-power-supply SRAM with 0.179 μm^2 Cell in 40 nm CMOS using level-programmable wordline driver," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2009, pp. 458–459.

[112] H.-H.Lu and C.-F. Wang, "High voltage wordline driver with a three stage level shifter," U.S. Patent 7 283 406, Oct. 16, 2007.

[113] V. Stojanovic et al., "Energy-delay tradeoffs in combinational logic using gate sizing and supply voltage optimization," in *Proc. 28th Eur. Solid-State Circuits Conf.*, Sep. 2002,pp. 211–214.

[114] I. Hubara et al., "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.

[115] M. Rastegari et al., "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis,* 2016, pp. 525–542.

[116] X. Sun et al., "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2018, pp. 1423–1428.

[117] X. Sun et al., "Fully parallel RRAM synaptic array for implementing binary neural network with (+1,–1) weights and (+1, 0) neurons," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2018, pp. 574–579.

[118] J. Yue et al., "14.3 a 65 nm computing-in-memory-based CNN processor with 2.9-to-35.8TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 234–236.

[119] P. Wang et al., "SNrram: An efficient sparse neural network computation architecture based on resistive random-access memory," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[120] T.-H. Yang et al., "Sparse ReRAM engine: Joint exploration of activation and weight sparsity in compressed neural networks," in *Proc. 46th Int. Symp. Comput. Archit.*, Jun. 2019, pp. 236–249.

[121] N. Gong et al., "Signal and noise extraction from analog memory elements for neuromorphic computing," *Nature Commun.*, vol. 9, no. 1, pp. 1–8, May 2018.

[122] A. Grossi et al., "Fundamental variability limits of filament-based RRAM," in *IEDM Tech. Dig.*, Dec. 2016, pp. 4–7.

[123] B. Crafton, S. Spetalnick, and A. Raychowdhury, "Counting cards: Exploiting weight and variance distributions for robust compute in-memory," 2020, *arXiv:2006.03117*. [Online]. Available: http://arxiv.org/abs/2006.03117

[124] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[125] M. Jaderberg et al., "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

[126] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[127] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[128] G. Huang et al., “Densely connected convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[129] R. Krishnamoorthi, “Quantizing deep convolutional networks for efficient inference: A whitepaper,” 2018, *arXiv:1806.08342*. [Online]. Available: http://arxiv.org/abs/1806.08342

[130] Y. Long, T. Na, and S. Mukhopadhyay, “ReRAM-based processing-in-memory architecture for recurrent neural network acceleration,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 12, pp. 2781–2794, Dec. 2018.

[131] C. L. Lawson et al., “Basic linear algebra subprograms for fortran usage,” *ACM Trans. Math. Softw. (TOMS)*, vol. 5, no. 3, pp. 308–323, Sep. 1979.

[132] M. Courbariaux et al., “Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1,” 2016, *arXiv:1602.02830*. [Online]. Available: http://arxiv.org/abs/1602.02830

[133] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, “Fixed point quantization of deep convolutional networks,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2849–2858.

[134] B. Jacob et al., “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[135] J. Deng et al., “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[136] N. Wang et al., “Training deep neural networks with 8-bit floating point numbers,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7675–7684.

[137] S. Wu et al., “Training and inference with integers in deep neural networks,” 2018, *arXiv:1802.04680*. [Online]. Available: http://arxiv.org/abs/1802.04680

[138] R. Banner et al., “Scalable methods for 8-bit training of neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5145–5153.

[139] X. Dong et al., “NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[140] P.-Y. Chen, X. Peng, and S. Yu, “NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.

[141] L. Xia et al., “MNSIM: Simulation platform for memristor-based neuromorphic computing system,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 5, pp. 1009–1022, May 2018.

[142] S. Jain et al., “RxNN: A framework for evaluating deep neural networks on resistive crossbars,” 2018, *arXiv:1809.00072*. [Online]. Available: http://arxiv.org/abs/1809.00072

[143] C. Fallin, C. Craik, and O. Mutlu, “CHIPPER: A low-complexity bufferless deflection router,” in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, Feb. 2011, pp. 144–155.

[144] S. K. Mandal et al., “Analytical performance models for NoCs with multiple priority traffic classes,” *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 5s, pp. 1–21, Oct. 2019.

[145] X. Peng, R. Liu, and S. Yu, “Optimizing weight mapping and data flow for convolutional neural networks on processing-in-memory architectures,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 4, pp. 1333–1343, Apr. 2020.

[146] D. Vantrease et al., “Corona: System implications of emerging nanophotonic technology,” *ACM SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 153–164, 2008.

[147] S. Deb et al., “Wireless NoC as interconnection backbone for multicore chips: Promises and challenges,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 2, pp. 228–239, Jun. 2012.

[148] L. Song et al., “PipeLayer: A pipelined ReRAM-based accelerator for deep learning,” in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 541–552.

[149] B. Crafton et al., “Breaking barriers: Maximizing array utilization for compute in-memory fabrics,” in *Proc. IFIP/IEEE 28th Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, 2020, pp. 1–6.

[150] P. Chi et al., “PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory,” *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 27–39, Oct. 2016.

[151] X. Qiao et al., “AtomLayer: A universal ReRAM-based CNN accelerator with atomic layer computation,” in

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.

Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC), Jun. 2018, pp. 1–6.

[152] H. Kwon et al., “Understanding reuse, performance, and hardware cost of DNN dataflows: A data-centric approach using MAESTRO,” 2018, *arXiv:1805.02566*. [Online]. Available: http://arxiv.org/abs/1805.02566

[153] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, p. 533, 1986.

[154] R. Hasan and T. M. Taha, “Enabling back propagation training of memristor crossbar neuromorphic processors,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2014, pp. 21–28.

[155] D. Soudry et al., “Memristor-based multilayer neural networks with online gradient descent training,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2408–2421, Oct. 2015.

[156] X. Peng et al., “DNN+NeuroSim V2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training,” 2020, *arXiv:2003.06471*. [Online]. Available: http://arxiv.org/abs/2003.06471

[157] T. Gokmen, M. Onen, and W. Haensch, “Training deep convolutional neural networks with resistive cross-point devices,” *Frontiers Neurosci.*, vol. 11, p. 538, Oct. 2017.

[158] B. Crafton et al., “Local learning in RRAM neural networks with sparse direct feedback alignment,” in *Proc. IEEE/ACM Int. Symp.Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.

[159] T. P. Lillicrap et al., “Random synaptic feedback weights support error backpropagation for deep learning,” *Nature Commun.*, vol. 7, no. 1, p. 13276, Dec. 2016.

[160] A. Nøkland, “Direct feedback alignment provides learning in deep neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1037–1045.

[161] D. Han et al., “A 1.32 TOPS/W energy efficient deep neural network learning processor with direct feedback alignment based heterogeneous core architecture,” in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. C304–C305.

[162] J. Park, J. Lee, and D. Jeon, “A 65-nm neuromorphic image classification processor with energy-efficient training through direct spike-only feedback,” *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 108–119, Jan. 2020.

[163] B. Craftonet al., “Direct feedback alignment with sparse connections for local learning,” *Frontiers Neurosci.*, vol. 13, p. 525, May 2019.

[164] C. Frenkel, M. Lefebvre, and D. Bol, “Learning without feedback: Direct random target projection as a feedback-alignment algorithm with layerwisefeedforward training,” 2019, *arXiv:1909.01311*. [Online]. Available: http://arxiv.org/abs/1909.01311

[165] I. Yoon et al., “Transfer and online reinforcement learning in STT-MRAM based embedded systems for autonomous drones,” in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2019, pp. 1489–1494.

Brian Crafton is currently pursuing a PhD with the Georgia Institute of Technology, Atlanta, GA, USA, under the supervision of Dr. Raychowdhury. His research interest includes in-memory and near-memory computing for energy efficient machine learning. Crafton has a BS in computer engineering from Northeastern University, Boston, MA (2017). He is a Student Member of IEEE.

Samuel Spetalnick is currently pursuing a PhD with Georgia Institute of Technology, Atlanta, GA, USA. His research interest includes applications and circuits for emerging nonvolatile memory devices. Spetalnick has a BS in EE and CE from Johns Hopkins University, Baltimore, MD, USA.

Yan Fang is currently a Postdoctoral Researcher with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. His research interests include brain-inspired computing systems based on emerging nanodevices, as well as smart materials that compute and applications in machine intelligence. Fang has an MS and a PhD in electrical and computer engineering from the University of Pittsburgh, Pittsburgh, PA, USA.

Arijit Raychowdhury is currently a Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA, where he joined in January 2013. His research interests include low power digital and mixed-signal circuit design, design of power converters, and sensors and exploring interactions of circuits with device technologies. Raychowdhury has a PhD in electrical and computer engineering from Purdue University, West Lafayette, IN, USA (2007). He is a Senior Member of IEEE.

■ Direct questions and comments about this article to Arijit Raychowdhury, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA; arijit.raychowdhury@ece.gatech.edu.

Authorized licensed use limited to: Georgia Institute of Technology. Downloaded on June 28,2021 at 12:46:45 UTC from IEEE Xplore. Restrictions apply.