AI hardware acceleration with analog memory: Microarchitectures for low energy at high speed

H.-Y. Chang
P. Narayanan
S. C. Lewis
N. C. P. Farinha
K. Hosokawa
C. Mackin
H. Tsai
S. Ambrogio
A. Chen
G. W. Burr

In this article, we present innovative microarchitectural designs for multilayer deep neural networks (DNNs) implemented in crossbar arrays of analog memories. Data is transferred in a fully parallel manner between arrays without explicit analog-to-digital converters. Design ideas including source follower-based readout, array segmentation, and transmit-by-duration are adopted to improve the circuit efficiency. The execution energy and throughput, for both DNN training and inference, are analyzed quantitatively using circuit simulations of a full CMOS design in the 90-nm technology node. We find that our current design could achieve up to 12–14 TOPs/s/W energy efficiency for training, while a projected scaled design could achieve up to 250 TOPs/s/W. Key challenges in realizing analog AI systems are discussed.

1 Introduction

State-of-the-art artificial intelligence (AI) has demonstrated unprecedented capabilities in the detection, recognition, and classification of complex features from real-world input such as images, speech, and text. Such deep neural networks (DNNs) [1] are typically trained with graphics processing units (GPUs) using repeated batch-based exposure of the network to a large training set, with weights updated via the backpropagation algorithm [2]. As such, both the "training" and the "forward inference" of previously trained DNNs depend critically on the speed and power with which large multiply accumulate operations can be performed. In order to continue scaling up the size of these AI systems, new hardware accelerators that can efficiently yet rapidly perform these computations will need to be developed.

Any such hardware accelerator will need to deliver high throughput (exceeding one trillion operations per second) and low-latency DNN computations with high energy efficiency, without sacrificing neural network accuracy. One approach is to design custom digital accelerators [3], "reimagining the GPU as if it had been expressly designed for deep learning." Another approach is to switch to a non-von Neumann scheme that would allow the computations to be performed at the location of the stored weight data [4].

Digital Object Identifier: 10.1147/JRD.2019.2934050

DNN hardware accelerators based on crossbar arrays of analog nonvolatile memories (NVMs) such as phase change memory (PCM) [5] or ReRAM (RRAM) [6] can speed up both training and forward inference of DNNs. Such analog accelerators naturally implement large vector-matrix-multiplication operations in $O(1)$ time by performing the multiplication with Ohm's law at the location of the memory devices and the accumulation with Kirchhoff's law for currents [4]. Small-scale array-level demonstrations have been carried out, including a single-layer perceptron for classifying a 3×3 image into three classes [7], a sparse-coding algorithm on a 32×32 array [8], and a 128×64 array for implementing MNIST [9].

Researchers have already shown that DNNs can be trained using the backpropagation algorithm in multiple ways, including both fully *in situ* training [10–13] and *ex situ* weight update with *in situ* forward and reverse propagation [14], as well as forward inference of previously trained networks [15, 16]. Specifications for the device and material needs were identified [13, 17, 18], leading to new materials candidates [19–21], new memory-cell designs for existing materials [22, 23], new circuit structures [24–26], new training techniques [25, 27], and the introduction of multisynaptic architectures [25, 28, 29], which have all helped improve training accuracy to the point where it

0018-8646/19 © 2019 IBM

can deliver the performance of the underlying neural networks [25].

However, while the energy benefits for hardware accelerators computed for a single crossbar array are impressive, DNNs involve a large number of layers. Thus, it is critical to measure and transmit the collected multiply accumulate results from one crossbar to the next in an extremely efficient manner while implementing the appropriate nonlinear squashing function for forward propagation (or the multiplication by the appropriate derivative during reverse propagation) [30]. Most researchers who discuss the transmission of data from the "south" edge of one crossbar array to the "west" edge of the next always invoke analog-to-digital converters (ADCs) [13, 15, 31]. However, if one implements enough ADCs to keep the latency low and throughput high, then the system power tends to be dominated by the ADCs and not by the crossbar itself [13, 31]; the ADCs and digital circuitry can potentially occupy up to 10× the area of the memory array itself [31]. Alternatively, one can keep area and power (but not energy) low by multiplexing many multiply accumulate results through the same small number of ADCs, but this has an unpleasant impact on system throughput and latency.

Previously, we surveyed the circuit requirements for analog-memory-based accelerators [30]. We described how these requirements resemble the well-known requirements found in conventional memory applications while still differing significantly from these requirements. We qualitatively discussed the fundamental tradeoffs that can influence both the throughput and energy efficiency of such on-chip learning accelerators [30].

In this article, we extend this work by describing a microarchitectural design approach for an analog memory-based implementation of multilayer DNNs that does not use any explicit ADCs. We describe a concept for conveying data in a fully parallel manner between crossbar arrays. While explicit digitization is never performed, the use of digital voltage signaling allows the use of buffers for signal regeneration. In addition, the implementation of the crossbar arrays described here helps finesse the otherwise unpleasant tradeoffs between the electrical length of the row- and column-lines and the logical size of the crossbar array. Execution energy and throughput are quantified using circuit simulations of a full CMOS design in the 90-nm technology node.

The main contributions of this article are as follows:

1) an NVM-based analog accelerator for DNN training, addressing design challenges for forward propagate, reverse propagate, weight update, and weight transfer tasks;
2) design choices for efficient circuitry, including the source follower-based readout, array segmentation, and transmit-by-duration ideas;
3) detailed energy benchmarking for the proposed architecture;
4) projections for further improvements in energy efficiency and throughput per unit area.

The rest of the article is organized as follows. Section 2 introduces the background and related work. Section 3 discusses a microarchitecture for forward and reverse read, and Section 4 addresses weight update and transfer. Section 5 analyzes power/energy benchmarking, and Section 6 discusses how further improvements may be achieved. Section 7 provides concluding remarks.

2 Background and related work

While GPUs have dominated hardware implementations of DNNs in the commercial space, CMOS-based custom hardware accelerators designed specifically for neural algorithms are exceeding GPUs in performance and energy efficiency. Accelerators for inference include the TrueNorth chip from IBM [32], the tensor processing unit (TPU) from Google [33], and the Eyeriss chip from MIT [3]. While training is a considerably harder problem, this is also being addressed in the digital space, with prominent examples including later-generation TPUs, as well as a recently reported digital deep learning processor core [34] that achieves multi-tera operations per second and over 90% sustained utilization across a range of neural network topologies.

To evaluate DNN hardware, two key metrics are often cited:

1) *Energy efficiency:* defined as the number of operations per Joule of energy spent, usually expressed in units of giga- or tera-ops per joule or equivalently giga (tera)-ops per second per watt (GOPs/s/W, TOPs/s/W).
2) *Performance per unit area:* defined as the number of operations achievable per unit area of silicon, usually expressed in units of giga (tera)-ops per second per square millimeter (GOPs/s/mm^2, TOPs/s/mm^2).

For instance, the NVIDIA V100 GPU achieves 100 GOPs/s/W and 37 GOPs/s/mm^2 [35]. The corresponding energy numbers for TPU v1 [33] are 2.3 TOPs/s/W (excluding off-chip data movement) and $\geq$280 GOPs/s/mm^2 (exact die size unavailable).

In comparison to these digital hardware chips, analog or mixed-signal accelerators can potentially achieve orders-of-magnitude improvement for critical DNN computations for both inference and training, especially on fully connected layers. To be competitive against future digital designs, a suggested target for analog systems is 10 fJ/op [31], which corresponds to 100 TOPs/s/W, representing a three-orders-of-magnitude improvement over the V100.

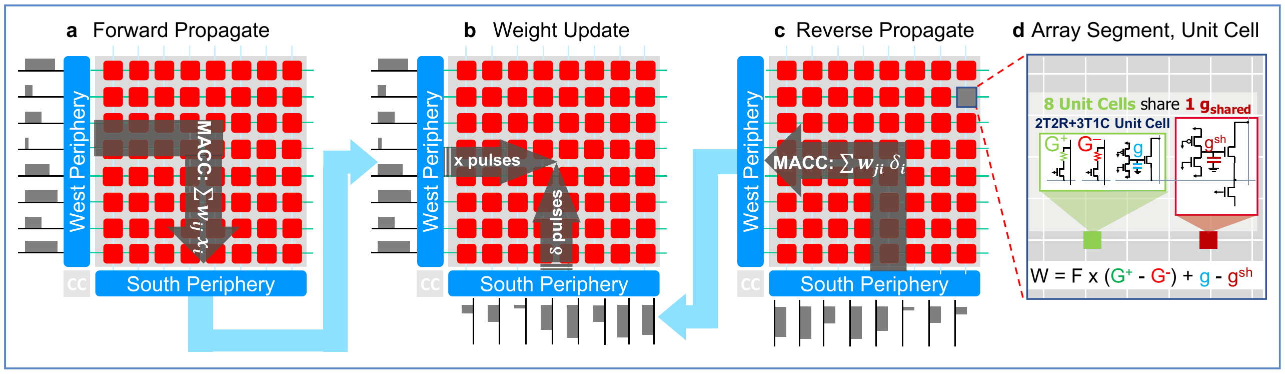

Figure 1

(a) During forward propagate (vector–matrix multiply), activations x_i arrive on the west side of an NVM array and are delivered onto rows. Current integration along columns implements multiply-accumulate (MACC) operations. (b) During weight update (vector outer product), x_i pulses from west overlap with error pulses δ_j from south to program the weight w_{ij}. (c) During reverse propagate (matrix-vector multiply), δ_j pulses arrive on the south side of an NVM array and are delivered onto columns, with current integration along rows implementing transpose MACC operations. Note that while forward, reverse, and weight update are shown side-by-side in this picture, these events are separated in time. (d) Graphical representation of unit cell, with higher significance $G^+$and G^-conductances implemented on NVMs, and lower significance g implemented on a capacitor. Since g allows bidirectional weight updates, area savings can be obtained by using a shared reference $g_{\rm sh}$—in our designs, one g_{sh} is shared across eight unit cells.

In analog systems, the training or inference of DNNs is implemented on dense crossbar arrays of nonvolatile memories. The synaptic weights are encoded in the conductance of NVMs, and the various MACC operations can be performed in NVM crossbar arrays in a massively parallel fashion at the location of the weights, with Ohm's Law providing the multiply operation and Kirchhoff's current law summing the products.

Figures 1(a), **(b)**, and **(c)**, respectively, show schematic representations of how forward, reverse, and weight update tasks of a neural network are mapped to crossbar arrays. As off-chip data movement is not required between layers, such analog memory arrays are not constrained by the von Neumann bottleneck between memory and processing present in conventional digital systems.

The tremendous potential for such analog computing systems in the context of DNN acceleration has been previously studied in [13, 15, 25, 31]. For instance, Marinella et al. [31] compared an analog ReRAM array with digital ReRAM array and SRAM array, demonstrating up to 430$\times$ in energy and 34$\times$ in latency over an SRAM-based accelerator. Similarly, Marinella et al. [13] projected 84 TOPs/s/W, using 4096$\times$4096 arrays. Another full-fledged accelerator, *in situ* analog arithmetic in crossbars [15], was projected to achieve performance per unit area of 479 GOPs/s/mm^2 and power efficiency of 644 GOPs/W on a set of CNN and DNN inference workloads. In all of the above, the dominant energy component comes from the peripheral ADC, which converts the analog voltage on an integration capacitor into a digital value that is then communicated to other arrays. Ideal symmetric and linear NVMs with low per-device conductance are assumed (e.g., tens of nanosiemens in both [31] and [13]). This makes both the energy consumption of the analog array very low (as read currents are small) and the area efficiency extremely good (as simple 1S1R or even 1R memory arrays can be assumed), without the need for tricks such as multiple conductance, capacitive elements, polarity inversion, and other similar techniques [25].

On the other hand, for the non-von Neumann approach using NVM-based synapses, to be practical and viable, NVM-based networks have to produce the same accuracies as the network trained with CPUs or GPUs despite the device nonidealities of NVMs and, at the same time, demonstrate significant advantages in power and/or speed. To this end, in this article, we describe and evaluate an analog system that combines 2T2R+3T1C unit cells—experimentally demonstrated to achieve software-equivalent accuracies with existing PCM technology [25]—together with novel circuit concepts such as transmit-by-duration to avoid the expensive ADC/DAC conversion process. In our evaluation, we fully assess the weight-transfer and polarity inversion operations that help ensure DNN accuracy despite imperfect NVM and incorporate a novel approach for scaling down the inherently large read currents from each unit cell, so as to support large arrays. We review the 2T2R+3T1C unit cell in the next section and then delve into the microarchitectural details and evaluations in the rest of this article.

2.1 Prior work on addressing NVM nonidealities

To overcome the nonidealities of NVM devices and achieve software-equivalent accuracy, we recently proposed a synaptic unit cell with multiple conductances of varying significance (see **Figure 1(d)** [25]). Each weight is the sum of the difference of the least significant pair (LSP) conductances, g and g_{sh}, together with the difference of the most significant pair (MSP), G^+ and G^-, scaled up by a factor $F > 1$, i.e., $W = F(G^+ - G^-) \pm (g - g_{\text{sh}})$. During training, all weight updates are applied to the LSP with MSP unchanged, which requires LSP devices to have linear and symmetric conductance changes [13, 36]. After every few thousand training examples, a weight transfer is conducted to program the entire weight W into the MSP, at which point the LSP is zeroed out (i.e., $g - g_{\text{sh}} = 0$). Since the weight-transfer process is performed only periodically, while the algorithm is busy with computation related to other layers of the neural network, there is sufficient time to perform closed-loop iterative programming on MSP devices to achieve accurate weight tuning.

In our design, G^+ and G^- in MSP utilize nonvolatile PCM devices that can maintain weight values over extended periods of time, while the LSP devices requiring extremely linear and symmetric update characteristics are implemented in CMOS capacitors. Specifically, this volatile conductance (g) circuit module consists of a three-transistor one-capacitor (3T1C) structure, where electric charge can be added or subtracted onto a field-effect-transistor (FET) gate capacitor through a p-type FET or an n-type FET, respectively. The 3T1C module can be programmed with shorter pulses and exhibits better symmetry than PCM, resulting in rapid training with improved characteristics. The 3T1C g device can also be programmed bidirectionally, i.e., the weight can be gradually increased by adding charge to the capacitor through short ON-pulses on the p-FET, or decreased similarly through the n-FET. Therefore, unlike PCM-based conductance pairs, it is unnecessary to independently program a g_{sh} device to tune the weights. Instead, a reference device shared among many unit cells g_{sh} can be used to support negative weight contribution.

The 3T1C capacitor's value is $\sim 11\,\text{fF}$, which allows for 30–100 levels (the exact number depends on the FET variations as well as the applied biases). This is a thick oxide MOSFET capacitor, with a decay constant in the order of hundreds of milliseconds, as the charge needs to be preserved for a full transfer interval (in the order of thousands of examples). The area of the capacitor, including contacts, is 3.05 μm^2, which is 24% of the unit-cell area. The five FETs together occupy another 2 μm^2 with the PCMs above, and the rest of the area is wiring overhead, given the relatively large number of signals per unit cell.

A "polarity inversion" technique—in which the $\pm$ symbol shown in front of the $(g - g_{\text{sh}})$ term is periodically and strategically changed between $+$ and $-$ on alternate transfer intervals—greatly increases the tolerance for fixed device-to-device variability between the p-FET and n-FET [25].

Eventually, our goal is to implement the LSP device pair with suitable analog NVM devices exhibiting optimal properties [36] to achieve high area- and energy efficiency. It has already been shown that the polarity-inversion technique would be able to compensate for significant device-to-device variability in such NVM-based LSP devices [36].

3 Read circuits: Forward and reverse propagate operations

In our design, each neural network layer maps onto one or more identical crossbar arrays, each containing neuron circuits at their edges and interconnected by a flexible routing network. Neuron circuitry at the west (south) edges generates pulses based on the accumulated current from the preceding forward (reverse) pass, driving appropriate voltages in the crossbar array on the basis of the mode of operation and the duration of the neuron pulse [see Figures 1(a), (c)]. The analog signals used during the forward or reverse pass have fixed amplitude but analog duration, which enables standard CMOS buffering approaches to communicate signals from the output of one array to the input of another array, avoiding the analog-to-digital converters that occupy large area and consume considerable power.

During forward propagation [see Figure 1(a)], time-encoded neuron activations, i.e., durations, arriving at the west edge of an array are directed to the unit-cell wordlines, which in turn reach the gates of the access transistors in each unit cell. The NVM elements in the unit cell have their own dedicated access transistors [see **Figure 2(b)**], whereas the access transistor for the 3T1C cell resides in its corresponding shared cell
[see Figure 1(d)]. Eight 3T1C cells along a row share the same g_{sh} element along the same row, with the sources of their read transistors connected to the drain of the common access transistor. Turning the access transistors ON sets up a current proportional to the individual conductances, which flows out of the vertical local bitlines [see Figure 2(b)] and implements the multiply operation. The mechanism is very similar for reverse read, with the distinction that the duration pulses corresponding to error δ's arrive at the south side circuits [see Figure 1(b)], and the direction of flow of current is along horizontal local bitlines [see **Figure 2(d)**]. In this case, all delta pulses are active low.

In order to support negative δ and x quantities, and to correctly multiply these variables against signed weights, a three-phase pulse sequence is used, involving a short "sign pulse" followed by two copies of the same time-encoded duration pulse. During the first phase, only positive deltas are applied to the array from the sending neurons. The receiving neurons are configured to add currents from the positive conductance branches and subtract current from the negative conductance branches. During the second phase, only negative deltas are applied to the array, with positive

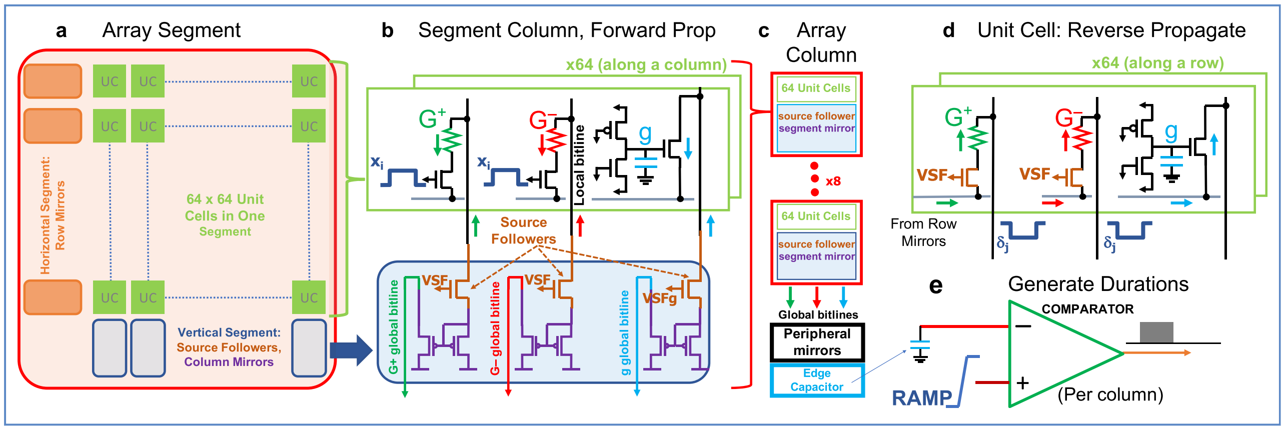

Figure 2

(a) Each array segment contains a patch of 64×64 unit cells, along with horizontal and vertical segment circuitry that helps scale down currents and provides the necessary voltages. (b) During forward propagation, a source-follower device in the vertical segment circuit helps hold the source voltage at a near constant value; pulses arriving on the gates of the unit cell access transistors sink current from the local bitlines. This current is scaled down by segment mirrors that steer current into corresponding global bitlines. (c) Array column consists of eight copies of the segment circuitry, each steering current into the same global bitlines. Before being deposited on the edge integration capacitor, currents (representing the total sum of 512 wx terms) may be scaled again by edge mirrors. (d) Unit-cell conditions for reverse propagate: The role of the source follower is fulfilled by the cell access transistors, and δ error signals are encoded into active low duration pulses along columns. (e) Comparator in each column compares the column capacitor voltage against a common ramp signal emanating from the control circuitry to generate a duration signal that is routed to the next array. Separate capacitors and comparators are used in the south and west circuitry to generate durations for x_i and δ_j, respectively—one of each for each row and column.

conductance branches subtracting and negative conductance branches adding to the capacitor.

3.1 Source follower circuitry

The linear relationship between current and conductance can only be maintained if the voltage across the NVM or read transistor can be pinned to a near-constant value. In the forward direction, since the access transistor is operated as a switch, this effectively means that the local bitline voltage must remain constant irrespective of the current flowing out of it. As opposed to using a more conventional OPAMP-based voltage follower circuit for this task, we employ a simple NFET source-follower device that can be both area- and power-efficient.

A source follower pins the voltage applied to each NVM to some near-constant value based on an input voltage VSF. Similar configurations based on the source follower may be used for both 2T2R and 3T1C structures. In the case of reverse propagate, the access transistors are held at an analog voltage level, fulfilling the role of the source follower and allowing current flow from the respective local bitline nodes [see Figure 2(d)]. In the forward direction, there are dedicated source-follower devices that are responsible for multiple unit cells along a column [see Figure 2(b)]. These reside outside the unit cells in the segment peripheral circuitry (described in the next section). While there can be some change in the source voltage of the source-follower device, which does affect the linearity between current and conductance, the effect is small so long as the current can be kept low (within a few micro-amps), through careful choice of VSF and source-follower device sizing. The effect is also less pronounced in the reverse direction since each conductance effectively has a dedicated source follower (the access FET), unlike in the forward direction.

3.2 Crossbar array segmentation

All other factors being equal, accumulated currents scale linearly with their respective dimension in the crossbar array for both forward inference and backpropagation. Accumulation of large currents is problematic in terms of IR drop and in the sizing of the capacitors in the south-side periphery that must accommodate the cumulative charge.

To address this, array segmentation is used. Each array is divided up into segments, with each segment containing a subset of unit cells alongside additional circuitry [see **Figures 2(a), (c)**]. For instance, in our design, a 512×512 array is divided into 8×8 segments of 64×64 unit cells each. Circuitry at the edge of the segment includes the source follower (in case of forward read), as well as current mirrors that allow for the accumulated current from each segment (say along a “local” bitline) to be mirrored to a lower value to be accumulated with the rescaled currents from the other segments (on a “global” bitline) [see Figure 2(b)].

Note that each column has dedicated source follower and current mirror devices for each of the three conductances G^+, G^-, and g [see Figure 2(b)]. The shared cells have separate mirror and follower devices too. In the case of reverse read, row segment circuitry contains the current mirrors, but no source-follower device. Additionally, transmission gates are used to bypass these mirrors when needed (e.g., row mirrors during forward read or column mirrors during reverse read). The additional segment circuitry exacts a small area cost, yet allows for more graceful scaling of array sizes and mitigation of IR drop, especially in the presence of highly conductive NVM or 3T1C devices.

A typical local bitline in 90-nm technology is expected to have approximately 200 Ω of resistance based on layout. The IR drop even under aggressive current conditions within a local bitline will be around 8 mV. Global bitlines can have $\sim$1.8 kΩ of resistance, but the scaling of currents in the segment current mirrors implies total IR drop would be approximately 33 mV.[1]

The significance factor F between the higher and lower significance conductance pairs can be encoded using a combination of the choice of source-follower voltages and the current mirror scaling ratio. In this article, we use a significance factor of $F = 3$.

3.3 Near-edge periphery

Accumulated currents along global bitlines arriving at the edge of the periphery are scaled through another set of current mirrors [see Figure 2(c)]. Again, there are dedicated mirrors for each one of G^+, G^-, and g for each column or row. All mirrors deposit current onto a per-column or per-row capacitor. Before integration, each capacitor is precharged to an intermediate initial voltage corresponding to "zero" excitation. The final voltage on the capacitor represents the multiply accumulate sum for any particular output neuron. At this point, an activation function may be applied and the data sent forth to the next layer. The array edge capacitor's value is 300 fF. This is a thin-oxide MOSFET capacitor (thick oxide is not needed since the charge needs to be preserved only in the order of microseconds). The area of this capacitor (including contacts) is $\sim 29\,\mu\mathrm{m}^2$.

3.4 Activation encoding and transmission

In a conventional approach, this capacitor voltage would be converted into a multibit digital value using an ADC, followed by conventional digital processing to implement activation functions, derivatives, etc. This method offers flexibility and familiarity but incurs a high area- and energy cost as the optimal throughput solution would require ADC circuitry embedded into each neuron, along with registers and digital processing units [13, 15]. To reduce this area cost and provide more flexibility in the implementation of activation functions, we recently explored the use of lower-resolution ADC circuitry for the neurons in conjunction with a look-up table for a "functional ADC" implementation [37]. However, in all of these cases, one would still need either DACs or pulsewidth modulation (PWM) circuitry at the receiving array to convert arriving multibit information back into either analog voltages or durations.

An alternative approach [38] involves transmitting analog voltages directly to the next array using a constant pulse duration. This could achieve both high-speed transmission and fast multiply-accumulate on the next array. However, there are two significant issues. This approach would: 1) require analog buffering (e.g., using an OPAMP) that is subject to offsets and thus may affect precision, and 2) require that the NVM exhibit a highly linear I–V characteristic over a wide range of input read voltages.

Our approach involves converting the analog voltage accumulated onto each capacitor directly into a digital pulse (which can be buffered using standard CMOS inverters/buffers), where analog information, i.e., the neuron activation, is encoded into the pulse duration. The generated pulse duration is immediately transmitted to the next array, and the integration of the next layer can commence as the pulse is being generated, without any additional latency for intervening storage registers, DACs, or PWM conversion steps.

The circuitry for generating the pulse duration involves the use of a per-neuron comparator [see **Figure 2(e)**]. One input to the comparator is the analog voltage of the edge capacitor, which is unique to every column (or row). The other input is a ramp signal that is common to all the neurons and is generated in shared circuitry that resides in the southwest corner of the array. As the ramp signal goes past the neuron voltage, the output of the comparator flips, creating a pulse duration proportional to the magnitude of the neuron voltage. The use of a ramp and comparator arrangement is quite similar to [31], yet the key difference is the immediate transmission and use of the pulse duration, as opposed to explicit digitization, transmission of that digital data to one or more arrays, and the subsequent remodulation back into pulses of variable duration (or voltages of variable amplitude).

The shape and the start and end points of the ramp determine the activation function being implemented. Two types of activation function are currently supported: a bounded ReLU (bounding is a consequence of fixed dynamic range and is inevitable in all analog and fixed-point digital representations) and piecewise linear approximations of tanh or sigmoid functions (sometimes referred to as "hard" tanh). More precise functions can be implemented if necessary using pulse shaping. Approximate versions of derivative functions needed for reverse propagate can be implemented using more than one ramp, with one or more bits in each row selecting the slope for the particular neuron. A steeper/shallower

[1] The IR drop issue obviously gets more challenging with technology scaling, but given that the unit cells are going to be far larger than minimum size, it may be possible to use wider wires as well as segmentation to keep IR drops within feasible limits.

Table 1 Energy consumption for forward/reverse propagate.

Energy (pJ)	Forward	Reverse
Per-row or per-column components		
Input logic, buffering	1.8	1.9
Current Integration	76	74
Comparator	0.4	0.4
Router	4.2	4.2
Shared components		
Ramp Generation	88	176
Control signals	62	92

ramp slope would lead to a shorter/longer pulse, in effect implementing multiplication with a smaller/larger derivative scale factor. In this article, we assume one "derivative" bit per row, which allows the choice of two different slopes.

3.5 Energy consumption—Forward and reverse read

We estimate the energy consumption for forward and reverse read operations using circuits designed in IBM's 90-nm PDK, together with Spectre circuit simulations. To achieve this, we first simulate "functional primitives" (e.g., a single row or a column), whose contributions can then be scaled to larger arrays and networks without loss of generality. For instance, the forward-per-column primitive calculates the energy for a 512 tall array and includes all the circuitry for one column, i.e., eight segments, each containing 64 unit cells along with source followers, segment mirrors, in addition to the edge peripheral mirrors and the final integration capacitor.

We assume an average NVM conductance of $3\,\mu S$, consistent with our phase-change memory devices. We also assume a significance factor of 3 and an average activation of 38.4 ns out of a maximum duration of 128 ns in each of the two phases for a "signed" duration value, which is somewhat less aggressive than the assumptions we made in [25].

The results are summarized in **Table 1** for both forward and reverse propagate operations. We note that the largest component by far is the current integration corresponding to the MACC operation, which consumes 76 and 74 pJ, respectively, for forward and reverse read operations across 512 unit cells. This energy is strongly dependent on both the conductance assumptions and the average duration of the neuron signal.[2] An energy-efficient comparator ensures that the analog-to-duration conversion energy is significantly lower. Note that this finding is in direct contrast to the results from [13, 31], where analog-to-digital conversion is the dominant factor. One reason for this is that those papers attempt to address the ultimate capabilities achievable with analog accelerators, and therefore assume more ideal NVMs with extremely low average conductance (tens of nanosiemens). The impact of reducing the average conductance in this architecture is discussed in Section 6.

[2] In later sections, where the energy for different neural networks is discussed, we will scale the energy by the average activation and error for that network.

4 Write circuits: Weight update and transfer

In a chip supporting *in situ* training, the conductance/voltage levels of the NVM and capacitor in the unit cell are changed using two mechanisms—weight update and transfer. The former is the actual training step, wherein an outer product of the neuron activation and error vectors is implemented in a crossbar compatible fashion [10]. This occurs in every example (in effect implementing a minibatch size of 1) and is applied on the 3T1C cell, but not the NVM.

Occasional transfer of weight information from the capacitor to the NVM is required in a unit cell with multiple conductances of varying significance. This occurs once every thousands of examples and ensures that weight information is not lost due to the volatility and/or the limited dynamic range of the capacitor. Note, however, that unlike forward propagate, reverse propagate, and weight update, the time and energy spent on transfer is not associated with an equivalent floating-point operation in a conventional accelerator, and thus represents pure overhead.

4.1 Outer product/weight update

During forward/reverse read, incoming durations are saved into capacitors at the west and south edges, respectively. During the weight update step, these voltages are compared against reference voltages generated from the common circuitry at the southwest corner to determine the number of programming pulses that each upstream (west) and downstream (south) neuron should fire. The downstream neuron selects either the Vup or Vdown pulses based on the direction of programming, and the upstream neuron fires pulses on *both* Vx_H and Vx_L nodes, as shown in **Figure 3(a)**. At any crosspoint, the combination of the right upstream and downstream pulses causes the weight to increase or decrease. This operation is necessarily "open-loop" for performance reasons, i.e., programming is done with no verification of whether the right conductance change was achieved on the crosspoints or not. This is because updates may need to happen every example (minibatch of 1) and potentially thousands of devices may need updating.

As small weight updates are desirable for training efficacy, the Vup and Vdown pulses are not rail-to-rail, but instead only turn the PMOS or NMOS programming transistors in the unit cell weakly ON. The Vx pulses, which are rail-to-rail, are turned ON for only a very short time (hundreds of picoseconds in this case). This requires careful buffering of the Vx signals from the southwest corner all

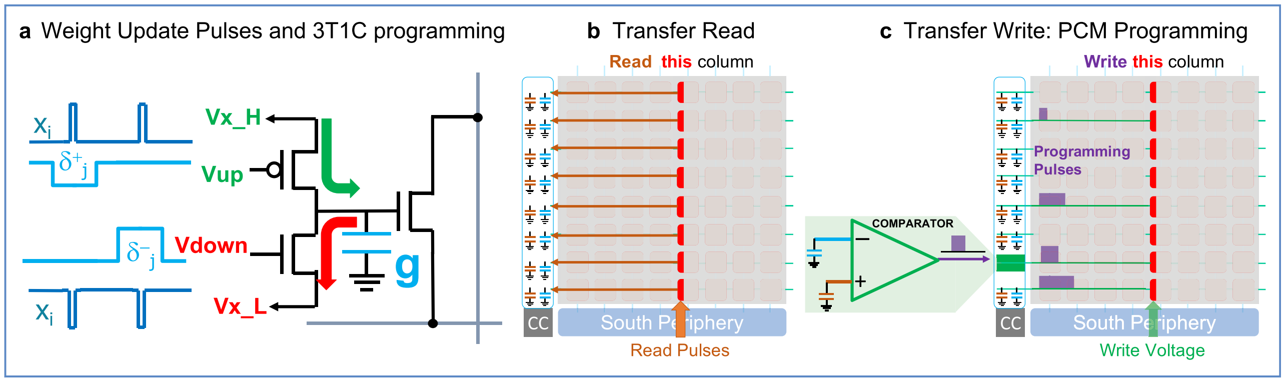

Figure 3

(a) During weight update, x_i programming pulses are applied to the drains of the programming FETs of the 3T1C cell, and a δ_j analog voltage level (Vup or Vdown) is applied to one of the gates. Any overlap of these pulses induces an increase or decrease in the voltage of the capacitor, which corresponds to a change in effective conductance, as sensed through the third read transistor. (b) Transfer operations are carried out on one column at a time, with all rows participating. The dataflow for the read operation is similar to reverse propagate, with active low pulses fired applied from the south-side circuitry, and current accumulation on capacitors on the west side. Two capacitors, one for storing the reference weight information and one for storing the effective weight of the higher significance pair ($F(G^+ - G^-)$), are used. (c) The write operation is driven from the row comparators, using NVM programming pulses that are proportional to the difference between the capacitor voltages.

the way across the array. This pulse generation and buffering energy is included in our estimations.

Furthermore, to avoid synchronization issues between the upstream and downstream pulses, Vup and Vdown are enabled for a much wider time window (several nanoseconds), within which a Vx pulse may appear. The choice of narrow pulses on Vx as opposed to Vup or Vdown is determined by two factors: Simple buffering is possible only on rail-to-rail signals, and Vx pulses go to the sources and drains of the transistors as opposed to the gates, implying less total capacitance to be switched.

The energy for the weight update operation is shown in **Table 2**. The main component of the energy is not the programming of the 3T1C cells themselves, as these require only a small amount of charge to be transferred. Instead, the highest contribution comes from applying the Vx pulses, switching the large wire- and diffusion-capacitances running across the length of the array. The next highest contribution is from saving the incoming x and δ durations onto capacitors in preparation for the weight update.

Table 2 Energy consumption for weight update.

	Energy (pJ)
Per-row or per-column components	
Save Incoming Durations	1
Row fire x programming pulse	2.6
Column δ Vup	0.8
Column δ Vdown	0.5
Shared components	
Short Pulse Generation and Buffering	0.6

4.2 Transfer operation

The transfer process is a multistep closed-loop programming sequence that changes the conductance values of both the NVM and the capacitor (or in a more generic case, the higher- and lower-significance conductance). The goal is to consolidate the weight information entirely onto the higher-significance pair, but in reality there will be some error in programming. In this case, we sacrifice some of the dynamic range of the lower-significance pair to achieve a more accurate representation of the net weight [36].

The approach is closed-loop to accommodate the inter- and intradevice variability of the higher-significance NVM [say phase-change memory (PCM)]. While this implies that the process of transfer is much slower than weight update, we expect there to be little to no performance penalty, as: 1) transfer needs to happen only once every thousands of examples and 2) transfer can occur in one array while another is executing the regular forward/reverse/weight update operation.

Transfer is carried out one column at a time as shown in **Figures 3(b), (c)**. All unit cells in the column participate in the transfer, with per-row circuitry at the edge used for storage and for generating cell-specific programming pulses. The steps for transfer are as follows.

1) The net weight comprising all conductances is read out to a per-row reference capacitor. A scale-up current mirror in the segment circuitry boosts the current such that a discernible change can be obtained on the edge capacitor. Despite this, the read operation has to be longer than the regular

Table 3 Energy consumption for transfer.

	Energy (pJ)
Read Weights or Conductances	18
Comparators	0.6
Write operation row wiring	2
Write operation column wiring	2.5
PCM SET Energy	5.3
PCM RESET Energy	3.8
Post-transfer Tuning row	2.5
Post-transfer tuning column	1.3

reverse read to overcome bitline capacitance and allow for intermediate voltages to stabilize before integration onto the edge capacitor.

2) The lower-significance conductance is zeroed out.
3) The higher-significance components are read out to a second per-row capacitor. The difference between the capacitors represents the change in conductance that needs to be programmed into the higher-significance pair.
4) Using the comparator, a programming pulse is generated to program either the G^+ or G^- conductance. The programming pulse is also part of a feedback loop, which switches a current source ON, adding current to—or subtracting current from—the second capacitor. In effect, this implements a ramp that takes the second capacitor's voltage past the reference capacitor voltage, switching the comparator output and terminating the programming pulse. This results in a programming pulse with a duration proportional to the desired conductance change, suitable for duration-sensitive NVM write processes such as PCM crystallization. Steps 3 and 4 are repeated multiple times, implementing closed-loop programming.
5) The net weight is read out to the second capacitor.
6) Tuning of the lower significance pair is then carried out, applying pulses on the Vx lines. This "post-transfer tuning" process [25] is again closed-loop, repeating steps 5 and 6.

In our simulations, we assume nine read and eight write steps for the closed-loop programming steps described previously. We assume higher-significance phase-change memory elements, with typical RESET voltage, current, and pulsewidth of 2.5 V, 300 μA, and 5 ns, respectively, and typical SET conditions of 2.1 V, 50 μA, and 50 ns, respectively.

The individual energy components are summarized in **Table 3**. While SET and RESET programming events do consume significant amounts of energy, they can be rather infrequent. Energy tends to be dominated by other components such as the multiple read operations (which happen irrespective of whether a particular transfer has already succeeded or not), as well as the post-transfer tuning, which uses short pulses for precision, and hence requires switching the row wire capacitance multiple times.

Table 4 Number of operations/pulses during MNIST training.

Number of ops	724,810
Avg. x activation (ns)	38.4
Avg. δ (ns)	3.8
x pulses per example	2072
δ pulses per example	173
Prorated transfers per example	0.0067
Effective number of SETs per example	464
Effective number of RESETs per example	13
Number of post-transfer tuning pulses	2136

5 Energy estimation for deep networks

Based on the energy numbers of circuit primitives from Tables 1, 2, 3, we estimated energy consumption numbers for training a four-layer perceptron with 528 input neurons, hidden layers of 250 and 125 neurons each, and 10 output neurons (abbreviated 528-250-125-10) on the MNIST dataset. MNIST images are cropped to 22$\times$24 to be consistent with our prior work, most notably [10] and [25].

To extract average x and δ durations, as well as the number of different programming events for weight update and transfer, we used a neural network training simulator that incorporates an experimentally matched PCM "jump-table" [10, 25] and a full CMOS variability model for 3T1C cells [25]. These numbers, extracted over several epochs, are prorated to obtain the equivalent cycle-accurate energy per example. For MNIST, the example triage technique [25] is used to filter out examples on which the neural network is already performing well, preserving the dynamic range of the lower-significance conductance and reducing wall-clock time. **Table 4** shows the scale factors we extracted.

Table 5 shows the energy per example. We note that for MNIST, the energy is dominated by the forward propagate operation ($\sim$60%). This is due to the fact that the first input layer is by far the largest (and does not require reverse propagation), the average x is 10$\times$ the average δ, and programming events are quite infrequent. The total energy per example is 48 nJ, for 66 fJ/op or 15 Tera-ops/s/W.

The total energy per example is in the same order of magnitude, yet different from the number we published in [25]. The reasons are the following: 1) The average activations are different by $>6\times$ (38 ns versus 6 ns), which affects the MACC energy significantly. 2) The device models for the 3T1C cell were different (IBM 90 nm PDK in this article, versus PTM [39] in [25]), and the operating point assumed in this article for the 3T1C cells involves lower voltages/currents for improved energy efficiency. Finally, 3) there was considerable redesign and optimization of the peripheral circuitry, helping maintain

Table 5 Energy consumption for MNIST (nJ).

Forward Propagate	28
Reverse Propagate	3.6
Weight Update	7
Transfer	9
Total Energy	48
Energy Efficiency (Tera-ops/sec/W)	15
Energy/operation (fJ/op)	66

high energy efficiency despite the more realistic and thus much larger average pulse durations assumed here.

In addition to MNIST training, we also estimated the inference energy for a larger LSTM network (200-200-200-10,000) that implements next word prediction for the Penn Tree Bank [40] dataset. From our neural network simulations, we estimated the average activations of the x and h vectors to be 27 ns out of a maximum duration of 128 ns. Results are summarized in **Table 6**. We find that the network requires 378 nJ per example (dominated by array MACCs at 73%, followed by routing at 20%), achieving a net efficiency of 14 Tera-ops/s/W, which is quite comparable to the MNIST result.

The energy efficiency numbers presented in this section ($\sim$15 Tera-ops/s/W for the various networks) compare quite favorably with present-day digital accelerators including the NVIDIA V100 (0.1 Tera-ops/s/W [35]) or the Google TPU generation 1 (2.3 Tera-ops/s/W for inference only [33], but excluding any data movement into or out of the ASIC).[3] However, there are several important caveats including the following.

1) Regarding the analog nature of the computation, while with digital accelerators it is possible to guarantee precision in individual computations, in systems with several cascaded analog circuit components, this is very difficult to quantify through simulation alone. The development and evaluation of small-scale prototypes is thus of paramount importance.
2) NVM devices still suffer from significant nonidealities. While weight update symmetry and linearity can be improved with ideas such as the 3T1C cell, this remains to be proven in an end-to-end hardware implementation. Additionally, NVM read noise is also an important consideration.
3) While these numbers also omit data movement onto and off the chip, such data would be limited to input examples and output classifications only. Thus for any suitably large DNN, this missing energy contribution can be expected to be relatively small.
4) The system as described here does not offer the full range of flexibility in terms of arbitrary mathematical functions that one could achieve with a digital accelerator. Therefore, initial designs may need to investigate hybrid systems that combine analog and digital cores.
5) Convolution layers, which are the workhorses of deep learning today, are more challenging to implement on crossbar arrays, especially for training. While it may be possible for an analog accelerator to be energy-efficient by using various data orchestration schemes, it is not likely to be efficient in terms of silicon area $(\mathrm{TOPs/s/mm^2})$. On the other hand, one important reason for the widespread use of Conv Nets is their suitability to existing digital hardware (compute-bound as opposed to memory-bound), and the availability of new hardware for MLP and LSTMs can be expected to also drive algorithm development.

[3] Numbers for newer TPU generations are not known at the time of writing. However, it is possible that these may be less energy-efficient, given the need to support higher precision as well as training operations.

Table 6 Inference on Penn Tree bank.

Average activation	20ns
Number of Multiply Accumulates	5303200
Total Number of Ops	5304400
Energy MACC (nJ)	275
Router Energy (nJ)	74
Total Energy (nJ)	378
Energy efficiency (Tera-ops/sec/W)	14
Energy per operation (fJ/op)	71

6 Discussion

The previous set of numbers quantifies the energy benefits of our microarchitectural approach for specific deep networks and for our specific hardware implementation with 2T2R+3T1C cells. In this section, we explore pathways for further improvements in energy efficiency and performance. In these analyses, we will assume a 4T4R unit cell, with the weight expressed as $W = F \times (G^{+} - G^{-}) \pm g^{+} - g^{-}$. We assume that both pairs of NVMs operate in the same conductance range, and that the lower significance pair is capable of linear and symmetric conductance response.

Additionally, as opposed to any one network, we assume a generic 512$\times$512 crossbar array. We do consider the routing energy to bring pulse durations to the west or south side of the array from a neighboring array, meaning that all necessary intermediate steps are fully covered. (As described previously, we neglect only the on- and off-chip data transport into and out of the first and last layers of the DNN.) We also assume that during weight update operations, roughly 31% of x_i's and 31% of δ's are active, leading to 10% of weights being updated. We assume a programming voltage of 2.5 V, a programming time of 10 ns, and a 20% overhead for the transfer operation, consistent with what we observed in the training of the MNIST network in the earlier section.

6.1 Effect of conductance scaling

From the earlier results, the largest contribution to energy comes from the raw analog currents flowing in the array

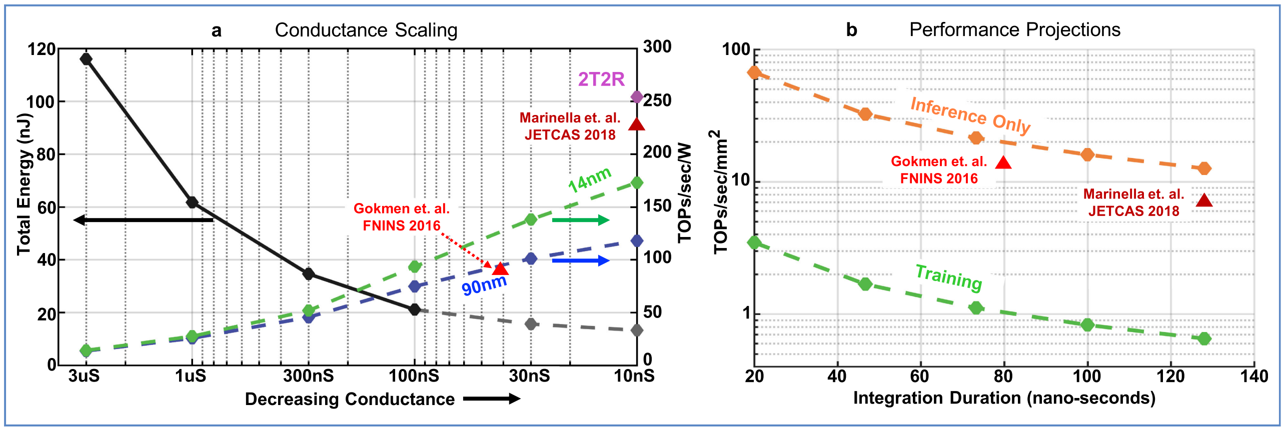

Figure 4

(a) Impact of conductance scaling on total energy for forward, reverse, update, and transfer, and the corresponding energy efficiencies. The scaling coefficient between the black (nJ/training example) and blue lines (TOPs/s/W) is (513 × 512 × 2 OP/Forward-MACC + 512 × 512 × 2 OP/Reverse-MACC + 513 × 512 × 2 OP/Update-MACC) = 1 574 912 OPs/array. (b) Projection of performance per unit area as a function of the integration duration. Training assumes 10% array utilization, and inference assumes 50% array utilization, of an array with 512×512 weights.

during forward and reverse read operations, which is consistent with the MACC operation. The development of more resistive NVMs could lead to significant reductions in energy consumption, as we demonstrate in **Figure 4(a)**. We explore the scaling of device conductance from 3 μS down to 10 nS and plot the total energy for forward, reverse, update, and transfer, as well as the resulting energy efficiency on the two y-axes. In this case, both the G^+ and g^+ conductances for all the devices in the array were assumed to be at the values on the x-axis, and the G^- and g^- values were assumed to be 30× smaller, consistent with the ON/OFF conductance ratio of the PCM devices in the earlier section. (Negative weights of the same weight magnitude would of course exhibit a large G^- with other conductances being small, thus incurring an identical read-energy impact.)

The four steps from 3 μS to 100 nS were simulated in Spectre, with progressive reduction in current mirror ratios to ensure the same final signal on the edge integration capacitor. We then fitted the total energy to project to the even lower conductance values of 30 and 10 nS. We also included a second trend line for technology scaling from 90 to 14 nm—here we assume that the energy for digital routing of durations between arrays can be reduced with capacitance and voltage scaling (1.2 to 0.8 V), while all other energy components are left unchanged.

We find that a 1.9×–1.2× reduction in energy can be obtained from step to step [black line in Figure 4(a)], with diminishing returns as the relative impact of the other energy components increases. The design could achieve up to 75 TOPs/s/W at 100 nS average conductance and 118 TOPs/s/W at 10 nS for the case with no technology scaling. The corresponding numbers for the scaled version are 93 TOPs/s/W at 100 nS and 172 TOPs/s/W at 10 nS. This analysis neglects the effects and potential implications of noise, which is beyond the scope of this article, but may become a critical concern at such low conductances.

For reference, we also show other datapoints from the literature on the same plot and note that they are consistent with our own numbers. However, we caution that there are considerable differences in the underlying assumptions; therefore, one must be careful when making direct simplistic comparisons. For instance, Marinella et al. [31] achieved 224 TOPs/s/W at 14 nm and 10 nS conductance, which at first glance is larger than our numbers. However, in addition to differing assumptions on programming voltages, pulsewidths, average activation duration, and number of weight updates per cycle, the most significant difference arises from the choice of unit cell. Our design is a 4T4R cell that both increases dynamic range and simplifies requirements on the individual device pairs in order to accommodate realistic NVM devices [25, 36]. In contrast, Marinella et al. [31] used a 1S1R design, which requires near-ideal device pairs, namely: 1) an NVM that can offer bidirectional linear and highly symmetric programming in addition to high endurance and retention, as well as 2) a bidirectional symmetric selector that must not exhibit any device-to-device variability in its IV characteristics within the exponential turn-ON region. The inclusion of such idealized devices in the analysis of our microarchitectural approach would similarly inflate our energy efficiency predictions as well. This then provides a strong incentive for continued research into compact yet highly capable NVM and selector devices.

We estimated the energy for a 2T2R version of our design. In this case, we did not reduce any parasitic capacitances (though this would be expected, as the array size would roughly halve). We did halve the contribution of the current integration—which was the dominant factor at 3 μS conductance, but has only a small impact when conductances are already very low. The biggest impact comes from the elimination of the transfer operation, which is now superfluous. Under these assumptions, our design could achieve 252 TOPs/s/W. Again, we emphasize that all of the caveats mentioned in the earlier section would still apply.

Based on these results, we note that a 100-nS NVM device range, while being a more achievable target for device engineering, could still achieve significant benefits at the system level. IR drop issues can be addressed by a combination of wider wires (since its likely access FETs are going to be needed anyway, given the even more stringent requirements for a selector in a 1S1R $4F^2$ unit cell), together with the array segmentation approach outlined previously.

6.2 Performance analysis

We estimated the effective performance per unit area (TOPs/s/mm^2) for both inference and training. We used full custom layouts of training arrays (2T2R+3T1C cells + peripheral circuitry) and inference arrays (4T4R cells + peripheral circuitry and routing). The total area for the former was 5.8 mm^2, and for the latter was 1.5 mm^2 at 90 nm. We quadratically scaled these areas from 90 to 14 nm using the M1 metal dimension (140 versus 32 nm). For training, we assumed an average utilization of 10% (array active once every ten steps), and for inference we assumed an average utilization of 50% [fully pipelined structure with odd (even) arrays computing (transmitting) in step 1 (2), and transmitting (computing) in step 2 (1)]. We assume 10-ns setup time for layers, with the maximum single pulse duration varying from 20 to 128 ns. This corresponds to total array compute time varying from 50 to 266 ns because of the two phases for “signed” durations.

Results for both inference and training are shown in Figure 4(b). The *x*-axis is the maximum single phase duration. We find that we can achieve 652 GOPs/s/mm^2 under 128-ns max duration pulses, and up to 3.5 TOPs/s/mm^2 under much shorter durations. This latter number is consistent with numbers reported in our previous publication [25]. We also find even better inference numbers, owing to the higher utilization and lower area footprint—with 12 TOPs/s/mm^2 at 128 ns and 67 TOPs/s/mm^2 at 20 ns max duration. We show the other reference points from literature (again assuming 50% utilization for inference). The significant difference for training is primarily due to the area requirements for the unit cell (2T2R+3T1C versus 1R arrays for [13, 31]).

While these projected numbers are significantly better than today’s systems, we must note that smaller durations could effectively reduce compute precision, which could impact accuracy of the neural network. While the high degree of utilization for inference is straightforward to implement for fully connected layers, LSTM and CNN orchestration challenges can significantly diminish these numbers for such recurrent and convolutional networks. While we expect the effect to be more acute for CNNs than for LSTMs, effective mapping strategies are still under investigation and will form a significant part of our future work.

7 Conclusion

We have presented a novel microarchitectural approach for analog acceleration of DNNs based on crossbar arrays of NVM devices. Through detailed simulations of circuit primitives, we quantified the energy consumption for a full 90-nm design, projecting $\sim$14 TOPs/s/W on fully connected and LSTM networks. We then explored avenues for further improvements in the energy and performance, using similar simulation studies on generic 512$\times$512 crossbar arrays. We showed that conductance scaling down to 100 nS, in conjunction with technology scaling to 14 nm, could achieve up to 93 TOPs/s/W. Performance/unit area estimations based on full custom layouts show up to 3.5 TOPs/s/mm^2 for training and up to 67 TOPs/s/mm^2 for inference, far exceeding the capabilities of present-day systems. While this work highlights the tremendous potential for analog computing, we caution that there are still several challenges, especially in demonstrating that imperfect NVMs and analog circuitry can achieve the same high accuracy as reliable digital hardware.

References

1. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
2. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by backpropagating errors,” *Nature*, vol. 323, pp. 533–536, 1986.
3. V. Sze, Y. H. Chen, T. J. Yang, et al., “Efficient processing of deep neural networks: A tutorial and survey,” *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.
4. H. Tsai, S. Ambrogio, P. Narayanan, et al., “Recent progress in analog memory-based accelerators for deep learning,” *J. Phys. D, Appl. Phys.*, vol. 51, no. 28, 2018, Art. no. 283001.
5. G. W. Burr, M. BrightSky, A. Sebastian, et al., “Recent progress in phase-change memory technology,” *IEEE J. Emerg. Sel. Topics Circ. Syst.*, vol. 6, no. 2, pp. 146–162, 2016.
6. D. Ielmini and H. S. P. Wong, “In-memory computing with resistive switching devices,” *Nature Electron.*, vol. 1, no. 6, pp. 333–343, 2018.
7. M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, et al., “Training and operation of an integrated neuromorphic network based on metal-oxide memristors,” *Nature*, vol. 521, pp. 61–64, 2015. [Online]. Available: https://doi.org/10.1038/nature14441
8. P. M. Sheridan, F. Cai, C. Du, et al., “Sparse coding with memristor networks,” *Nature Nanotechnol.*, vol. 12, pp. 784–789, 2017. [Online]. Available: https://doi.org/10.1038/nnano.2017.83
9. C. Li, D. Belkin, Y. Li, et al., “Efficient and self-adaptive in-situ learning in multilayer memristor neural networks,” *Nature Commun.*, vol. 9, no. 1, 2018, Art. no. 2385. [Online]. Available: https://doi.org/10.1038/s41467-018-04484-2
10. G. W. Burr, R. M. Shelby, C. di Nolfo, et al., “Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element,” in *Proc. IEEE Int. Electron Devices Meeting*, 2014, pp. 29.5.1–29.5.4.

11. G. W. Burr, R. M. Shelby, S. Sidler, et al., “Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses) using phase-change memory as the synaptic weight element,” *IEEE Trans. Electron Devices*, vol. 62, no. 11, pp. 3498–3507, Nov. 2015.
12. G. W. Burr, P. Narayanan, R. M. Shelby, et al., “Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power),” in *Proc. IEEE Int. Electron Devices Meeting*, 2015, pp. 4.4.1–4.4.4.
13. T. Gokmen and Y. Vlasov, “Acceleration of deep neural network training with resistive cross-point devices: Design considerations,” *Front. Neurosci.*, vol. 10, 2016, Art. no. 333.
14. S. R. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, “Mixed-precision architecture based on computational memory for training deep neural networks,” in *Proc IEEE Int. Symp. Circuits Syst.*, 2018, pp. 1–5
15. A. Shafiee, A. Nag, N. Muralimanohar, et al., “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit.*, 2016, pp. 14–26.
16. M. Hu, J. P. Strachan, Z. Li, et al., “Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication,” in *Proc. 53rd ACM/EDAC/IEEE Des. Autom. Conf.*, 2016, pp. 1–6.
17. S. Agarwal, S. J. Plimpton, D. R. Hughart, et al., “Resistive memory device requirements for a neural algorithm accelerator,” in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 929–938.
18. S. Yu, P. Y. Chen, Y. Cao, et al., “Scaling-up resistive synaptic arrays for neuro-inspired architecture: Challenges and prospect,” *IEDM Tech. Dig.*, pp. 17.3.1–17.3.4, 2015.
19. Y. van de Burgt, E. Lubberman, E. J. Fuller, et al., “A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing,” *Nature Mater.*, vol. 16, no. 4, pp. 414–418, 2017.
20. E. J. Fuller, F. El Gabaly, F. Léonard, et al., “Li-ion synaptic transistor for low power analog computing,” *Adv. Mater.*, vol. 29, no. 4, 2017, Art. no. 1604310.
21. J. Tang, D. Bishop, S. Kim, et al., “Ecram as scalable synaptic cell for high-speed, low-power neuromorphic computing,” in *Proc. IEEE Int. Electron Devices Meeting*, 2018, pp. 13.1.1–13.1.4.
22. I. Giannopoulos, A. Sebastian, M. Le Gallo, et al., “8-bit precision in-memory multiplication with projected phase-change memory,” in *Proc. IEEE Int. Electron Devices Meeting*, 2018, pp. 27.7.1–27.7.4.
23. F. Merrikh-Bayat, X. Guo, M. Klachko, et al., “High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays,” *IEEE Trans. Neur. Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4782–4790, Oct. 2018.
24. S. Kim, T. Gokmen, H. M. Lee, et al., “Analog CMOS-based resistive processing unit for deep neural network training,” in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst.*, 2017, pp. 422–425.
25. S. Ambrogio, P. Narayanan, H. Tsai, et al., “Equivalent-accuracy neuromorphic hardware acceleration of neural network training using analog memory,” *Nature*, vol. 558, no. 7708, pp. 60–67, 2018.
26. Y. Li, S. Kim, X. Sun, et al., “Capacitor-based cross-point array for analog neural network with record symmetry and linearity,” in *Proc. IEEE Symp. VLSI Technol.*, 2018, pp. 25–26.
27. I. Boybat, C. di Nolfo, S. Ambrogio, et al., “Improved deep neural network hardware-accelerators based on non-volatile-memory: The local gains technique,” in *Proc. IEEE Int. Conf. Rebooting Comput.*, Nov. 2017, pp. 1–8.
28. I. Boybat, M. Le Gallo, S. R. Nandakumar, et al., “Neuromorphic computing with multi-memristive synapses,” *Nature Commun.*, vol. 9, no. 1, 2018, Art. no. 2514.
29. S. Agarwal, R. B. Jacobs-Gedrim, A. H. Hsia, et al., “Achieving ideal accuracies in analog neuromorphic computing using periodic carry,” in *Proc. Symp. VLSI Technol.*, 2017, pp. T174–T175.
30. P. Narayanan, A. Fumarola, L. Sanches, et al., “Towards on-chip acceleration of the backpropagation algorithm using non-volatile memory,” *IBM J. Res. Dev.*, vol. 61, no. 4/5, pp. 11:1–11:11, 2017.
31. M. J. Marinella, S. Agarwal, A. Hsia, et al., “Multiscale co-design analysis of energy, latency, area, and accuracy of a ReRAM analog neural training accelerator,” *IEEE J. Emerg. Sel. Topics Circ. Syst.*, vol. 8, no. 1, pp. 86–101, Mar. 2018.
32. P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, et al., “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014. [Online]. Available: http://science.sciencemag.org/content/345/6197/668
33. N. P. Jouppi, C. Young, N. Patil, et al., “In-datacenter performance analysis of a tensor processing unit,” 2017, *arXiv:1704.04760*.
34. B. Fleischer, S. Shukla, M. Ziegler, et al., “A scalable multi-TeraOPS deep learning processor core for AI training and inference,” in *Proc. IEEE Symp. VLSI Circuits*, 2018, pp. 35–36.
35. “Nvidia volta Gv100 12nm finfet GPU detailed,” 2017. [Online]. Available: wccftech.com/nvidia-volta-gv100-gpu-tesla-v100-architecture-specifications-deep-dive/
36. G. Cristiano, M. Giordano, S. Ambrogio, et al., “Perspective on training fully connected networks with resistive memories: Device requirements for multiple conductances of varying significance,” *J. Appl. Phys.*, vol. 124, no. 15, 2018, Art. no. 151901.
37. M. Giordano, G. Cristiano, K. Ishibashi, et al., “Analog-to-digital conversion with reconfigurable function mapping for neural networks activation function acceleration,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 367–376, Jun. 2019.
38. X. Liu, M. Mao, B. Liu, et al., “Reno: A high-efficient reconfigurable neuromorphic computing accelerator design,” in *Proc. 52nd ACM/EDAC/IEEE Des. Autom. Conf.*, 2015, pp. 1–6.
39. Y. Cao, “Predictive Technology Model (PTM),” 2012. [Online]. Available: ptm.asu.edu.
40. “Penn Tree Bank (PTB),” 2015. [Online]. Available: https://github.com/tomsercu/lstm/tree/master/data

Received December 20, 2018; accepted for publication July 3, 2019

Hung-Yang Chang *IBM Research, San Jose CA 95120 USA (hyc4@illinois.edu)*. Mr. Hung-Yang received a bachelor’s degree in 2018 from National Tsing Hua University (NTHU), Hsinchu, Taiwan. He is currently working towards a Ph.D. degree in electrical and computer engineering at the University of Illinois at Urbana-Champaign (UIUC), Urbana, IL, USA. From October 2017 to April 2019, he was a Research Assistant at National Chiao Tung University (NCTU), Hsinchu, and attended the joint student program between NCTU and IBM Research – Almaden, San Jose, CA, USA, from October 2018 to April 2019. In IBM, he worked on AI hardware micro-architectures for low energy at high speed. His current research interests lie at the intersection of machine learning on resource-constrained platforms, circuit design, and architecture for machine learning.

Pritish Narayanan *IBM Research, San Jose, CA 95120 USA (pnaraya@us.ibm.com)*. Dr. Narayanan received a Ph.D. degree in electrical and computer engineering from the University of Massachusetts Amherst, Amherst, MA, USA, in 2013. He joined IBM Research as a Research Staff Member. His current research interests include hardware systems for machine learning and cognitive computing.

Scott C. Lewis *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (sclewis@us.ibm.com)*. Mr. Lewis graduated from MIT in 1968 and joined the IBM team designing a 128-bit bipolar SRAM chip. In 1973, as Lead Designer, he began work on a 36-kb DRAM, which in 1975, was qualified and became IBM’s first production DRAM. From then until retirement in 2005, he worked for IBM, usually as Technical Leader, on subsequent DRAM generations up through 256-Mb. In 2008, he rejoined IBM to work on phase change memory.

Nathan C. P. Farinha *IBM Research, San Jose, CA 95120 USA (nathan.farinha@usp.br)*. Mr. Farinha is currently pursuing an M.S. degree in computer engineering from the University of Sao Paulo, Brazil, for 2019. His research area is in computer networks, focused on content delivery networks. He spent one year in the Science and Technology department at IBM Research – Almaden, San Jose, CA, USA, working on hardware accelerators. He is also an active member in data science worldwide competitions, such as Kaggle.

Kohji Hosokawa *IBM Research, Shin-Kawasaki, Kanagawa 2120032, Japan (khosoka@jp.ibm.com)*. Mr. Hosokawa is a Senior Technical Staff Member in the Science and Technology department at IBM Research – Tokyo.

Charles Mackin *IBM Research, San Jose, CA 95120 USA (charles.mackin@ibm.com)*. Dr. Mackin is currently a Research Staff Member with IBM Research – Almaden, San Jose, CA, USA. He received a Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2018. His current research interests include sensors and hardware accelerators for machine learning.

Hsinyu Tsai *IBM Research, San Jose, CA 95120 USA (htsai@us.ibm.com)*. Dr. Tsai received a Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2011. She joined the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, in 2011 as a Research Staff Member. Her current research interests include analog memory for machine learning and cognitive computing.

Stefano Ambrogio *IBM Research, San Jose, CA 95120 USA (stefano.ambrogio@ibm.com)*. Dr. Ambrogio received a Ph.D. degree in 2016 from Politecnico di Milano, Milan, Italy, studying the reliability of resistive memories and their application on neuromorphic networks. He is now a Research Staff Member at IBM Research – Almaden, San Jose, CA, USA, in the Neuromorphic Devices and Architectures Team, working on hardware accelerators based on non-volatile memories for neural networks.

An Chen *IBM Research, San Jose, CA 95120 USA (chenan@us.ibm.com)*. Dr. Chen received a Ph.D. degree in electrical engineering from Yale University, New Haven, CT, USA, in 2004. His research interests include nonvolatile memories, beyond-CMOS devices, and novel computing. He is currently managing NRI and nCORE programs at SRC as an IBM assignee. He is a senior member of the IEEE.

Geoffrey W. Burr *IBM Research, San Jose, CA 95120 USA (gwburr@us.ibm.com)*. Dr. Burr received a Ph.D. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 1996. He joined IBM Research – Almaden, San Jose, CA, USA, in 1996, where he is currently a Distinguished Research Staff Member. His current research interests include nonvolatile memory and cognitive computing. He is a senior member of the IEEE.