A Multi-Memristive Unit-Cell Array With Diagonal Interconnects for In-Memory Computing

Riduan Khaddam-Aljameh, *Graduate Student Member, IEEE*, Michele Martemucci, *Student Member, IEEE*, Benedikt Kersting, *Graduate Student Member, IEEE*, Manuel Le Gallo, *Member, IEEE*, Robert L. Bruce, Matthew BrightSky, and Abu Sebastian, *Senior Member, IEEE*

***Abstract*—Memristive crossbar arrays can be used to realize matrix-vector multiplication (MVM) operations in constant time complexity by exploiting the Kirchhoff's circuit laws. This is enabled by the parallel read of the entire array in a single time step. However, parallel writing is prohibitive in such arrays due to limitations on the current that could be accumulated along the wires. Hence, loading the matrix elements into such an array still incurs significant time penalty. Another key challenge is the achievable computational precision. To overcome these challenges, we propose a unit-cell array design where each unit-cell comprises four memristive devices each attached to a selection transistor. Moreover, the array is organized in such a way that the selection transistors can be turned on in a diagonal fashion. We experimentally demonstrated this concept by fabricating a 2 × 2 unit-cell array based on projected phase-change memory (PCM) devices in 90 nm CMOS technology. It is shown that using the diagonal connections, the write operations can be parallelized while maintaining the current limit of the back-end-of-the-line metallization. Moreover, the increase in write time due to having more devices per unit-cell is minimized through a combination of single-shot and iterative programming schemes. Finally, we present experimental results on MVM operations that demonstrate improved computational precision exceeding that of a 4-bit fixed-point implementation.**

***Index Terms*—In-memory computing, phase-change memory, analog computation.**

I. Introduction

IN-MEMORY computing (IMC) is an emerging paradigm where certain low-level computational tasks are executed within a memory array using the physical attributes of the memory devices and their array-level organization [1], [2].

Manuscript received March 8, 2021; accepted April 26, 2021. Date of publication May 10, 2021; date of current version November 24, 2021. This work was supported in part by the IBM Research AI Hardware Center (ibm.co/ai-hardware-center) and in part by the European Research Council (ERC) through the European Union's Horizon 2020 Research and Innovation Programme under Grant 682675. This brief was recommended by Associate Editor E. Vianello. *(Corresponding author: Riduan Khaddam-Aljameh.)*

Riduan Khaddam-Aljameh is with the Cloud and AI Systems Research, IBM Research Europe, 8803 Rüschlikon, Switzerland, and also with the Integrated Systems Laboratory, ETH Zurich, 8092 Zürich, Switzerland (e-mail: rid@zurich.ibm.com).

Michele Martemucci, Benedikt Kersting, Manuel Le Gallo, and Abu Sebastian are with the Cloud and AI Systems Research, IBM Research Europe, 8803 Rüschlikon, Switzerland.

Robert L. Bruce and Matthew BrightSky are with Phase Change Memory, IBM Research, Yorktown Heights, NY 10598 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3078614.

Digital Object Identifier 10.1109/TCSII.2021.3078614

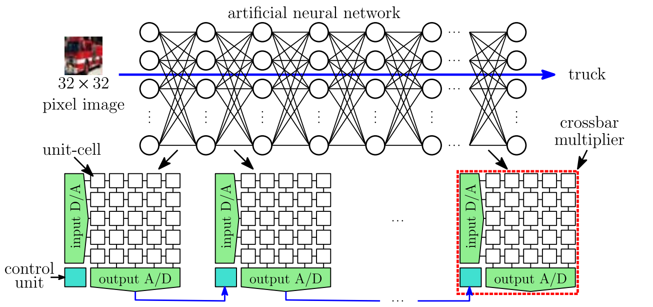

Fig. 1. Schematic illustration of a multilayer artificial neural network that can be used for image classification. The MVMs that are executed in each layer can be implemented using memristive crossbar arrays.

For example, memristive devices [3] storing the matrix elements in terms of their conductance values when organized in a crossbar configuration can perform matrix-vector multiplication (MVM) operations in constant time complexity by exploiting the Kirchhoff's circuit laws. This makes IMC very attractive for applications such as deep neural network (DNN) inference (Fig. 1) and training [4], [5], pattern matching [6], compressed sensing [7] and random number generation [8].

The $\mathcal{O}(1)$ complexity MVM operation is enabled by the parallel read of the entire crossbar array. This can also be done in reverse, by swapping the axis from which the inputs are applied to the crossbar, thus executing a transposed MVM. However, it is quite challenging to write or program multiple devices in parallel and this significantly limits the latency incurred when storing the matrix elements. Existing techniques [9], [10] that aim at maximizing the parallelization of write-operations require abandoning the support of transpose MVM and bidirectional read operations [see Fig. 2(a)], which is when both positive and negative inputs are applied simultaneously. If a different crossbar topology is chosen that maintains all IMC-related functionalities, then parallel write-operations would fail due to high current accumulation on the metal wires [Fig. 2(b)]. This results in limited scalability, decreased reliability and increased programming stochasticity due to state dependent voltage-drop [11]. Achieving at least $\mathcal{O}(N)$-complexity for carrying out programming operations on memristive crossbars is desirable to accelerate the import of pre-trained weights for DNN inference [4] and is essential to enable applications such as DNN training [12]–[14].

Another key challenge associated with MVM using IMC is the lack of precision [15]. This is addressed both at the

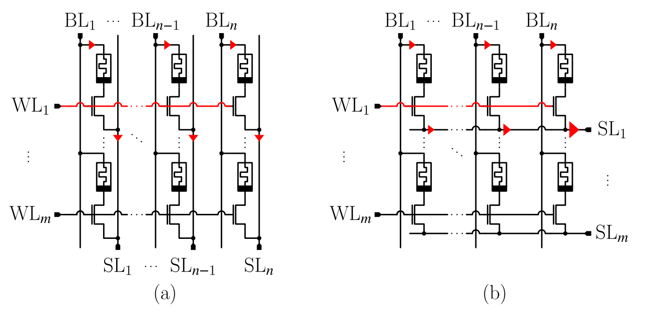

Fig. 2. Conventional memristive crossbar designs with different arrangements of bit-lines (BLs), word-lines (WLs) and source-lines (SLs). (a) This design with parallel BLs and SLs allows parallel programming at $\mathcal{O}(N)$ complexity. However, this crossbar only supports unidirectional inputs and no transposed MVM, since the modulation happens via the WL-signals. (b) The alternative design permits bidirectional input application due to the orthogonal BLs and SLs. Parallel write however fails, as the high programming currents of the full row would accumulate on a single wire.

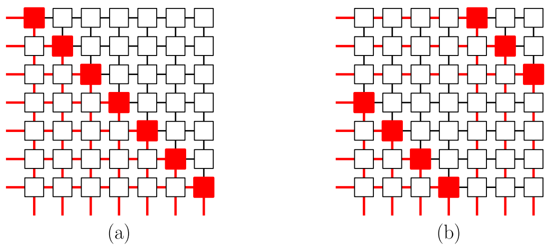

Fig. 3. Diagonal selection concept that facilitates O(N) programming in an $N \times N$ crossbar array. If there is provision to select unit-cells in a diagonal manner, it is possible to select N devices in parallel such that exactly one unit-cell is selected per row and column. Hence, independent of the array size, the maximum current in the wiring will never surpass the programming current for a single device. Two representative examples are shown where (a) the unit-cells along the main diagonal are selected and where (b) the cells along two peripheral diagonals are selected.

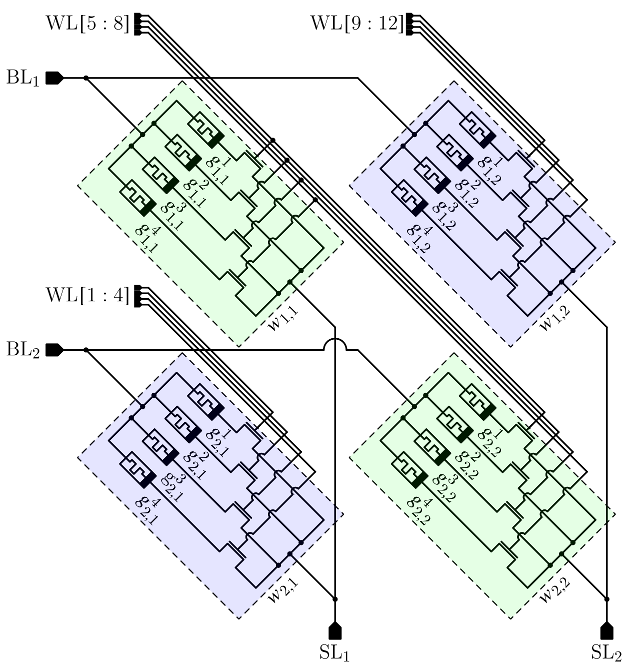

Fig. 4. Schematic illustration of the proposed unit-cell array. A 2×2 crossbar segment of the array is shown that is able to store 4 synaptic weights w_{11}, w_{12}, w_{21} and w_{22}. Each unit cell is comprised of 4 PCM devices each with an NMOS access device. The top electrodes of the PCM devices are shorted together to the corresponding bit-lines BL$_i$, while the transistor sources connect to the source lines SL$_j$. 12 WLs connected to the gates of the transistors facilitate the diagonal selection.

device-level with concepts such as projected phase-change memory (PCM) [16]–[18] and at the architectural level via multi-memristive unit-cells [19], [20]. Here, we present an array design that facilitates $\mathcal{O}(N)$ programming, allows normal and transposed analog MVM operations, and by combining the concepts of projected PCM and multi-memristive unit-cells, achieves computational precision that exceeds a digital 4-bit fixed point implementation. The design is experimentally validated using a 2×2 array-prototype based on PCM devices fabricated in 90 nm CMOS technology.

II. Unit-Cell Array Design

The proposed concept consists of a crossbar array of unit-cells storing the matrix elements. To facilitate programming of the array with $\mathcal{O}(N)$ time complexity, the unit-cells are selected in a diagonal manner as shown in Fig. 3. As a proof of concept, a crossbar with 2×2 unit-cells was proposed as shown in Fig. 4. To increase the computational precision associated with the MVM operation, each matrix element, $w_{i,j}$, is encoded in terms of the conductance value of four memristive devices denoted by $g^1_{i,j}$ to $g^4_{i,j}$. Through parallel connection, the sum of the device conductance values encode each matrix element, namely, $w_{i,j} = \sum_{k=1}^{4} g^k_{i,j}$. Each memristive device is attached to an NMOS transistor which serves as the selection device. The top electrodes of the memristive devices per unit-cell are connected to the bit-lines BL$_i$ and the NMOS transistor sources are connected to the source-lines SL$_j$. Thus, a bi-directional crossbar is formed, as the read direction can be arbitrarily set, given a sufficiently symmetric device (in terms of the $I - V$ characteristics) and a large enough access transistor. Within one unit-cell, each access NMOS transistor gate connects to a different WL, which is routed diagonally through the array. In the IMC-mode, all diagonal WLs are activated, enabling all devices and unit-cells, so that MVM operations can be performed. However, during programming, either a single WL from the center diagonals or a pair of WLs from the peripheral ones is activated. This facilitates programming of the array with $\mathcal{O}(N)$ time complexity. The diagonal wiring ensures that always just a single device is active per BL and SL. This also facilitates reliable, leakage current-free read-out of an individual device conductance which is essential for iterative programming.

The proposed crossbar array was designed and fabricated using mushroom-type PCM devices with a projection liner above the bottom electrode [16], [18]. The layout of the 2×2 crossbar is shown in Fig. 5. To achieve maximum areal density, the PCM devices are located directly on top of the transistors, as can be seen by the highlighted device $g^1_{2,1}$. It can be noted that the access device is kept conservatively large, beyond the common minimal size in 90 nm [21], so that devices with experimental material compositions that require high currents at high voltages can be tested. As a result, the unit-cell footprint of 11.76 µm^2 is defined by the size of the access-transistors. An optimistic projection of the

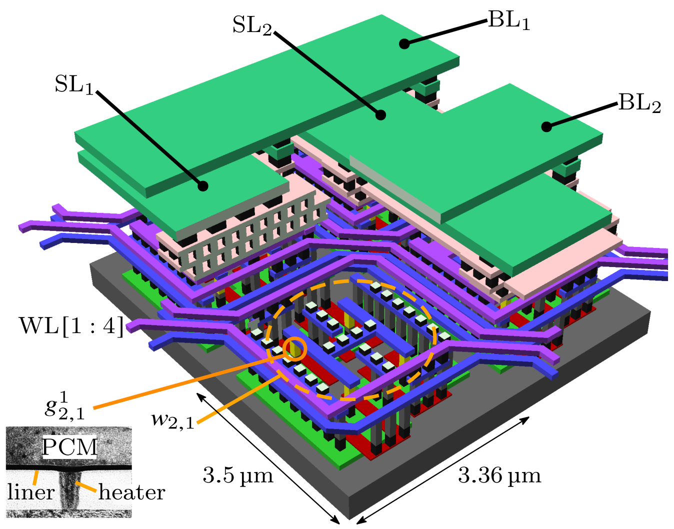

Fig. 5. 3-d view of the 2 × 2 array prototype. The four diagonal gate-lines per unit-cell are realized in metal 1 (blue) and 2 (purple). Unit-cell $w_{2,1}$ is exposed, so that the insertion point of the PCM devices $g^1_{2,1}$ to $g^4_{2,1}$ (in yellow) directly below the metal 1 becomes visible. The inset shows the TEM image of a representative mushroom-type PCM device with projection liner.

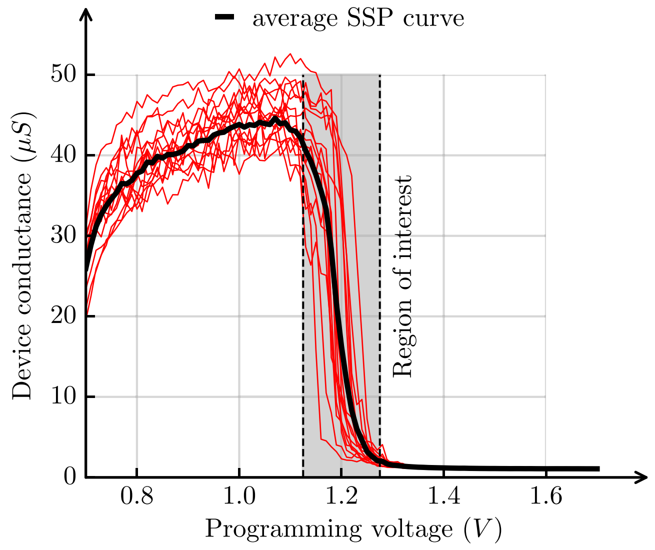

Fig. 7. Measured single-shot programming (SSP) curves for 16 PCM devices starting from the threshold voltage. Programming voltages are applied as 100 ns block pulses to devices in RESET state. The region of interest for SSP is in between 1.125 V and 1.275 V. For voltages higher than ca. 1.275 V the device stays in RESET.

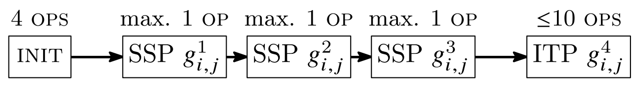

Fig. 6. Block diagram description of the programming algorithm that combines single-shot (SSP) with iterative programming (ITP).

area assuming conventional minimal-sized access-transistors of 100 F^2 would be ca. 3.2 µm^2, which is around 28 % of the current size.

The four diagonally connected WL signals per unit-cell are wired on the first two metals, which are highlighted in blue and purple. Note how those wires encircle the unit-cell and connect each via three redundant staggered vias to the gate poly below. Diagonal routing, which is explicitly permitted in standard foundry rules of 90 nm technologies, is restricted to the crossings from one unit-cell to the next.

The routing of BL$_i$ and SL$_j$ is done on the most conductive metal available in the stack that can be used within the given cell size. In this case it was the two double-width metals on the top, which are coloured green in Fig. 5. This is to minimize the cell-to-cell resistance and any undesirable interconnection resistance (IR)-drop effects during the massively parallel read operations in the IMC-mode [11].

III. Programming Algorithm

The fabricated array prototype was used to store the matrix elements and to perform MVM operations. As the programming current is increased, the size of the amorphous region increases progressively and this results in a near monotonic relation between the device conductance and the programming current. This characteristic, typically referred to as the programming curve, however, exhibits significant inter and even intra-device variability. Hence it is essential to resort to iterative programming (ITP) to accurately program a PCM device to a desired conductance value [22]. Multiple write-verify steps are executed as part of ITP.

To minimize the energy and time that is required to program the conductance of one unit-cell $w_{i,j}$ to a target value of $w^*_{i,j}$, we propose a combined single-shot programming (SSP) and ITP scheme, as shown in Fig. 6. At first the unit-cell is initialized by programming $g^1_{i,j}$, $g^2_{i,j}$ and $g^3_{i,j}$ to a RESET state (close to zero conductance) and $g^4_{i,j}$ to a SET state (maximum conductance). If the residual conductance $w^*_{i,j} - g^4_{i,j}$ is greater than zero, then in three successive SSP steps, the conductances $g^1_{i,j}$ to $g^3_{i,j}$ are programmed to account for it. The SSP current value is determined based on the characterized mean programming curve that is obtained from multiple device measurements, as shown in Fig. 7. Finally, after the last SSP step, the unit-cell is in a state where the target conductance $w^*_{i,j}$ can be reached via ITP on $g^4_{i,j}$. Should $w^*_{i,j}$ be smaller than the conductance $g^4_{i,j}$, which is initialized into a SET state, then the SSP steps are omitted and the algorithm directly proceeds with ITP on $g^4_{i,j}$. In case of failure, e.g., the accumulated error during the SSP steps becomes larger than the tuneable range of the ITP of $g^4_{i,j}$, the algorithm has to restart. Note that the dynamic range of the unit-cell is extended fourfold whereas the average number of programming steps only increased by three – one more step for each additional device in the unit-cell.

The performance of the algorithm was experimentally validated on the array-prototype using a custom-built characterization platform that comprised a probe card, a signal generator, a switching matrix, and a source measure unit. This is done by programming all four unit-cells to different states using the proposed scheme. Due to the diagonal connections, two devices can be programmed in parallel, so that per programming-step a device pair from unit-cells $w_{1,1}$ and $w_{2,2}$ or $w_{1,2}$ and $w_{2,1}$ is activated. As it can be seen from the

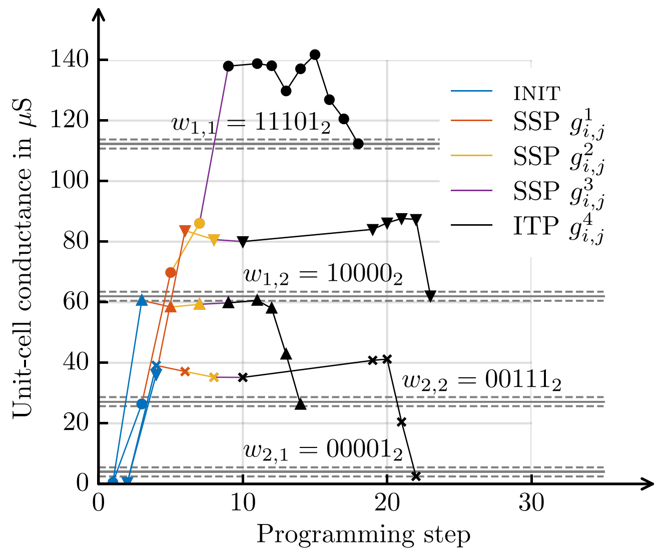

Fig. 8. Illustration of diagonal selection and programming using the combined SSP/ITP algorithm. Device pairs on a matrix diagonal, in this case w_{11} & w_{22} or w_{21} & w_{12}, are programmed simultaneously to their respective 5-bit target values. The different operations associated with the programming algorithm are color coded. Convergence is achieved once the conductance value enters a tolerance band of $\pm 0.5 \cdot$ LSB.

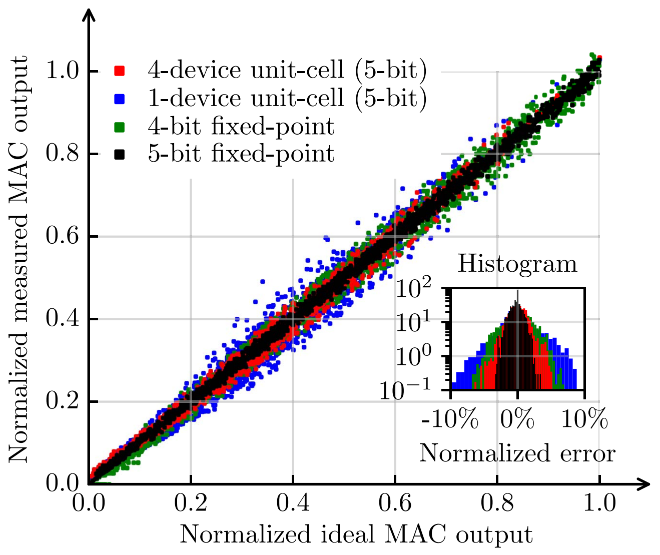

Fig. 9. Experimental results showing the precision associated with the MVM operations carried out using the 2×2 array-prototype. The measured MVM result in terms of the SL current is plotted against the ideal MVM result. It is seen that we can achieve higher computational precision using all four devices in the unit cell as opposed to using just one device.

results shown in Fig. 8, the proposed programming scheme concludes after 23 steps, which is when all four unit-cells have reached the tolerance band of their respective 5-bit target value. Note that each programming step consists of a combined write and read-back operation, so that programming errors can be corrected in the subsequent cycles.

IV. MVM Results

Finally, to quantify the computational precision, MVM operations were performed using the array-prototype employing either a single PCM-device or the full unit-cell. The results were also compared to that of low-precision fully digital 4-bit and 5-bit fixed-point implementations. First, the unit-cells are programmed to various combinations of 5-bit conductance states, such that given defined input vectors, the ideal MVM results would cover the full output range. The application of the input stimuli is achieved by means of read-voltage amplitude modulation along the BLs and the MVM is obtained by digitizing the measured current sinked from the SLs. As indicated in Fig. 9, the computational precision is significantly higher when employing all the 4 PCM devices in the unit-cell compared to a single device. Moreover the computational precision of the 4 device implementation exceeds that of the 4-bit fixed-point implementation.

V. Conclusion

In summary, a unit-cell array is proposed for IMC where each unit-cell comprises 4 PCM devices connected in parallel to increase the computational precision. Moreover, the devices are selected in a diagonal fashion. An efficient programming scheme combining single-shot and iterative programming schemes is proposed to partially alleviate the increased time and energy cost associated with programming 4 devices. Experimental results on a 2×2 array-prototype fabricated in 90 nm CMOS technology node demonstrate the efficacy of the proposed approach. This work could extend the application domain for IMC by reducing the write latency and increasing the computational precision.

Further improvements of the proposed work could include the support for signed weight storage in the unit-cell and the demonstration of a fully integrated diagonal select-based crossbar system with on-chip ADCs and programming units [23].

Acknowledgment

The authors thank R. Heller, U. Egger, P. Jonnalagadda, P.-A. Francese, E. Eleftheriou, and L. Benini for technical and management support.

References

[1] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nat. Nanotechnol.*, vol. 15, pp. 529–544, Mar. 2020.
[2] Y. Xi *et al.*, "In-memory learning with analog resistive switching memory: A review and perspective," *Proc. IEEE*, vol. 109, no. 1, pp. 14–42, Jan. 2021.
[3] H. Pozidis, N. Papandreou, and M. Stanisavljevic, "Circuit and system-level aspects of phase change memory," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 3, pp. 844–850, Mar. 2021.
[4] V. Joshi *et al.*, "Accurate deep neural network inference using computational phase-change memory," *Nat. Commun.*, vol. 11, no. 1, p. 2473, May 2020.
[5] C.-X. Xue *et al.*, "15.4 a 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2020, pp. 244–246.
[6] C.-J. Jhang, C.-X. Xue, J.-M. Hung, F.-C. Chang, and M.-F. Chang, "Challenges and trends of SRAM-based computing-in-memory for AI edge devices," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 5, pp. 1773–1786, May 2021.

[7] M. L. Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed sensing with approximate message passing using in-memory computing," *IEEE Trans. Electron Devices*, vol. 65, no. 10, pp. 4304–4312, Oct. 2018.

[8] H. Jiang *et al.*, "A novel true random number generator based on a stochastic diffusive memristor," *Nat. Commun.*, vol. 8, no. 1, p. 882, Oct. 2017.

[9] J. Chen, H. Wu, B. Gao, J. Tang, X. S. Hu, and H. Qian, "A parallel multibit programing scheme with high precision for rram-based neuromorphic systems," *IEEE Trans. Electron Devices*, vol. 67, no. 5, pp. 2213–2217, May 2020.

[10] C. Mackin *et al.*, "Neuromorphic computing with phase change, device reliability, and variability challenges," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Apr. 2020, pp. 1–10.

[11] A. Chen, "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics," *IEEE Trans. Electron Devices*, vol. 60, no. 4, pp. 1318–1326, Apr. 2013.

[12] S. R. Nandakumar *et al.*, "Mixed-precision deep learning based on computational memory," *Front. Neurosci.*, vol. 14, p. 406, May 2020.

[13] H. 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, Jul. 2018, Art. no. 283001.

[14] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proc. IEEE*, vol. 106, no. 2, pp. 260–285, Feb. 2018.

[15] A. S. Rekhi *et al.*, "Analog/mixed-signal hardware error modeling for deep learning inference," in *Proc. 56th Annu. Design Autom. Conf. DAC*, Las Vegas, NV, USA, 2019, pp. 1–6.

[16] W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann, and E. Eleftheriou, "Projected phase-change memory devices," *Nat. Commun.*, vol. 6, p. 8181, Sep. 2015.

[17] I. Giannopoulos *et al.*, "8-bit precision in-memory multiplication with projected phase-change memory," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Dec. 2018, pp. 27.7.1–27.7.4.

[18] R. 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. Symp. Rel. Phys.*, 2021, pp. 1–6.

[19] I. Boybat *et al.*, "Neuromorphic computing with multi-memristive synapses," *Nat. Commun.*, vol. 9, no. 1, p. 2514, Jun. 2018.

[20] S. Yu, R. Shafik, T. Bunnam, K. Chen, and A. Yakovlev, "Self-amplifying current-mode multiplier design using a multi-memristor crossbar cell structure," in *Proc. 27th IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Nov. 2020, pp. 1–4.

[21] G. F. Close *et al.*, "A 256-Mcell phase-change memory chip operating at 2+ bit/cell," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 6, pp. 1521–1533, Jun. 2013.

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

[23] R. Khaddam-Aljameh *et al.*, "HERMES core—A 14nm CMOS and PCM-based in-memory compute core using an array of 300ps/LSB linearized CCO-based ADCs and local digital processing," in *Proc. IEEE Symp. VLSI Technol.*, to be published.