

# Accurate Emulation of Memristive Crossbar Arrays for In-Memory Computing

Anastasios Petropoulos\*, Irem Boybat<sup>†‡</sup>, Manuel Le Gallo<sup>†</sup>, Evangelos Eleftheriou<sup>†</sup>,  
Abu Sebastian<sup>†</sup>, and Theodore Antonakopoulos\*

\*University of Patras, Dept. of ECE, 26504 Patras, Greece, Email: {a.petropoulos, antonako}@ece.upatras.gr

<sup>†</sup>IBM Research - Zurich, 8803 Rüschlikon, Switzerland, Email: {ibo, anu, ele, ase}@zurich.ibm.com

<sup>‡</sup>Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland

**Abstract**—In-memory computing is an emerging non-von Neumann computing paradigm where certain computational tasks are performed in memory by exploiting the physical attributes of the memory devices. Memristive devices such as phase-change memory (PCM), where information is stored in terms of their conductance levels, are especially well suited for in-memory computing. In particular, memristive devices, when organized in a crossbar configuration can be used to perform matrix-vector multiply operations by exploiting Kirchhoff's circuit laws. To explore the feasibility of such in-memory computing cores in applications such as deep learning as well as for system-level architectural exploration, it is highly desirable to develop an accurate hardware emulator that captures the key physical attributes of the memristive devices. Here, we present one such emulator for PCM and experimentally validate it using measurements from a PCM prototype chip. Moreover, we present an application of the emulator for neural network inference where our emulator can capture the conductance evolution of approximately 400,000 PCM devices remarkably well.

**Index Terms**—In-memory computing, neural networks, phase-change memory, hardware emulator

## I. INTRODUCTION

The explosive growth in data-centric artificial intelligence related applications has necessitated the exploration of non-von Neumann computing paradigms such as in-memory computing. In in-memory computing, the physical attributes of memory devices are exploited to perform computational tasks in place without the need to shuttle around data between the memory and the processing units [1], [2], [3], [4], [5], [6]. A new class of emerging memory devices known as resistive memory or memristive devices are particularly well suited for in-memory computing [7]. For example, the memristive devices, when organized in a crossbar configuration can be used to perform matrix-vector multiply operations. Here, the matrix elements are stored in terms of the conductance values of the memristive devices. By exploiting Ohm's law and Kirchhoff's current summation law, the matrix-vector multiply operation can be performed in constant time. This computational capability makes in-memory computing especially interesting for applications such as deep learning training and inference, where cascaded stages of matrix-vector multiplications form the bulk of computation [8], [9], [10]. The forward propagation (inference) stage, as well as the backpropagation, can be realized by merely reading the array.

In spite of the promise of in-memory computing for applications such as deep learning, several open questions need to be

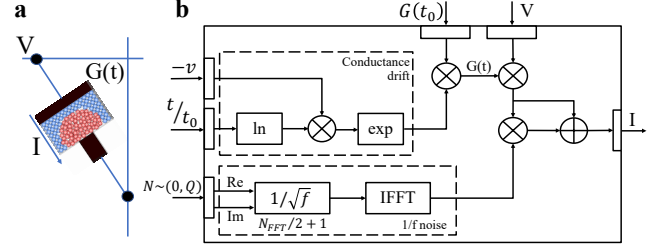


Fig. 1: (a) Schematic illustration of a PCM device with a mushroom-type device geometry. (b) Corresponding PCM cell model used for the FPGA-based emulator design.

addressed. First, it is essential to understand the computational reliability and accuracy of memristive in-memory cores for a range of applications since memristive devices exhibit non-idealities, such as temporal variations of conductance values. Identifying desired device characteristics for target applications can provide useful insight into future device designs. Furthermore, it is critical to develop efficient system architectures that involve cascaded memristive in-memory cores for applications such as deep learning. Note that compared to all-digital implementations, in-memory computing is more amenable for highly pipelined dataflows. Finally, it is of significant importance to develop a versatile software stack that can map the applications to the multi-core in-memory computing hardware. An accurate and fast hardware emulator of memristive devices and computing cores will be an indispensable tool to address all of these goals. In comparison to a software simulator, a custom-designed hardware counterpart can perform the prototyping of an in-memory computing core in a more rapid manner.

An FPGA-based hardware emulator for PCM arrays, which can mimic the temporal conductance evolution of PCM, has been previously demonstrated in [11]. The system is shown to perform a matrix-vector multiplication on a 256x256 emulated array in only 136.16 microseconds. However, functional verification with experimental data has not been demonstrated yet. In this paper, we show for the first time an FPGA-based hardware emulator that can reliably capture experimental PCM characteristics. In Section II, we present the emulation of single PCM devices in an FPGA where we capture the key physical attributes such as conductance drift and  $1/f$  noise. We validate the emulator using experimental measurements

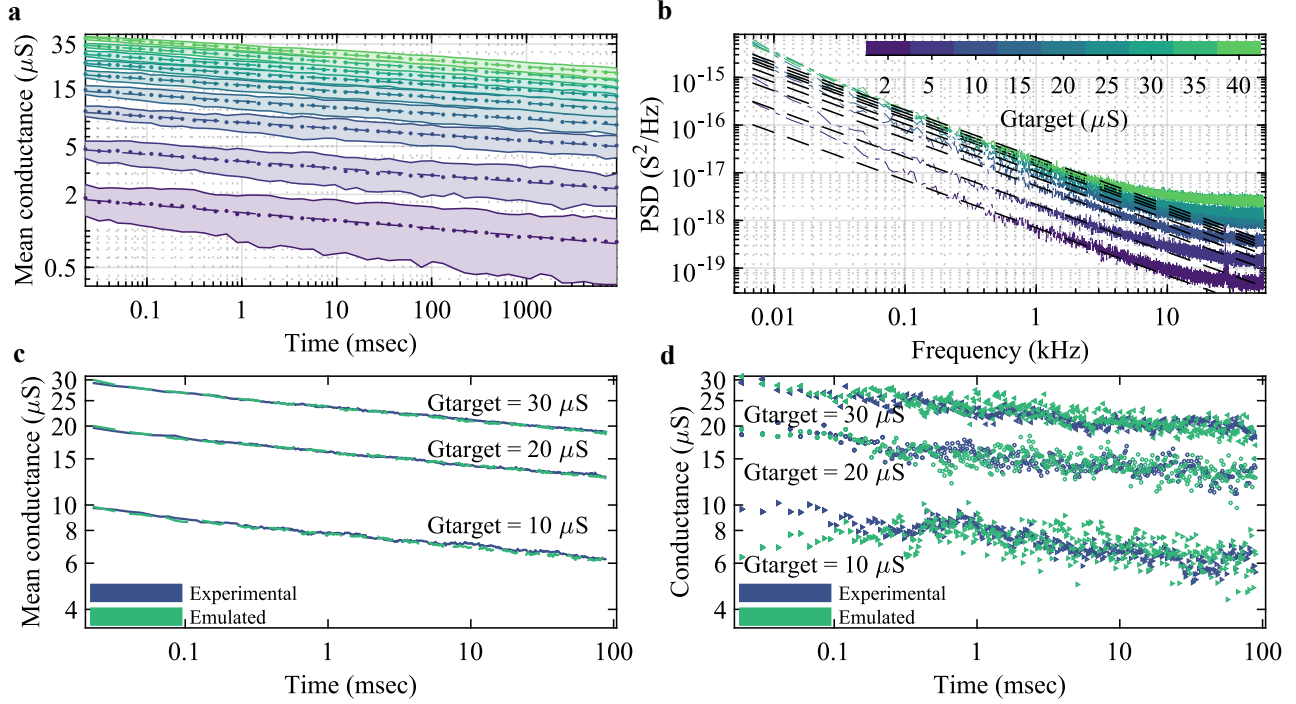


Fig. 2: Experimental measurements on mushroom-type PCM devices fabricated in 90 nm technology node. (a) Mean conductance evolution of 100 devices programmed to different target conductances and the corresponding linear fits according to Eq. (1). The shades denote one standard deviation. (b) Power spectral density (PSD) of the conductance signals for each target conductance level and the corresponding fit according to Eq. (2). (c) Emulation of the mean conductance evolution of 100 devices programmed to three different target conductance levels. (d) Emulation of the conductance evolution of 3 PCM devices.

from 100 devices from a prototype PCM chip programmed to various conductance levels. In Section III, we present how a PCM multi-cell crossbar array emulation can be constructed. Finally, in section IV, we illustrate the application of the PCM crossbar emulator for neural network inference, and we validate our results with an experiment involving approximately 400,000 PCM devices.

## II. PCM CELL EMULATION

PCM is arguably the most advanced resistive memory technology and has been widely employed for in-memory computing [12], [13], [14], [15], [16]. PCM exploits the behavior of certain phase-change materials such as  $\text{Ge}_2\text{Sb}_2\text{Te}_5$  that can be switched reversibly between amorphous and crystalline phases of different electrical resistivity. A PCM device consists of a certain volume of this phase-change material sandwiched between two electrodes (see Fig. 1a). By applying suitable electrical pulses, referred to as programming pulses, it is possible to alter the phase configuration within the PCM device and achieve different conductance values. By iterative programming schemes comprising multiple program-and-verify steps, it is possible to obtain any desired conductance value within a certain error margin [17]. However, the programmed conductance values exhibit temporal variations such as drift, which is attributed to the structural relaxation of the unstable amorphous phase [18], and  $1/f$  noise. These temporal variations are shown to be detrimental for PCM-based implementations [19] and hence need to be well captured by a PCM cell emulator.

As shown in Fig. 1b, the PCM cell emulator consists of two functional modules, one for the conductance drift and one for the  $1/f$  noise. The drift is modeled according to the following power-law equation and can be rearranged for ease of hardware implementation.

$$G(t) = G(t_0) \left( \frac{t}{t_0} \right)^{-\nu} = G(t_0) \exp \left( -\nu \ln \left( \frac{t}{t_0} \right) \right) \quad (1)$$

In Eq. (1),  $G(t)$  denotes the conductance value at time instance  $t$ ,  $G(t_0)$  denotes the conductance at time  $t_0$ , and  $\nu$  is the drift exponent. For 90 nm doped GST devices,  $\nu$  is reported to take values between 0.03 and 0.1, depending on the initial amorphous volume created with the programming pulse [20], [21]. There is also variability associated with  $\nu$  [20], [22], [23]. To capture these observations, we sample the drift exponent of individual devices from a Gaussian distribution with a certain mean and standard deviation. For the  $1/f$  noise module, a hardware block was designed in order to implement the equation:

$$S_{I_{\text{noise}}}(f) = I_{\text{read}}^2 Q \frac{1}{f} \quad (2)$$

$S_{I_{\text{noise}}}(f)$  denotes the power spectral density associated with the read noise [24].  $I_{\text{read}}$  denotes the mean read current when biased by the read voltage,  $V$ . As shown in Fig. 1b, we generate two independent and normally distributed random vectors with a dimension of  $N_{FFT}/2+1$ , with known variance,  $Q$ , and zero mean, and we use them as a complex Gaussian random vector in the frequency domain. The amplitude of the complex

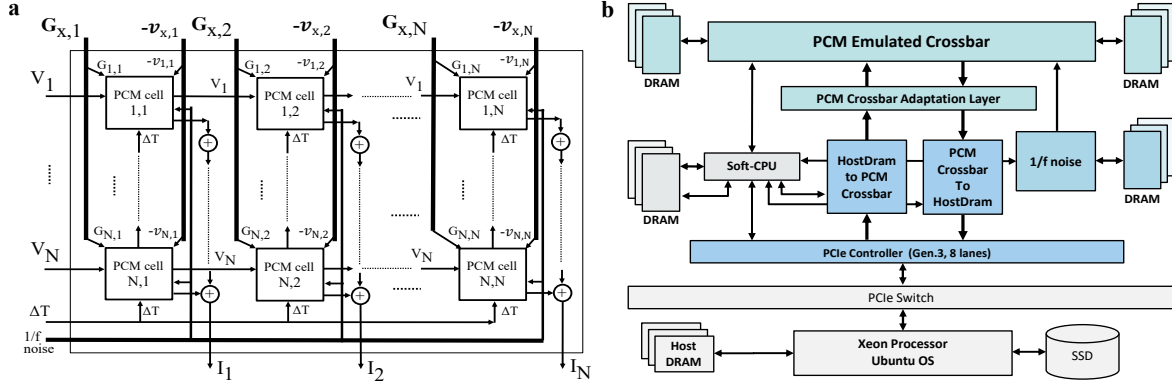


Fig. 3: The PCM crossbar emulator. (a) Functional diagram [11]. (b) Hardware diagram of the PCIe-based FPGA system.

Gaussian random vector is scaled by the  $1/\sqrt{f}$  factor. Then the negative frequency spectral samples are determined to satisfy for Hermitian symmetry. Finally, the inverse Fourier transform ( $N_{FFT}$  points) is applied for generating a real-valued time series with the desired noise characteristics [25].

The underlying functions associated with the drift and noise functional modules were implemented with the utilization of floating-point cores that integrate DSP slices, which are pipelined for achieving high throughput, if a large number of cells has to be emulated. With this model, we can investigate the influence of drift and  $1/f$  noise on scalar multiplication.

For the experimental validation of the PCM cell emulator, we used an experimental platform with 1 million mushroom-type PCM cells, with doped  $\text{Ge}_2\text{Sb}_2\text{Te}_5$  as the phase-change material, and fabricated in 90 nm CMOS technology. In order to verify our PCM cell emulator for different conductance levels, we programmed 100 devices to a range of  $2 \mu\text{S}$  to  $40 \mu\text{S}$  using iterative programming. Subsequently, the read current from each device measured for approximately 9 seconds with a sampling rate of 112 kHz. The device conductances were estimated based on the read voltage of  $V = 0.2 \text{ V}$ .

Fig. 2a shows the evolution of PCM conductance states. It can be seen that the mean behavior matches the relationship predicted by Eq. (1). For each targeted conductance level, a line is fitted on the average conductance evolution. The drift exponent is calculated from the slope of this linear fit. Note that  $\nu$  depends on the initially created amorphous volume. We assume a constant standard deviation of 0.02 for  $\nu$ . To estimate the noise and its power spectral density, we used the read measurements obtained during the last second. The reason for this is to decouple the effect of drift from the  $1/f$  noise measurement as drift slows down significantly with time. Fig. 2b presents the PSD of the  $1/f$  noise and the corresponding fitting curves with respect to Eq. (2) for different target conductance levels. Based on these measurements, it is observed that  $Q$  becomes higher as the target conductance level becomes lower, also reported by [23], [26]. The observed values of  $Q$  were from  $5.1 \times 10^{-5}$  to  $1.1 \times 10^{-3}$ . Typically, the noise in PCM follows a  $1/f^\gamma$  relationship, where  $\gamma$  is reported to be within the range of 0.9-1.1 [27]. The deviation from the  $1/f$  behavior is also evident in Fig. 2b. However, for modeling simplicity, we assume an ideal  $1/f$  relationship, where we use  $N_{FFT} = 1024$

points for the inverse Fourier transform in the emulator. The extracted experimental parameters were used in the PCM cell emulator, and the device behavior was emulated. As shown in Fig. 2c and Fig. 2d, the emulated conductance evolution over time matches the experimental behavior remarkably well. Both the mean conductance behavior and individual device evolution are faithfully captured.

### III. PCM CROSSBAR EMULATION

The PCM cell emulation model was utilized for emulating a multi-cell PCM crossbar architecture, as shown in Fig. 3a. The depicted architecture uses  $N \times N$  PCM emulated cells, where each one has its own conductance and drift exponent model parameter. Also, all cells are supplied with samples from the same  $1/f$  noise generator, but each emulated cell is fed with a different instantiation of noise samples.

A sequential execution of  $N$  dot-products in the emulator can be used to simulate the matrix-vector multiplication of a  $N \times N$  crossbar in hardware. For our scenario, each element-wise multiplication of the dot-product is performed by one emulated PCM cell. The element-wise multiplication can be used to implement a  $k$ -element dot-product in a column of the crossbar, where the  $k$ -factor depends on the available hardware resources. Thus, a dot-product with a dimension greater than the  $k$ -factor can be achieved by executing its operation several times with the addition of the partial results using a tree structure of adders and an accumulator [11], [28].

The components of the system, which implements the emulated crossbar design, are shown in Fig. 3b. The emulated PCM crossbar consists of two dedicated DRAM memories for storing the crossbar conductances and drift coefficients, while another DRAM is used for storing pre-generated  $1/f$  noise samples. Two dedicated data mover engines (HostDRAM - PCM Crossbar) are used for high speed (8 GBps) transfers of data between the crossbar and the server's memory. A PCM crossbar adaptation layer is used for encoding weights to conductances, transforming vector data to voltage values and also decoding the resulted current values. Also, it contains nonlinearity functional blocks (i.e., RELU, sigmoid, tanh) for neural network applications. In addition, the system incorporates a soft-CPU for initialization and control. The soft-CPU interacts with a host application using a dedicated device driver

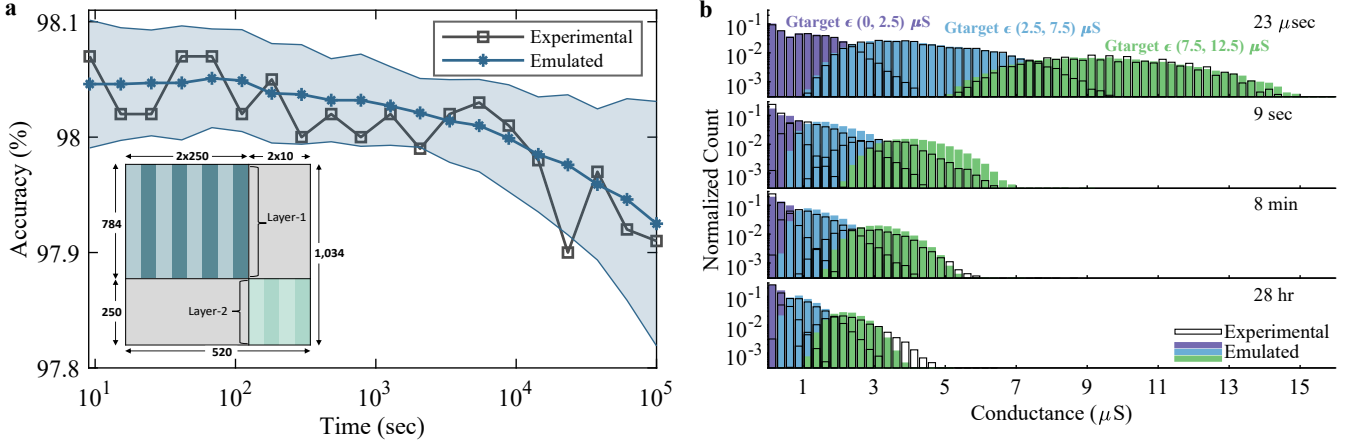


Fig. 4: Neural network inference results. (a) Evolution of accuracy over time from PCM experimental results compared with the mean behavior of emulated results and their shaded region representing one standard deviation. The inset presents the mapping scheme of the network's layers in a single emulated crossbar. (b) Experimental and emulated conductance distribution of the neural network's encoded weights for three different conductance ranges.

and descriptor-structured data transfers. The PCM crossbar emulator has been implemented on a Kintex UltraScale FPGA and has been tested on a high-end Xeon server.

For a matrix-vector multiplication scenario, the host initiates requests for downloading, conductance encoding, and conductance data storing to the FPGA's DRAM memories. Also, it initializes a dedicated DRAM area with  $1/f$  noise samples. In order to start the emulated crossbar for matrix-vector operations, three main procedures are performed: (a) vector data are received through the host, passing by the PCM crossbar adaptation layer and transformed to voltages, (b) concurrently, the conductance matrix is loaded from the DRAMs to the crossbar along with  $1/f$  noise values, and the computation is started, (c) finally, the resulted currents are processed by the adaptation layer and then uploaded to the host's DRAM. Such a versatile process can operate in a pipelined fashion using several crossbars.

#### IV. NEURAL NETWORK INFERENCE USING THE EMULATED PCM CROSSBAR

For the evaluation of the emulated PCM crossbar, we considered the task of MNIST handwritten digit recognition. For that purpose, we compared emulated, and experimental inference results from PCM arrays over time for a fully-connected neural network with two layers. The network dimensions are 784-250-10, and it was trained in software using single-precision floating-point weights. Next, the trained weights were iteratively programmed to conductance values on the PCM prototype chip, utilizing approximately 400,000 PCM devices. These weights are linearly mapped to conductance values. A differential PCM configuration is used for each synapse where one device denotes the positive part of the weight, and the other device denotes the negative part of the weight. According to the sign of the weight, one device of each differential pair is set close to  $0 \mu S$ . We use the programmed conductance values of PCM devices at  $23 \mu sec$  as the emulator's initial state [10]. The subsequent conductance values of later time steps are determined with model parameters (i.e.,

drift exponent,  $1/f$  variance  $Q$  factor). For simplicity, we adopt the parameters used to emulate the behavior of devices with target conductance of  $5 \mu S$  in Section II ( $\bar{\nu} = 0.06$ ,  $\sigma_{\nu} = 0.02$ ,  $Q = 4 \times 10^{-4}$ ). Note that the target conductances representing the network weights are mostly contained within the range of 0 to  $5 \mu S$ .

In Fig. 4a, we present accuracy results of neural network inference for a time period greater than 27 hours. The evolution of the mean accuracy over time from the experiment is well captured by the emulator. Additionally, to further verify our model regarding the conductance drift and noise, we show the evolution of the network's weight distribution encoded to conductances. As depicted in Fig. 4b, the emulated results capture well the temporal evolution of the conductance distributions for different target conductance states.

For this inference application, both weight layers of the neural network were emulated in a single crossbar in a pipelined fashion. This is achieved by using a crossbar size of  $1034 \times 520$ , to fit both layer dimensions, with redundant cells (zero conductance) in appropriate places of the weights' matrix (see Fig. 4a inset). With this mapping approach, our emulator achieves a processing rate of 8.8 kilo-images per second and  $227 \mu sec$  latency.

#### V. CONCLUSION

In this work, we presented an accurate FPGA-based hardware emulator for phase-change memory that captures the key physical attributes such as temporal drift of conductance values as well as  $1/f$  noise. The PCM cell emulator and its extension to the PCM crossbar emulator were experimentally validated using a prototype PCM array based on a deep learning inference hardware experiment that involves approximately 400,000 PCM devices. The presented hardware emulator can be a powerful tool for the exploration of in-memory computing and its applications. This approach is scalable to larger networks and more complex problems, while the application domain is not restricted to neural network inference as this emulator can benefit other in-memory computing scenarios.



## REFERENCES

- [1] A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, and E. Eleftheriou, "Temporal correlation detection using computational phase-change memory," *Nature Communications*, vol. 8, no. 1, p. 1115, 2017.
- [2] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, p. 333, 2018.
- [3] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019.
- [4] A. Serb, J. Bill, A. Khat, R. Berdan, R. Legenstein, and T. Prodromakis, "Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses," *Nature communications*, vol. 7, p. 12611, 2016.
- [5] S. Yu, "Neuro-inspired computing with emerging nonvolatile memories," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, 2018.
- [6] I. Vourkas and G. C. Sirakoulis, "Emerging memristor-based logic circuit design approaches: A review," *IEEE Circuits and Systems Magazine*, vol. 16, no. 3, pp. 15–30, 2016.
- [7] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.
- [8] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti *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 Transactions on Electron Devices*, vol. 62, no. 11, pp. 3498–3507, 2015.
- [9] Z. Wang, S. Joshi, S. Savelev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, Y. Zhuo *et al.*, "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electronics*, vol. 1, no. 2, p. 137, 2018.
- [10] A. Sebastian, I. Boybat, M. Dazzi, I. Giannopoulos, V. Jonnalagadda, V. Joshi, G. Karunaratne, B. Kersting, R. Khaddam-Aljameh, S. Nandakumar *et al.*, "Computational memory-based inference and training of deep neural networks," in *2019 Symposium on VLSI Technology*. IEEE, 2019, pp. T168–T169.
- [11] A. Petropoulos and T. Antonakopoulos, "Accurate PCM crosspoint emulator and its use on eigenvalues calculation," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2018, pp. 549–552.
- [12] G. W. Burr, M. J. Brightsky, A. Sebastian, H.-Y. Cheng, J.-Y. Wu, S. Kim, N. E. Sosa, N. Papandreou, H.-L. Lung, H. Pozidis *et al.*, "Recent progress in phase-change memory technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 146–162, 2016.
- [13] A. Sebastian, M. Le Gallo, and E. Eleftheriou, "Computational phase-change memory: beyond von Neumann computing," *Journal of Physics D: Applied Physics*, vol. 52, no. 44, p. 443002, 2019.
- [14] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *Nature Electronics*, vol. 1, no. 4, p. 246, 2018.
- [15] I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," *Nature communications*, vol. 9, no. 1, p. 2514, 2018.
- [16] S. 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 *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.
- [17] N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, and E. Eleftheriou, "Programming algorithms for multilevel phase-change memory," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. IEEE, 2011, pp. 329–332.
- [18] M. Le Gallo, D. Krebs, F. Zipoli, M. Salinga, and A. Sebastian, "Collective structural relaxation in phase-change memory devices," *Advanced Electronic Materials*, vol. 4, no. 9, p. 1700627, 2018.
- [19] V. Joshi, M. L. Gallo, I. Boybat, S. Haefeli, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Accurate deep neural network inference using computational phase-change memory," *arXiv preprint arXiv:1906.03138*, 2019.
- [20] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed sensing recovery using computational memory," in *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2017, pp. 28–3.
- [21] I. Boybat, S. Nandakumar, M. Le Gallo, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Impact of conductance drift on multi-pcm synaptic architectures," in *2018 Non-Volatile Memory Technology Symposium (NVMTS)*. IEEE, 2018, pp. 1–4.
- [22] M. Boniardi, D. Ielmini, S. Lavizzari, A. L. Lacaita, A. Redaelli, and A. Pirovano, "Statistics of resistance drift due to structural relaxation in phase-change memory arrays," *IEEE Transactions on Electron Devices*, vol. 57, no. 10, pp. 2690–2696, 2010.
- [23] S. Nandakumar, I. Boybat, V. Joshi, C. Piveteau, M. Le Gallo, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Phase-change memory models for deep learning training and inference," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2019, pp. 727–730.
- [24] G. Close, U. Frey, M. Breitwisch, H. Lung, C. Lam, C. Hagleitner, and E. Eleftheriou, "Device, circuit and system-level analysis of noise in multi-bit phase-change memory," in *2010 International Electron Devices Meeting*. IEEE, 2010, pp. 29–5.
- [25] J. Timmer and M. König, "On generating power law noise," *Astronomy and Astrophysics*, vol. 300, p. 707, 1995.
- [26] P. Fantini, A. Pirovano, D. Ventrice, and A. Redaelli, "Experimental investigation of transport properties in chalcogenide materials through 1/f noise measurements," *Applied physics letters*, vol. 88, no. 26, p. 263506, 2006.
- [27] P. Fantini, G. B. Beneventi, A. Calderoni, L. Larcher, P. Pavan, and F. Pellizzer, "Characterization and modelling of low-frequency noise in pcm devices," in *2008 IEEE International Electron Devices Meeting*. IEEE, 2008, pp. 1–4.
- [28] A. Petropoulos and T. Antonakopoulos, "A versatile PCM-based circuits emulator and its use on implementing linear algebra functions," in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 167–171.