Fully On-Chip MAC at 14 nm Enabled by Accurate Row-Wise Programming of PCM-Based Weights and Parallel Vector-Transport in Duration-Format

P. Narayanan, *Senior Member, IEEE*, S. Ambrogio, *Member, IEEE*, A. Okazaki, *Member, IEEE*, K. Hosokawa, H. Tsai, *Senior Member, IEEE*, A. Nomura, T. Yasuda, C. Mackin, S. C. Lewis, A. Friz, M. Ishii, *Member, IEEE*, Y. Kohda, H. Mori, K. Spoon, R. Khaddam-Aljameh, *Member, IEEE*, N. Saulnier, M. Bergendahl, *Member, IEEE*, J. Demarest, K. W. Brew, V. Chan, S. Choi, I. Ok, I. Ahsan, F. L. Lie, *Member, IEEE*, W. Haensch, V. Narayanan, *Senior Member, IEEE*, and G. W. Burr, *Fellow, IEEE*

***Abstract*— Hardware acceleration of deep learning using analog non-volatile memory (NVM) requires large arrays with high device yield, high accuracy Multiply-ACcumulate (MAC) operations, and routing frameworks for implementing arbitrary deep neural network (DNN) topologies. In this article, we present a 14-nm test-chip for Analog AI inference—it contains multiple arrays of phase change memory (PCM)-devices, each array capable of storing 512 × 512 unique DNN weights and executing massively parallel MAC operations at the location of the data. DNN excitations are transported across the chip using a duration representation on a parallel and reconfigurable 2-D mesh. To accurately transfer inference models to the chip, we describe a closed-loop tuning (CLT) algorithm that programs the four PCM conductances in each weight, achieving <3% average weight-error. A row-wise programming scheme and associated circuitry allow us to execute CLT on up to 512 weights concurrently. We show that the test chip can achieve near-software-equivalent accuracy on two different DNNs. We demonstrate tile-to-tile transport with a fully-on-chip two-layer network for MNIST (accuracy degradation ∼0.6%) and show resilience to error propagation across long sequences (up to 10 000 characters) with a recurrent long short-term memory (LSTM) network, implementing off-chip activation and vector-vector operations to generate recurrent inputs used in the next on-chip MAC.**

***Index Terms*— Analog accelerator, analog AI, analog multiply-accumulate (MAC) for deep neural networks (DNNs), deep learning accelerator, inference, in-memory computing, non-volatile memory (NVM), phase-change memory (PCM).**

Manuscript received August 2, 2021; accepted September 13, 2021. Date of publication October 11, 2021; date of current version December 1, 2021. This work was supported by IBM Research AI Hardware Center. The review of this article was arranged by Editor P. Grudowski. *(Corresponding author: P. Narayanan.)*

P. Narayanan, S. Ambrogio, H. Tsai, C. Mackin, A. Friz, K. Spoon, and G. W. Burr are with IBM Research–Almaden, San Jose, CA 95120 USA (e-mail: pnaraya@us.ibm.com).

A. Okazaki, K. Hosokawa, A. Nomura, T. Yasuda, M. Ishii, Y. Kohda, and H. Mori are with IBM Research, Tokyo 212-0032, Japan.

S. C. Lewis, W. Haensch, and V. Narayanan are with IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA.

R. Khaddam-Aljameh is with IBM Research Zurich, 8803 Rüschlikon, Switzerland.

N. Saulnier, M. Bergendahl, J. Demarest, K. W. Brew, V. Chan, S. Choi, I. Ok, I. Ahsan, and F. L. Lie are with IBM Albany NanoTech, Albany, NY 12203 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TED.2021.3115993.

Digital Object Identifier 10.1109/TED.2021.3115993

I. Introduction

DEEP learning algorithms have revolutionized several machine learning disciplines over the last decade, notably in the areas of image recognition [1], speech recognition [2], language translation [3], etc. This success has been driven by the availability of copious amounts of real-world data together with the compute power of graphical processing units (GPUs), which have made it possible to train and deploy larger and larger deep neural network (DNN) models that achieve near- or better-than-human accuracy on several enterprise-relevant tasks.

With the trend that larger models perform better, memory and computational requirements on hardware have also been steadily expanding. For instance, Amodei and Hernandez [4] shows an exponential increase in training energy just over the last few years. These exponential trends have created an opportunity for new hardware accelerators since high initial investments can be justified given the widespread applicability. As an example, several ASICs and FPGA solutions based on existing digital scaled CMOS technologies have been proposed and/or manufactured in the last few years [5]–[9].

There is also a longer term opportunity potentially using new technologies and re-thinking multiple layers of the

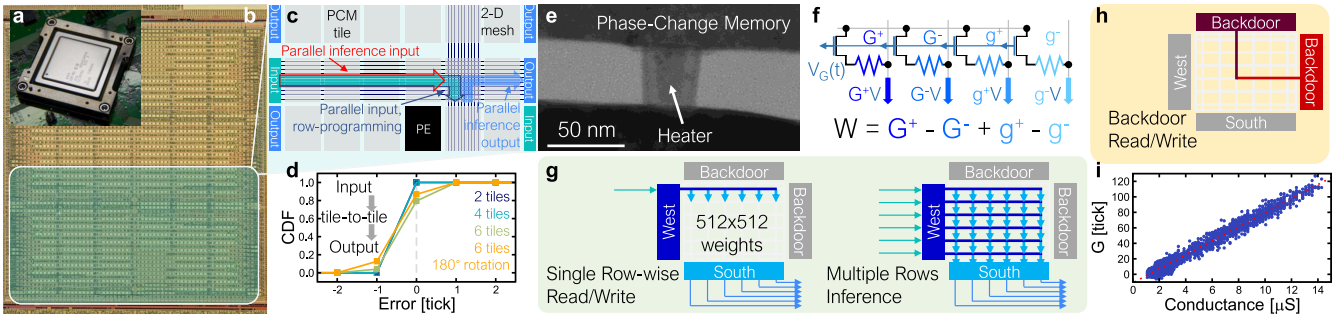

Fig. 1. (a) and (b) IBM's 14-nm inference chip and micrograph. (c) Functional blocks include input LP, output LP, PCM tiles. Duration transport across tiles using a 2-D parallel-signal mesh is also shown. (d) CDFs of bit-errors show highly accurate across-chip duration transport for travel distances up to six tiles. (e) Transmission Electron Microscope image of a PCM device integrated in 14-nm back end. (f) Each DNN weight is encoded using four PCM devices. (g) Row-wise read/write (left) during programming uses the same circuit paths (and associated non-idealities) experienced during full inference (right). (h) Single-device ("backdoor") sense-amp read circuitry can measure device conductance in μS. (i) Measurements using modes described in (g) and (h) are well-correlated.

design hierarchy–including new devices, circuits, architectures, and algorithms—that has received considerable interest in the last several years. This involves using either arrays of capacitors [10] or resistive non-volatile memory (NVM) [11]–[18] for accelerating Multiply-ACcumulate (MAC) operations, which account for the vast majority of computations in several DNNs (see [9]). In NVM-based accelerators, weights are implemented in the conductance value G of analog resistive elements, and excitations are implemented using some form of voltage or time-encoding $[V(t)]$. Ohm's law ($V(t) = I(t) \times R$, rewritten as $I(t) = V(t) \times G$) accomplishes multiplication, and Kirchhoff's current law accomplishes accumulation. This is an example of a non-Von Neumann computing architecture where compute is performed at the location of data. In addition to being non-volatile, such NVM arrays can achieve high density and implement fast and massively parallel computations. Importantly, they also eliminate the expensive data transfer between processing units and off-chip memory that is a source of considerable energy consumption in conventional digital systems [19].

However, building large-scale systems based on analog memory poses several challenges. To run useful workloads, we need large NVM arrays with high yield at competitive technology nodes, which are not usually part of most foundry processes. We also need an on-chip high bandwidth routing framework to map complex networks. Furthermore, this routing needs to be reconfigurable to allow mapping of different neural network topologies to the same hardware. We need to achieve iso-accuracy, meaning that the analog computation with imperfect and noisy analog devices must achieve the same accuracies as software on neural network tasks. For inference, this often means we need closed loop tuning (CLT) strategies to program the weights of the network accurately. However, in order to be able to load and run real-world models in the hardware, we also need this closed loop process to be scalable or parallelizable with minimum overhead. And finally, we need accurate and efficient neuron circuitry that should be able to accurately convert the raw current coming out of the array into MAC values and implement other computational primitives of the neural network including activation functions, normalization, and vector–vector operations.

The following sections address several of these challenges through a series of experimental demonstrations carried out on a 14-nm all-analog inference chip. Section II describes the architecture and key features of the chip. Section III presents details on device programming, including the CLT strategy and circuit details. Section IV presents programming results. Section V describes inference experiments and results. Section VI concludes this article.

II. Chip Overview

The packaged 14-nm all-analog inference chip is shown in Fig. 1(a), along with a chip micrograph in Fig. 1(b). The chip consists of phase change memory (PCM) tiles [see Fig. 1(c)], with input and output landing pad blocks (ILP, OLP). A 2-D routing mesh is used for input-to-tile, tile-to-tile, and tile-to-output data transmission. The data is transported in duration-format, with pulse widths representing excitation values, and standard digital re-buffering is used for maintaining pulse integrity. Pulsewidth modulators and duration-to-byte converters at the chip edge allow conversion between duration and digital bits at the ILPs and OLPs. Controllers on each tile configure the transmit and receive direction of each local mesh, allowing arbitrary routing topologies. Fig. 1(d) shows the error cumulative distribution for durations at propagation lengths of up to six tiles, demonstrating highly accurate duration transmission ($<\pm 1$ tick error, where 1 tick is $\sim$1.2 ns) across the chip.

Another key feature is that the chip does not have analog-to-digital converters (ADCs) for conversion of MAC outputs into the digital domain at each tile. Instead, to save area and energy, analog voltages are directly transformed into durations using a simple ramp and comparator scheme, and these durations are transferred on to the 2-D mesh. This is described in more detail in a later section.

Each tile stores 512 $\times$ 512 unique DNN weights, and each weight consists of four PCM conductances labeled G^+, G^-, g^+, g^- [see Fig. 1(f)]. Front-end and early metal levels were fabricated in a commercial foundry. Integration of doped Germanium-Antimony-Tellurium (GST) mushroom-cell PCM as well as top metal levels was done at the IBM Albany Nanotechnology Center [see Fig. 1(e)]. To access each of

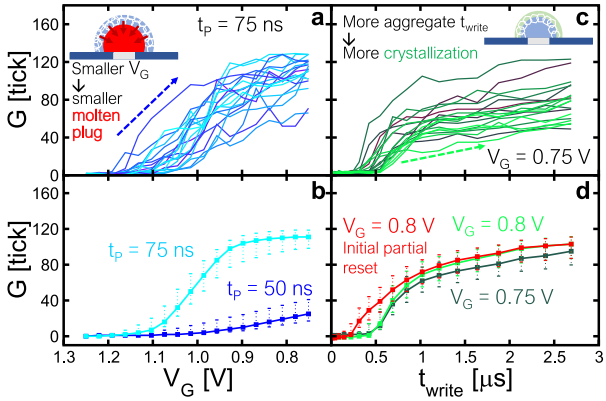

Fig. 2. Conductance trajectories for PCM devices starting from initial deep RESET, for (a) decreasing gate voltage V_G and (c) increasing total t_{write} (aggregating multiple programming pulses t_P). (b) Median, $\pm 1\sigma$ evolution across 512 × 512 devices. (d) Impact of starting from a partial-RESET rather than deep-RESET state.

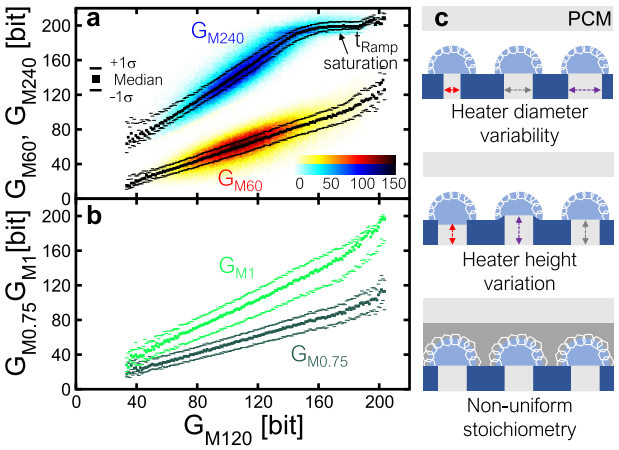

Fig. 3. PCM device-to-device variability measurements. (a) Correlation plot 512 × 512 devices) showing conductance achieved using a programming pulsewidth of either 60 (red) or 240 (blue) ns versus a programming pulsewidth of 120 ns. Devices that reach low/high conductances with a 120-ns pulse also similarly reach low/high conductances with 60- or 240-ns pulses. Lines show median and 1σ deviations at each point, demonstrating correlation is maintained across the entire range. (b) Similar results are obtained when varying the programming pulse amplitude V_G (0.75 and 1.0 V shown), which controls the compliance current for SET programming. (c) Device-to-device variability can be attributed to fixed physical variations in fabricated devices, including diameter, height and stoichiometry variations. [Note that since single unsigned conductances are being measured, the ramp start voltage was lowered to allow a wider dynamic range (maximum durations up to 200 ticks).]

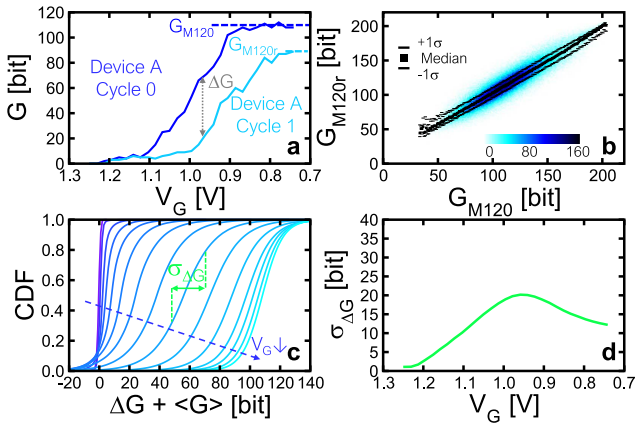

Fig. 4. (a) Cycle-to-cycle variation on a single device, showing significant deviation in device conductance evolution even with identical pulses applied. (b) While conductance evolution tends to be correlated, there is still non-negligible spread in the correlation plot, as shown both in the cloud of points (blue) and the median and 1σ deviations. (c) CDFs of conductance deviation (offset by the median conductance for readability) as well as (d) plots of standard deviation show that cycle-to-cycle variation is worse at intermediate states.

the four PCM devices, two types of read/write circuitry [see Fig. 1(g)] are integrated next to the PCM tiles: 1) row-wise circuitry able to inference one full row of either individual PCMs or entire weights, using the same inference path (and thus experiencing the same circuit non-linearities) as full-tile DNN inference and 2) random access ("Backdoor") read/write circuitry using a sense amp enabling calibrated but slow evaluation of individual PCM conductance in units of μS [see Fig. 1(h)]. Strong correlation is observed between the two read schemes [see Fig. 1(i)].

III. PCM Device Characterization and Weight Programming

PCM conductance change requires heating and quenching of the GST material by the application of an electrical pulse through a heating element. Our PCM programming methodology combines two different knobs for modulating the energy in a single programming pulse as shown in Fig. 2. One is the voltage amplitude, which controls current flow through the PCM, reducing the size of the amorphous plug over successive pulses and increasing the conductance of the device [see Fig. 2(a) and (b)]. The other is the pulse duration, where more crystallization occurs for longer pulses, increasing the conductance [see Fig. 2(c) and (d)].

While one could envision a programming scheme where each target conductance or weight would require a series of predetermined pulses of known amplitude and duration, PCMs (and other NVMs such as RRAM) suffer from a high degree of device-to-device and cycle-to-cycle variability. Device-to-device variability is shown in correlation plots Fig. 3(a) and (b). Here, the x-axis labeled G_{120} represents the final conductance achieved from a series of pulses of varying amplitude V_G and fixed duration of 120 ns. One can immediately see that for different devices, a wide range of final G values between 40 and 120 ticks is observed. Programming with shorter (60 ns) or longer (240 ns) durations shows a similar wide range, producing conductances strongly correlated with those achieved by programming with 120 ns [red and blue clouds in Fig. 3(a)]. This implies fixed device-to-device variability arising from variations in device structure and stoichiometry, as shown in Fig. 3(c). Similar trends are observed when programming with fixed pulsewidth but varying voltage amplitude V_G [see Fig. 3(b)].

Cycle-to-cycle variations were also studied. A single device programmed using the same sequence of pulses [see Fig. 4(a)] is shown to start and end at comparable conductance values, but follows a wildly different trajectory across two repeated

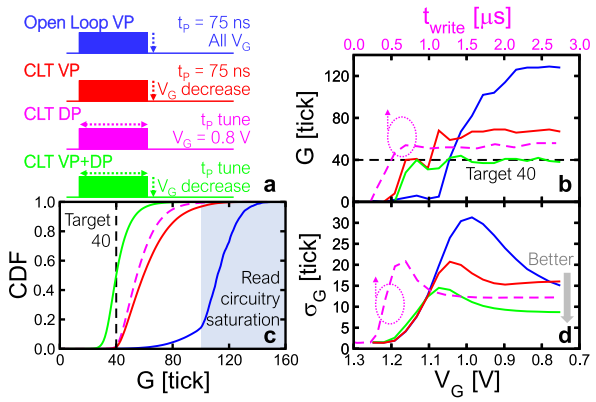

Fig. 5. (a) CLT can use voltage programming (VP), duration programming (DP) or both. (b) Conductance evolution for a single device using CLT, with a target of 40 ticks. (c) CDF of final conductance across 512 × 512 devices. (d) Best precision (lowest σ_G from CDF) is achieved using VP + DP CLT (green curve).

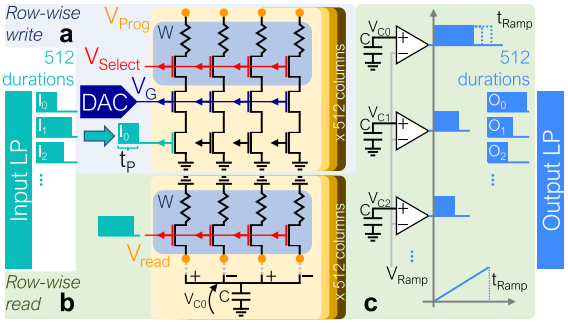

Fig. 6. (a) During row-wise write operation, a single DAC sets a common V_G for all columns, and per-column programming pulse-widths t_P are provided from the ILP. (b) During row-wise and inference read, current is integrated on to one peripheral capacitor per column. (c) Common ramp generator and at-column comparators then convert the voltage on the capacitors V_C into durations, which are routed on the 2-D mesh to the OLP.

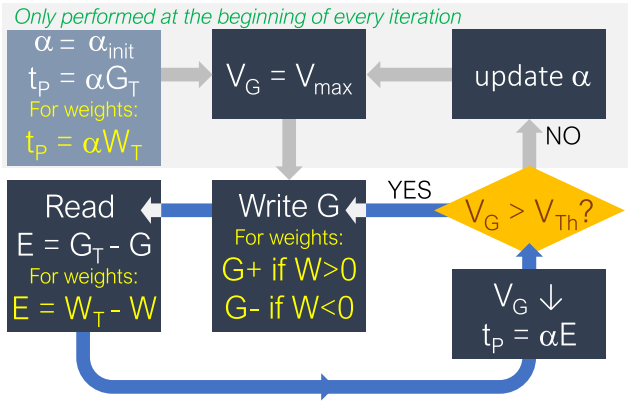

Fig. 7. Flow diagram for the CLT procedure for conductances (G) and weights (W). For simplicity, an optional final step for overshoot compensation is not shown.

experiments. This effect is analyzed and quantified across the entire array of 512 × 512 PCM conductances, as shown in correlation plots [see Fig. 4(b)], CDFs of the adjusted difference [ΔG offset by the median conductance, Fig. 4(c)] as well as standard deviation [see Fig. 4(d)]. All show significant non-negligible variation in conductance along intermediate states, making it clear that the exact conductance trajectory of the devices, and conversely the exact sequence of pulses to reach a precise intermediate state, is very difficult to predict.

To overcome this issue, CLT strategies–a series of read and write operations with programming conditions for a write operation adjusted based on the result of the previous read–are required for converging to the target weight. Additionally, in the case of programming a weight composed of multiple conductances, secondary devices g^+, g^- allow us to compensate for undershoot or overshoot in the primary G^+ or G^-. Our experiments across large arrays have also shown that combining the two knobs–descending gate voltage together with shorter pulse durations as we get closer to the target–enables "self-quenching" of PCMs producing tight conductance CDFs (see Fig. 5).

However, programming a single device at a time using CLT would not be scalable to large arrays and multiple tiles needed for large inference models. Hence the chip allows for row-wise programming, where all weights along a selected row can be tuned at the same time. The circuitry for programming is shown in Fig. 6(a). V_{Select} chooses a single row. A tile-based DAC sets a common programming voltage V_G for all columns. To allow for unique programming conditions on each column, unique durations are supplied on the 2-D mesh from the ILP. For row-wise read [see Fig. 6(b)], the ILP now supplies a constant duration to the selected row, allowing a constant V_{READ} voltage to conduct read-current up through 1, 2 or 4 PCMs per unit-cell. Each column's current is either the current from a single unit cell (for CLT read) or aggregated across all 512 rows (for inference) and integrated over time onto a column capacitor.

The tile then executes voltage-to-duration conversion on all 512 columns, using per-column comparator circuits and a common ramp [see Fig. 6(c)]. During an inference operation, this conversion is also implementing an *in situ* bounded ReLU, hard sigmoid or hard tanh activation function depending on the shape of the ramp and the exact start-stop voltages. Durations are sent in parallel over the 2-D mesh. During inference, these durations would then be received and consumed on another tile. During weight programming reads, they are captured and digitized to 8 bits by the OLP (1 tick ∼ 1.2 ns).

In general, to provide sufficient read signal strength across different scenarios (single device versus 512 devices, widely varying PCM conductance ranges), a range of read circuit scale factors are available. These include PCM read voltages (V_{READ}), integration capacitor sizes, total time of integration, current mirror ratios, and ramp speeds. These scale factors do impact the final output of a read operation as measured in ticks. However, when needed a conventional sense amplifier can be used to draw correspondence between ticks and the raw device conductance in micro-siemens, as shown in Fig. 1(i).

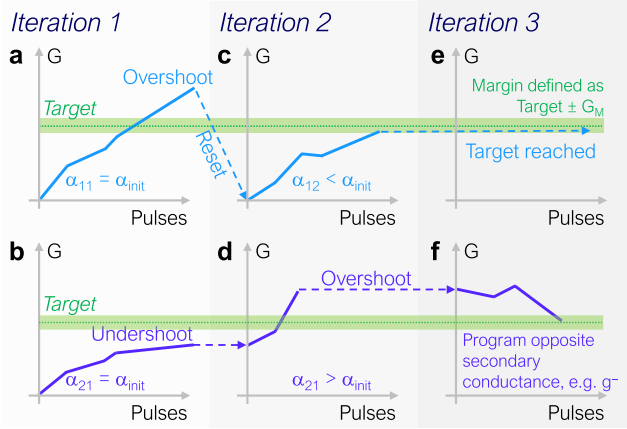

Fig. 8. Per-cell proportionality constant α determines the relationship between weight error and programming duration. (a) and (b) Initially, all devices are initialized to the same α_{init}. (c) and (d) Then, devices that overshoot/undershoot have their individual α values decreased/increased. (e) and (f) Optional final correction is done using the opposite secondary conductance.

The complete row-wise CLT programming algorithm, developed based on device programming behavior and optimized for the available circuitry, is described in Fig. 7. The algorithm is applicable to both individual conductances and full weights. Within each programming cycle, square pulses of decreasing pulse amplitude (V_G) are applied in succession, with a read operation after each pulse. An FPGA calculates the weight error and then determines the next vector of programming pulse durations t_P based on per-cell proportionality constants α. These durations are loaded into the input landing pad for the next write. All devices are initialized to a constant α, which changes depending on the response of a particular device [devices that overshoot/undershoot have their α reduced/increased in the next cycle (see Fig. 8)]. An optional final step programs the opposite secondary conductances to compensate for any residual overshoot [see Fig. 8(f)].

IV. Programming Results

To evaluate the efficacy of the row-wise CLT algorithm, we programmed simple linear target patterns [see Fig. 9(a), (c), and (e)] on 512 × 512 unit cells. The actual conductances/weights achieved after CLT are shown in Fig. 9(b), (d), and (f) and in the CDFs in Fig. 9(g), (h), and (i). While most of the weight information can be programmed into a single primary conductance, as shown in Fig. 9(a), (b), and (g), having one or three additional conductances allows us to compensate for overshoots and undershoots.

While our PCM weight tuning approach is based on continuously variable analog targets, and is not a true multilevel cell (MLC) with distinct states, we have tried to quantify the effective bits-of-precision for our programming approach. Fig. 9(j) and (k) show average weight-error, and the average absolute weight error versus weight target in number of ticks for each approach. The gray bands in each figure represent the error bounds for 3-, 4-, and 5-b precision. From these figures, our average weight error is <3%, corresponding to >3 bits of precision.

V. DNN Demonstrations

To evaluate weight-programming precision, on-chip MAC accuracy and the 2-D mesh data transport circuitry, we first programmed MNIST weights from a cropped two-layer fully-connected network (512 × 252 × 10) onto two PCM tiles on chip [see Fig. 10(a)]. Input activations were loaded into the ILP using an FPGA. These were converted into durations and routed on the mesh to the first tile, where a MAC operation was performed. A hard-sigmoid activation function was implemented at the output of the first stage on each column using the ramp-and-duration conversion circuits [see Fig. 6(c)]. Durations were then routed to the second tile, a second MAC performed, resulting durations captured in the OLP and analyzed on the external FPGA. The color maps in Fig. 10(b) and (c) show excellent correlation between actual and target weights for the two layers. Each tile also features eight bias rows, as shown in Fig. 11(a). These bias rows receive a constant input during MAC operations, and while the neural network itself was trained without bias, the bias weights in these rows were used for calibration. The procedure involves using the CLT algorithm to program these weights to counteract the intrinsic circuit offset of each column [see Fig. 11(b) and (c)].

This fully on-chip multi-tile demonstration achieved near-software-equivalent test accuracy [97.13% experimental versus 97.73% in software, Fig. 10(d)] without any external processing or off-chip communication. The primary source for the slight drop in accuracy seems to be the on-chip MAC operations since a mixed hardware–software (HS) experiment with ideal software MAC operations performed on the as-programmed hardware weight results in an accuracy of 97.55%. Improved calibration strategies to further reduce the hardware MAC error are currently under investigation.

MAC errors could have a much more pronounced impact on recurrent neural networks such as long short-term memory (LSTM), commonly used in language tasks such as prediction, translation etc. These networks are inherently sequential, with the output at any given time step h_t used to generate the next output h_{t+1}. In such cases, there can be considerable error accumulation over a long sequence, even with small errors in individual MACs.

We implemented character prediction for the Alice in Wonderland dataset. Here we evaluated a one-layer LSTM, with embedding and output layers in software [see Fig. 12(a)]. The LSTM weight matrices were programmed on chip [see Fig. 12(c)], using the closed-loop weight programming algorithm. High programming accuracy was achieved, as shown in Fig. 12(d), and the MAC was implemented on chip. However, activation and vector-vector calculations to determine the next h_t and c_t vectors were done on the FPGA as shown in Fig. 12(a). The off-chip calculations also simplified the offset calibration process–this flexibility offered by digital arithmetic is a strong argument in favor of eventual adoption of ADCs and digital vector processing units, provided the area and power impact can be carefully managed.

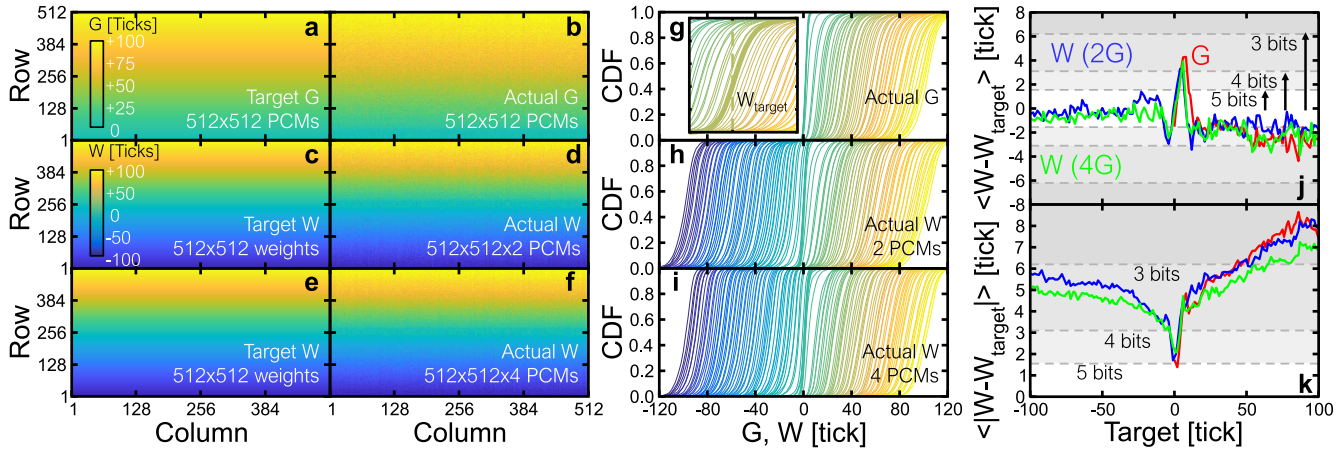

Fig. 9. (a), (b), and (g) Single-PCM, (c), (d), and (h) 2-PCM, and (e), (f), and (i) 4-PCM weight programming. (a), (c), and (e) Using a simple linear target pattern, (b), (d), and (f) 512 × 512 weights are programmed using CLT; (g), (h), and (i) resulting CDFs by target-level. (j) Average weight error is <3% and (k) for >3 effective bits-of-precision.

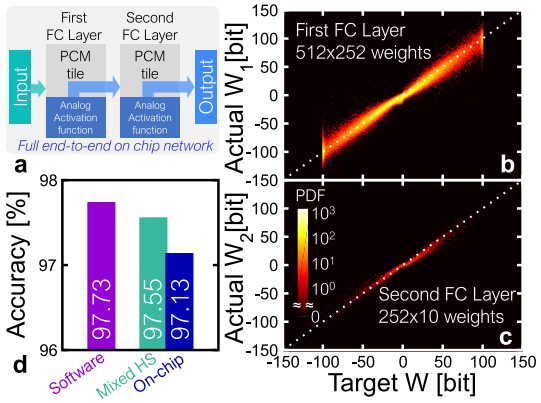

Fig. 10. (a) End-to-end two-layer MNIST demonstration with on-chip duration communication. (b) and (c) 512-252-10 weights programmed in two tiles, achieving (d) near software-equivalent accuracy.

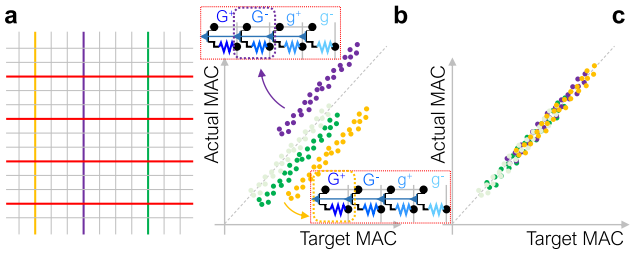

Fig. 11. (a) Fixed circuit offset compensation is accomplished by programming bias rows (shown in red). (b) Columns that are overshooting/undershooting have their bias weights reduced/increased, respectively, using the CLT algorithm, leading to offset-corrected MACs on all columns (c).

In addition to the mixed HS experiment to evaluate weight programming accuracy, a "Perfect recurrence" experiment was carried out to evaluate the impact of error propagation across multiple time steps (see Fig. 13). In this experiment, while the on-chip MAC is calculated at each step, it is not used as an input for the next step; instead the ideal inputs are used. In the "Actual Recurrence" experiments we allow for error propagation across time steps.

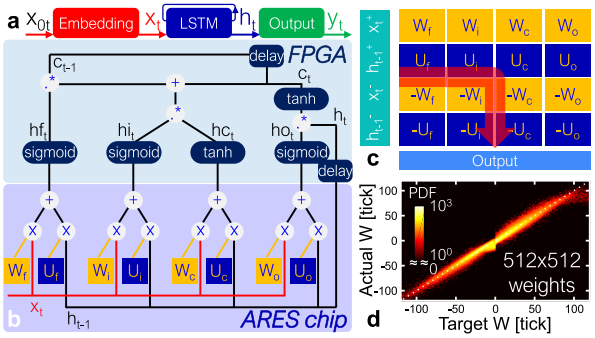

Fig. 12. LSTM layer details: hidden-layer size is 128. (a) Embedding and output layers are implemented on the FPGA, LSTM weights are programmed on-chip using CLT algorithm. (b) MACs are on chip, activation functions are in FPGA. (c) Software weights are replicated with inverted sign to enable positive-only inputs. (d) Actual programmed weights.

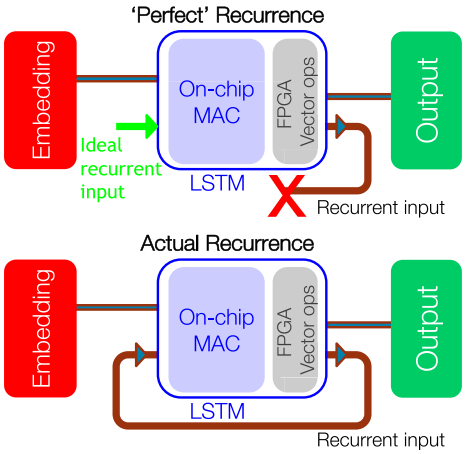

Fig. 13. In "perfect" recurrence (top), noise accumulation over multiple time steps is eliminated by providing ideal inputs at each step. In actual recurrence (bottom), the outputs calculated from on-chip MAC operations become the input for the next step.

Correlation between actual and ideal MAC are shown for the four cases (on-chip versus mixed HS, perfect versus actual recurrence) in Fig. 14. There is around 1.2%–1.4% increase in

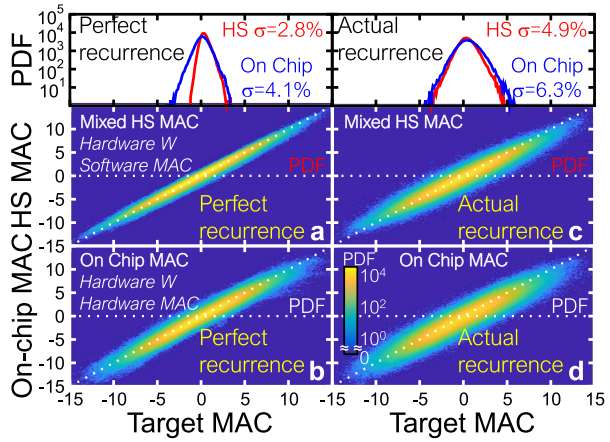

Fig. 14. Comparisons of (a) and (c) mixed HS MAC and (b) and (d) on-chip MAC show small degradation versus software baseline.

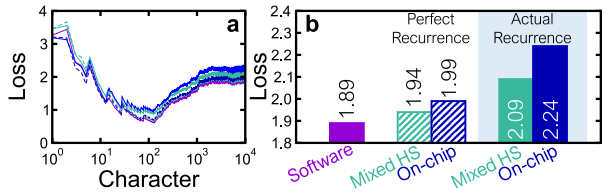

Fig. 15. Evolution of LSTM loss over a long test sequence (a) quantifying how well the y_t vector predicts the next character of "Alice in Wonderland" and (b) for software, mixed HS, and on-chip MACs under both Perfect and Actual Recurrence. Lower loss indicates more accurate predictions.

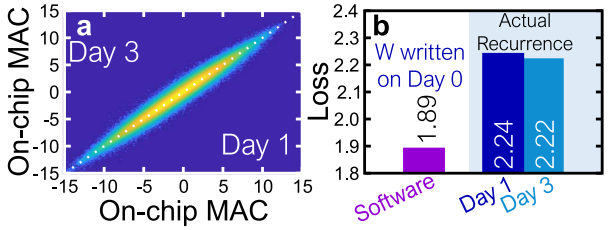

Fig. 16. Impact of resistance drift on LSTM. (a) on-chip MACs at days 1 and 3 after weight programming exhibit (b) no loss degradation.

MAC-error (σ of the MAC-correlation) going from mixed HS to on-chip MAC, for both the perfect and actual recurrence cases, and a 2.2% increase due to error feedback across a long sequence (up to 10 000 characters). Overall this leads to a higher (worse) on-chip cross entropy loss of 2.24 (versus 1.89 for software), as shown in Fig. 15.

Finally, we evaluated the impact of PCM resistance drift [20], [21] on DNN accuracy. MAC correlation and loss plots (see Fig. 16) show negligible impact even after three days. More drift studies are ongoing. Drift mitigation will need a multi-pronged approach across device engineering [22], programming procedure, circuit slope correction [23] as well as algorithms, including drift-aware training approaches– [17], [24], [25].

VI. Conclusion

A multi-tile inference chip implementing in-memory MAC on analog PCM arrays was demonstrated. The memory elements are integrated above 14-nm CMOS. The chip does not have ADCs for digital conversion, instead transforming analog voltages into durations which are buffered and communicated on a reconfigurable 2-D mesh. This design choice saves area and power, at some cost to flexibility. For accurate closed loop weight-tuning, we employed a row-wise programming algorithm that efficiently programs the four PCM devices in each analog weight with minimal overshoot. This scheme achieved on average <3% weight-error on tiles with up to 512 × 512 weights. We implemented MNIST at near-software-equivalent accuracy, demonstrating tile-to-tile transport with a fully-on-chip two-layer network. We also tested resilience to error propagation with a recurrent Alice LSTM, using off-chip activation functions to calculate recurrent inputs for the next on-chip MAC. Finally, we showed that drift did not impact accuracy over three days. Future work includes large networks and quantifying performance and energy-efficiency metrics.

Acknowledgment

This article was invited for consideration in the TED Special Issue for select papers from VLSI 2021. The authors would like to thank IBM Albany Nanotech Center, Albany, NY, USA, and IBM Bromont, Bromont, QC, Canada, for device and module fabrication; and W. Wilcke, S. Narayan, S. Munetoh, S. Yamamichi, C. Goldberg, C. Osborn, J. Burns, R. Divakaruni, and M. Khare for management and logistical support.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[2] Z. Zhang, J. Geiger, J. Pohjalainen, A. E.-D. Mousa, W. Jin, and B. Schuller, "Deep learning for environmentally robust speech recognition: An overview of recent developments," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 5, pp. 1–28, Jul. 2018.

[3] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," Sep. 2016, *arXiv:1609.08144*. [Online]. Available: https://arxiv.org/abs/1609.08144

[4] D. Amodei and D. Hernandez, "AI and compute," OpenAI, San Francisco, CA, USA, Tech. Rep. [Online]. Available: https://openai.com/blog/ai-and-compute/

[5] B. Fleischer *et al.*, "A scalable multi-TeraOPS deep learning processor core for AI trainina and inference," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2018, pp. 35–36.

[6] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 367–379.

[7] J. Fowers *et al.*, "A configurable cloud-scale DNN processor for real-time AI," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 1–14.

[8] T. Chen *et al.*, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proc. 19th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, New York, NY, USA, 2014, pp. 269–284.

[9] N. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, New York, NY, USA, 2017, pp. 1–12.

[10] H. Jia *et al.*, "A programmable neural-network inference accelerator based on scalable in-memory computing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, Feb. 2021, pp. 236–238.

[11] H.-Y. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr, "Recent progress in analog memory-based accelerators for deep learning," *J. Phys. D, Appl. Phys.*, vol. 51, no. 28, 2018, Art. no. 283001.

[12] S. Ambrogio *et al.*, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, pp. 60–67, Jun. 2018.

[13] B. Yan *et al.*, "RRAM-based spiking nonvolatile computing-in-memory processing engine with precision-configurable *in situ* nonlinear activation," in *Proc. Symp. VLSI Technol.*, Jun. 2019, pp. T86–T87.

[14] F. Cai *et al.*, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electron.*, vol. 2, no. 7, pp. 290–299, Jul. 2019.

[15] P. Yao *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, Jan. 2020.

[16] I. Boybat *et al.*, "Neuromorphic computing with multi-memristive synapses," *Nature Commun.*, vol. 9, p. 2514, Jun. 2018.

[17] V. Joshi *et al.*, "Accurate deep neural network inference using computational phase-change memory," *Nature Commun.*, vol. 11, no. 1, p. 2473, May 2020.

[18] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1769–1774.

[19] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, Mar. 1995.

[20] J. Li, B. Luan, and C. Lam, "Resistance drift in phase change memory," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Apr. 2012, pp. 6C.1.1–6C.1.6.

[21] S. Lavizzari, D. Ielmini, D. Sharma, and A. L. Lacaita, "Reliability impact of chalcogenide-structure relaxation in phase-change memory (PCM) cells—Part II: Physics-based modeling," *IEEE Trans. Electron Devices*, vol. 56, no. 5, pp. 1078–1085, May 2009.

[22] R. L. Bruce *et al.*, "Mushroom-type phase change memory with projection liner: An array-level demonstration of conductance drift and noise mitigation," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Mar. 2021, pp. 1–6.

[23] S. Ambrogio *et al.*, "Reducing the impact of phase-change memory conductance drift on the inference of large-scale hardware neural networks," in *IEDM Tech. Dig.*, Dec. 2019, pp. 6.1.1–6.1.4.

[24] S. Kariyappa *et al.*, "Noise-resilient DNN: Tolerating noise in PCM-based AI accelerators via noise-aware training," *IEEE Trans. Electron Devices*, vol. 68, no. 9, pp. 4356–4362, Sep. 2021.

[25] K. Spoon *et al.*, "Toward software-equivalent accuracy on transformer-based deep neural networks with analog memory devices," *Frontiers Comput. Neurosci.*, vol. 15, p. 53, Jul. 2021.