Combined HW/SW Drift and Variability Mitigation for PCM-based Analog In-memory Computing for Neural Network Applications

Combined HW/SW Drift and Variability Mitigation for PCM-Based Analog In-Memory Computing for Neural Network Applications

Alessio Antolini, *Graduate Student Member, IEEE*, Carmine Paolino, *Graduate Student Member, IEEE*, Francesco Zavalloni, Andrea Lico, *Graduate Student Member, IEEE*, Eleonora Franchi Scarselli, *Member, IEEE*, Mauro Mangia, *Member, IEEE*, Fabio Pareschi, *Senior Member, IEEE*, Gianluca Setti, *Fellow, IEEE*, Riccardo Rovatti, *Fellow, IEEE*, Mattia Luigi Torres, Marcella Carissimi, and Marco Pasotti

***Abstract*— Matrix-Vector Multiplications (MVMs) represent a heavy workload for both training and inference in Deep Neural Networks (DNNs) applications. Analog In-memory Computing (AIMC) systems based on Phase Change Memory (PCM) has been shown to be a valid competitor to enhance the energy efficiency of DNN accelerators. Although DNNs are quite resilient to computation inaccuracies, PCM non-idealities could strongly affect MVM operations precision, and thus the accuracy of DNNs. In this paper, a combined hardware and software solution to mitigate the impact of PCM non-idealities is presented. The drift of PCM cells conductance is compensated at the circuit level through the introduction of a conductance ratio at the core of the MVM computation. A model of the behaviour of PCM cells is employed to develop a device-aware training for DNNs and the accuracy is estimated in a CIFAR-10 classification task. This work is supported by a PCM-based AIMC prototype, designed in a 90-nm STMicroelectronics technology, and conceived to perform Multiply-and-Accumulate (MAC) computations, which are the kernel of MVMs. Results show that the MAC computation accuracy is around 95% even under the effect of cells drift. The use of a device-aware DNN training makes the networks less sensitive to weight variability, with a 15% increase in classification accuracy over a conventionally-trained Lenet-5 DNN, and a 36% gain when drift compensation is applied.**

Manuscript received 15 October 2022; revised 21 December 2022; accepted 9 January 2023. Date of publication 2 February 2023; date of current version 20 March 2023. This work was supported in part by the Electronics Components and Systems for European Leadership (ECSEL) Joint Undertaking (JU) under Grant 101007321; and in part by JU through the European Union's Horizon 2020 Research and Innovation Programme and France, Belgium, Czech Republic, Germany, Italy, Sweden, Switzerland, and Turkey. This article was recommended by Guest Editor J. L. Rossello. *(Corresponding authors: Alessio Antolini; Carmine Paolino.)*

Alessio Antolini, Francesco Zavalloni, Andrea Lico, Eleonora Franchi Scarselli, Mauro Mangia, and Riccardo Rovatti are with the "'Ercole De Castro" Research Centre on Electronic Systems for Information and Communication Technologies and the Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi," University of Bologna, 40126 Bologna, Italy (e-mail: alessio.antolini2@unibo.it).

Carmine Paolino and Fabio Pareschi are with the Department of Electronics and Telecommunications, Politecnico of Torino, 10129 Turin, Italy (e-mail: carmine.paolino@polito.it).

Gianluca Setti is with CEMSE, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia.

Mattia Luigi Torres, Marcella Carissimi, and Marco Pasotti are with STMicroelectronics, 20864 Agrate Brianza, Italy.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2023.3241750.

Digital Object Identifier 10.1109/JETCAS.2023.3241750

***Index Terms*— Analog in-memory computing (AIMC), phase-change memory (PCM), deep neural network (DNN), noise-aware training, drift compensation.**

I. Introduction

IN THE era of big data a plethora of applications require low-power-consumption computations involving large amount of information [1], [2]. In this scenario, the performance of conventional digital computers is limited by the intrinsic communication bottleneck of the Von Neumann architecture, which needs data to be moved back and forth between the memory and the processing unit [3], [4]. In recent years, novel computational approaches have been investigated to overcome this limitation. Among those, Analog In-memory computing (AIMC) based on resistive memory devices has proven to be a promising non-Von Neumann solution for the fast and energy-efficient execution of Matrix-Vector Multiplication (MVM) [5]. As an example, MVMs represent a heavy workload for both training and inference in deep learning applications, and being able to perform them at O(1) time complexity through AIMC solutions would lead to huge benefits.

The goal of AIMC is to perform computations within the memory unit, typically leveraging the physical properties of the memory devices themselves, and taking advantage of Ohm's and Kirchhoff's laws [4], [6], [7]. Among the technologies that have been considered for AIMC, Phase-Change Memory (PCM) is one of the most promising due to its long-term storage of multi-bit quantities and its compatibility with the traditional CMOS fabrication processes [8], [9]. However, due to the peculiarities of the technology, only limited computation precision can be achieved. This, in turn, affects the accuracy of the analog accelerators and thus the performance at the application level.

The main issues associated to PCM devices are: i) the non linearity of the I-V characteristic; ii) the random conductance drift over time; and iii) the variability of the programmed cell states [10]. The I-V non linearity is typically addressed by encoding input values in time as the width of a constant-voltage signal applied across each cell of the array

[11], [12], [13], [14], [15], [16]. Conductance drift over time is usually compensated by post-processing techniques [13], [17] or employing HW-aware training solutions [18]. Finally, state-of-art iterative program-and-verify algorithms [19] can be used to reduce, but not completely remove, the variability in the state of identically programmed cells. Therefore, device-to-device variability, due to both drift and programming state, is still a widely investigated subject that limits the accuracy of PCM-based systems.

A deeper look at solutions applied to limit the effects of conductance drift in AIMC applications highlights the possibilities available at the technological, circuit and software level.

In [20] an investigation on two back end fabrication processes of Ge-rich GeSbTe (Ge-GST) PCM cells is presented along with their influence on devices electrical characteristics, whose empirical model is discussed in [21].

Circuit solutions are analyzed in [13] and [22]. In the former, each analog weight matrix is extended, as time progresses, by the introduction of additional columns (i.e., neurons) to account for the lower dynamic range of the MVM output as conductances become progressively smaller. In the latter, conversely, it is observed that the typical implementation of negative weights with positive-only conductance, i.e. having a second analog array whose output is subtracted from the first, already leads to some measure of drift compensation. Again, the dynamic range of the output is shrinked, thus requiring a renormalization to preserve performance. The renormalization proposed therein requires an additional array of PCM cells to estimate the drift of the SET state conductance (for binary-level applications, i.e. only using cells in the SET and RESET state).

Finally, solutions can be applied at the software level or in any case in the digital section of the processing chain. Authors in [23] define an ad-hoc regularization function applied during the NN training to limit the variability observed at the neuron level resulting from perturbations of the individual conductances. In [24] drift is addressed by renormalizing the drifted MVM output by modeling the median conductance decay and rescaling the argument of the nonlinear activation function following each layer to ensure that the entire nonlinearity domain is excited as expected for non-drifting weights. In [18] a periodic calibration procedure is used to update the parameters of the batch normalization layers, so that even when weights start to drift, those layers can still remap their outputs to zero-mean, unit-variance distributions.

Obviously, each technique comes with its own set of drawbacks, i.e. requiring a different fabrication process technology [25], a considerable area overhead associated to the AIMC unit [13], [22], reliance on accurate device models [24] or the periodic recalibration of the system [18]. By applying multiple techniques simultaneously the requirements on each of them can be relaxed, with potential reduction of the incurred cost.

This paper analyzes a circuit-level solution to mitigate the impact of the conductance drift on the inference task of Deep Neural Networks (DNNs), employing PCM devices as the core synaptic model. The technique being used in this work is potentially compatible with many of the methodologies previously described and could lead to even better performance in their simultaneous application.

This work is supported by empirical results obtained with an AIMC unit testchip [26] designed to implement Multiply-and-Accumulate (MAC) operations, i.e., the core of MVMs. The prototype, implemented in a 90-nm STMicroelectronics CMOS technology, includes the AIMC unit and a PCM IP, and it is conceived to extend the features of the memory IP without modifying its internal structure. The hardware architecture reduces the effects of PCM cells conductance drift on MAC operations at the circuit level with negligible area overhead, so that no post-processing conditioning is required; moreover, non-linearities in the device I-V characteristics are not excited. The conductance variability is then addressed in the context of a DNN-based classification task, by implementing the DNN training phase in a device-aware fashion, with measurements-based models. The benefits of the proposed solutions are evaluated on the CIFAR-10 dataset, employing two neural networks having significantly different complexities, i.e., Lenet-5 and VGG-8.

The paper is organized as follows. In Section II the architecture and the implementation of the employed testchip are recalled. In Section III the experimental results of hardware drift-compensation and accuracy are expanded and discussed. In Section IV numerical models are fit against programming variability and drift measurements. The combination of hardware compensation and device-aware training procedures is validated in Section V on the CIFAR-10 classification task. Finally, the conclusions are drawn.

II. AIMC Prototype

This section briefly describes the architecture of the AIMC prototype proposed in [26]. The system is used to execute one-step Multiply-and-Accumulate (MAC) operations with both signed inputs and coefficients. The prototype, depicted in Fig. 1(a), contains a peripheral AIMC unit interfaced with an embedded PCM (ePCM) IP [27]; the whole testchip is manufactured in a 90-nm STMicroelectronics CMOS technology, which features a specifically optimized Ge-rich GeSbTe (GST) alloy for PCM cells. The AIMC unit is directly connected to the Main BitLines (MBL) of the ePCM IP (Fig. 1(b)), and extends its functionalities by adding MAC execution features, while preserving its use as a standard binary ePCM memory array.

A. Computing Architecture

To perform MAC tasks, the AIMC unit sets the read voltage of each MBL and reads the current of the cells belonging to the addressed WordLine (WL). Therefore, the computation of a MVM requires multiple sequential activations of different WLs. The proposed implementation of the AIMC unit architecture has been conceived to mitigate part of the aforementioned PCM non-idealities cells. In particular, the adoption of time-coded inputs, along with cells being read at a fixed voltage, address the issue of a nonlinear device I-V characteristic. In addition, as explained in detail

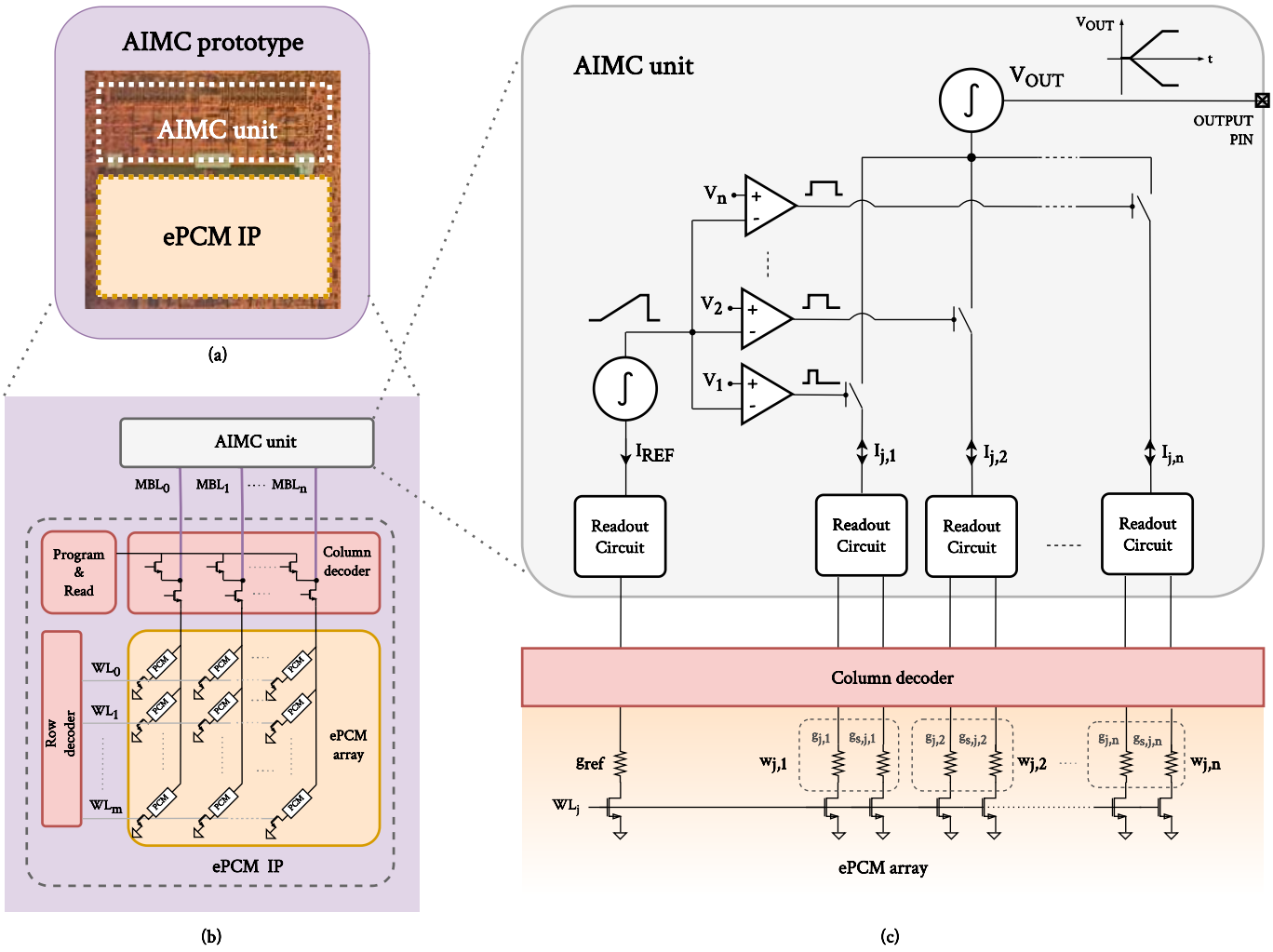

Fig. 1. (a) Die micrograph. The AIMC unit is designed on the top of the ePCM IP. (b) Structure of the AIMC prototype: the AIMC unit is interfaced with the memory array through its MBLs; the programming and read circuits, as well as the row and column decoders are not modified to grant both AIMC and standard digital employments. (c) Structure of the AIMC unit architecture, where the MAC computation principle is shown.

in Section III-C, PCM cells drift compensation is achieved by a hardware technique that makes the MAC output dependent on a conductance ratio, instead of an absolute conductance value.

Let us first consider the mathematical description of a MVM, i.e., $\mathbf{z} = \mathbf{W}\mathbf{x}$, with $\mathbf{z} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{W} \in \mathbb{R}^{m \times n}$. The j-th element of $\mathbf{z}$, is indeed computed through the MAC operation:

$$z_j = \mathbf{W}_j \cdot \mathbf{x} = \sum_{i=1}^{n} w_{j,i} x_i \,, \tag{1}$$

where $\mathbf{W}_j$ is the j-th row of $\mathbf{W}$. Each element $w_{j,i}$ of $\mathbf{W}_j = [w_{j,1}, \ldots, w_{j,n}]$ is expressed through a conductance $g_{j,i}$ for its magnitude, and through $g_{S,j,i}$ for its sign, each stored in a single PCM cell of the j-th wordline; thus, from a functional point of view, the implementation of each weight $w_{j,i}$ is:

$$w_{j,i} = \begin{cases} g_{j,i} & \text{if } g_{S,j,i} < g_{\text{th}} \\ -g_{j,i} & \text{if } g_{S,j,i} \geq g_{\text{th}} \end{cases} \tag{2}$$

where g_{th} is the conductance threshold to encode a positive or negative weight sign. The actual details of the device-level implementation can be found in [26]. Fig. 1(c) shows the detailed structure of the AIMC unit designed to perform a single signed MAC operation.

Each element x_i of the input array $x = [x_1, \ldots, x_n]$ is represented by the $V_1, \ldots, V_n$ analog voltages. The corresponding time-coded version is obtained by comparing each V_i, with a shared ramp signal generated by the integration of a reference current I_{REF}, flowing through a reference cell g_{REF} having the constant V_{REF} voltage across. The constant read voltage V_{REF} is also applied across each weight cell $g_{j,i}$ obtaining the $I_{j,i}$ current. The MBL voltage is forced to V_{REF} by the Readout Circuits [26], which are interfaced with each MBL of the ePCM IP. The weighted summation is then obtained integrating over a fixed time the sum of $I_{j,i}$, each having a duration proportional to x_i. The expression of the differential output V_{OUT_j}, associated to the j-th wordline is:

$$V_{\text{OUT}_j} = k \left[\sum_{i=1}^{n} \left(\pm \frac{g_{j,i}}{g_{\text{REF}}} V_i \right) \right] \tag{3}$$

where k is a dimensionless constant value accounting for the effects of circuit parameters. The sign of the elementary product, which corresponds to the direction of the $I_{j,i}$ current being integrated, is managed by the Readout Circuit according to the sign of the original product $w_{j,i}x_i$ [26]. Considering V_i and $g_{j,i}/g_{\mathrm{REF}}$ as the absolute values of x_i and $w_{j,i}$, respectively, $V_{\mathrm{OUT}_j} = k\sum_i w_{j,i}x_i = kz_j$ is therefore proportional to the ideal signed MAC result z_j already expressed in (1).

This result, obtained with PCM devices being read at a constant voltage, does not suffer from memory cells nonlinearity. The adoption of ramp-driven time-coded inputs is indeed a common solution to overcome the behavior of PCM devices I-V characteristic [11], [12], [13], [14], [15]. Though, the key-feature of the proposed architecture lies in the introduction of a conductance ratio in implementing the MAC weights, which is beneficial to mitigate the effects of cells drift on the MAC results. It has been proven [28] that the drift of a generic cell conductance $g(t)$ follows the power law $g(t) = g_0(t/t_0)^{-\alpha}$, where g_0 is the conductance at an arbitrary initial time t_0, and α is the drift coefficient, which is positive and cell-to-cell variable. Being the MAC result proportional to the conductance ratio $g_{j,i}(t)/g_{\mathrm{REF}}(t)$, and plugging the drift model of $g(t)$ in (3), the MAC operation evaluated at time $t_0 + t$, becomes:

$$V_{\mathrm{OUT}_j}(t) = k\left[\sum_{i=1}^{n}\left(\pm\frac{g_{0,j,i}}{g_{0,\mathrm{REF}}}\left(\frac{t}{t_0}\right)^{-(\alpha_{j,i}-\alpha_{\mathrm{REF}})}V_i\right)\right] \quad (4)$$

where $g_{0,j,i}$ and $g_{0,\mathrm{REF}}$ are the conductances of the weight and reference cells at t_0, respectively. Here time t has to be intended as significantly longer than an actual integration window, wherein conductances are assumed to be constant. The notable aspect of (4) is the effective reduction of the drift coefficient to $(\alpha_{j,i} - \alpha_{\mathrm{REF}})$, so that drift is partially compensated. At the circuit level, the compensation mechanism acts on the slope of the shared ramp signal which decreases as g_{REF} experiences drift. This leads to an increase in the integration time and compensates for the drift-induced drop of the weight cells currents. To implement this, the architecture exploits only a single PCM cell to generate a reference ramp signal [26]. In case no compensation is adopted, the reference ramp still must be generated through a reference conductance. Therefore, the overhead due to the drift compensation consists only of the use of one PCM cell per WL, which can not be consequently used to store MAC weights. The fact of having the MAC output depend on the ratio of two correlated quantities, i.e. PCM conductances, leads to a general resilience of the system towards variabilities.

B. Testchip Implementation

The testchip prototype [26] allows for $n = 12$, i.e., twelve inputs and coefficients. Its circuits have been designed for a $V_{\mathrm{DD}} = 1.2$ V power-supply and a read voltage $V_{\mathrm{REF}} = 0.3$ V, leading to PCM cells currents ranging from hundreds of nA to tens of μA. The result of each MAC operation was obtained measuring the voltage V_{OUT_j} on an output pin. Experimental data were converted using a 16-bit ADC available on a dedicated evaluation board.

The design and implementation of this prototype are mainly focused on demonstrating the validity of its hardware drift compensation feature, and no specific strategy has been adopted to enhance the computing power consumption. Although the energy required to implement in-memory inference tasks depends upon the interconnection scheme of the network layers [2], [29], theoretical energy efficiency estimation of the architecture can be still provided. In this scenario, a MAC operation of size $n = 12$ is estimated to be implemented with a $\sim 0.34\,\mathrm{TOPS\,W^{-1}}$ energy efficiency, which is approximately two orders of magnitude lower than the state of the art [12], [18], [30], [31], [32], but still improvable in future designs based on this first prototype. In particular, the power consumption is firstly dominated by the PCM cells current. As in more competitive solutions [12], [18], cells should be read at lower BL voltages, as well as programmed at lower conductances. A considerable portion of power consumption is ascribed to the comparators too, which have been designed to get high switching speed, so that the MAC computation is more precise. A different strategy to encode the inputs may be considered (e.g., with fully-digital solutions [33]). Finally, the high power consumption associated to the output integrator is due to the large amount of cells currents being integrated on its feedback capacitance [26]. Alternative solutions to the current integrator, as for example a direct current-to-digital conversion [12], must be taken in consideration.

III. Experimental Evaluation of the AIMC Unit

In this section, the performance of the AIMC unit is observed in terms of peripheral-circuit accuracy and of drift-compensation capability, providing a more detailed characterization with respect to [26]. The accuracy of the peripheral-circuit is estimated by first replacing the PCM devices with programmable integrated resistors, having a 32-levels resolution, so as to neglect the effects of the memory devices. In order to understand the variability in the cells programmed conductance, the iterative programming algorithm is detailed below as well. Drift compensation is then tested employing PCM cells as both MAC coefficients and reference conductance.

A. Accuracy of the AIMC Unit

To evaluate the accuracy of the peripheral circuitry, without the effects of the PCM cells non-idealities, the AIMC prototype was initially tested by performing $m = 10000$ random MAC operations $\mathbf{z} = [z_1, \ldots, z_m]$, with $z_j = \sum_{i=1}^{12} w_{j,i}x_i$. In this test mode, the ePCM array has been replaced with an integrated test unit, where integrated resistors act both as weight coefficients and compensation reference conductance. Each conductance can be selected among 32 available levels. Measurements have been performed on four different testchips, which showed no significant differences among them. Results related to a sample testchip are reported in Fig. 2(a), where the yellow curve refers to MAC

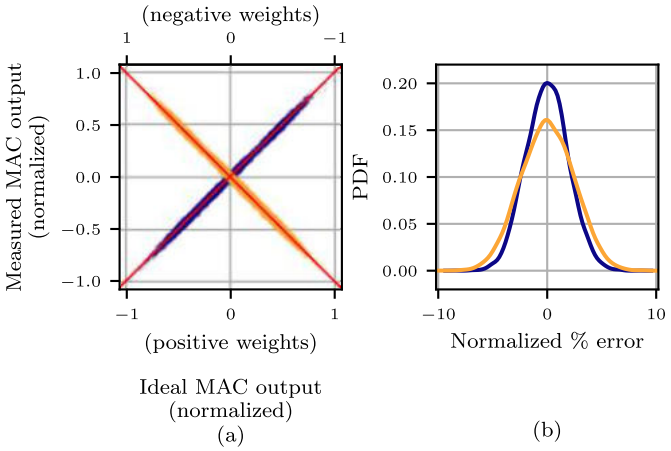

Fig. 2. Accuracy of the AIMC unit: (a) Measured MAC operations as a function of the ideal MACs, where MAC coefficients are implemented with programmable integrated resistances. MAC weights w_i employed for the yellow (purple) data are negative (positive). (b) MAC error distributions of the two data sets.

operations with negative weights, whereas the others employs positive weights. The experimental data $\mathbf{z}$ are distributed around the red lines representing the ideal MAC output $\mathbf{z}^{id}$, obtained by evaluating (1) with the nominal values of the integrated resistors used as $g_{i,j}$ and g_{REF}.

The distribution of the MAC error $\boldsymbol{\varepsilon} = \mathbf{z}^{id} - \mathbf{z}$ for the two cases is reported in Fig. 2(b). The accuracy of the circuit, defined as $(1 - \sigma_{\boldsymbol{\varepsilon}})$ [12], where $\sigma_{\boldsymbol{\varepsilon}}$ is the standard deviation of $\boldsymbol{\varepsilon}$, is then equal to 98.9% for the positive-weights MACs, and it is equal to 98.4% for the negative ones.

B. PCM Cells Programming Algorithm

Various works have shown the possibility to reach multilevel conductance levels in PCM devices through the application of suitable SET and RESET current pulses [34]. These pulses cause the heating of a relevant portion of the cell, modifying his internal structure by an electro-thermal process. The current pulses are characterized by their maximum amplitude, time duration and falling-edge slope:

- A SET pulse is a trapezoidal current pulse, consisting of a melting phase, followed by a slow crystallization phase, leading the cell resistance to decrease.
- A RESET pulse is a single, high-amplitude, rectangular current pulse that melts the central portion of the cell, which will rearrange itself randomly, bringing it into a high-resistance state.

The experimental environment employed in this work is the one presented in [35]. The programming algorithm, based on a SET Stair-Case (SSC) approach [36], is outlined in Fig. 3. Once the target conductance level $\widehat{g}$ and its absolute tolerance δg have been defined, a power RESET pulse of amplitude A_R is applied to the cell. Then a sequence of partial SET pulses is applied to the cell, starting with an initial amplitude A_{S0}. The cell conductance value is measured after the application of each pulse. If the conductance level falls within the target interval, the programming algorithm is then terminated. If the target interval is exceeded, the algorithm will start again from the power RESET; otherwise, if the

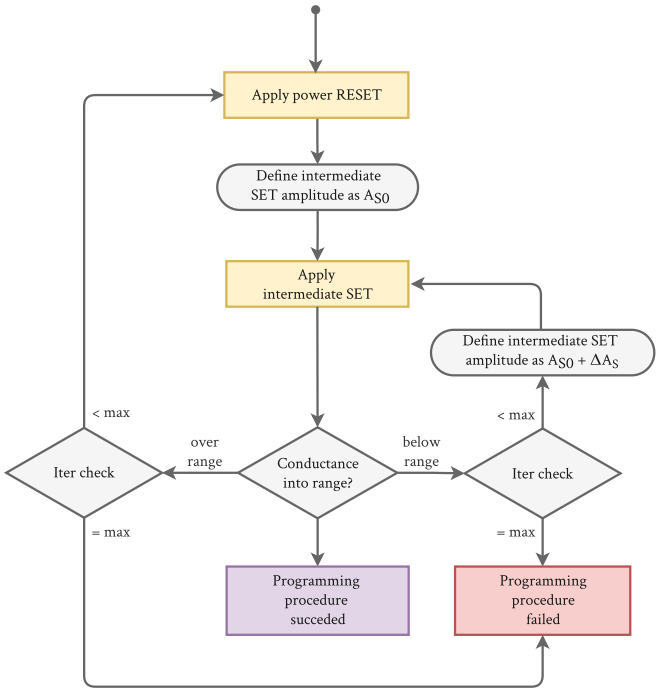

Fig. 3. Outline of the cells programming algorithm.

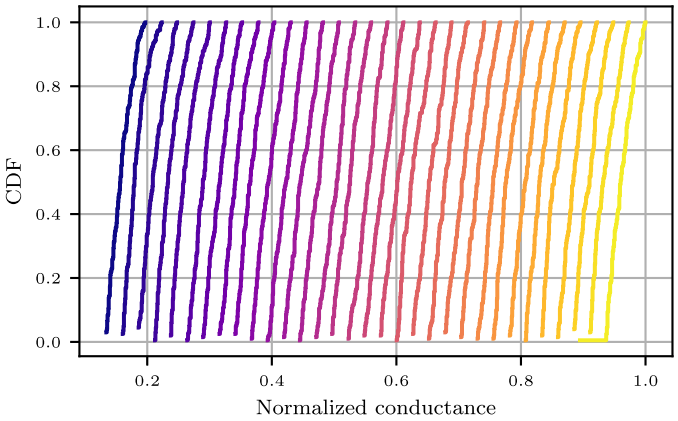

Fig. 4. CDFs of 32 levels of programmed cell conductances.

reached conductance is below the target conductance range, a new SET pulse is applied, whose amplitude is increased by ΔA_S. To prevent the algorithm from running indefinitely, a maximum number of programming attempts is enforced. Whenever the targeted conductance value is not reached within the maximum number of iterations, the cell will be excluded from further calculations. This operation corresponds to the *Iter check* box in Fig. 3.

The minimum amplitude A_{S0} of the initial SET pulse is selected taking into account the target conductance of each cell, in order to reduce the number of SET pulses to be applied. In particular, A_{S0} is gradually increased from a minimum of A_{MIN} to a maximum of $3A_{\text{MIN}}$ as a function of the target conductance range, and the SET-amplitude step ΔA_S was chosen equal to $A_{\text{MIN}}/5$.

In this work, 32 conductance levels have been chosen to program 6400 PCM devices. The targets are equally spaced

below a maximum target g^{MAX}, with an absolute tolerance $\delta g = \pm 0.025 g^{\mathrm{MAX}}$. The result of the programming procedure at $t = t_0$ is shown in Fig. 4, where the cells cumulative distribution functions (CDFs) for each programming target is reported. All the 6400 cells were programmed successfully within a maximum number of iterations equal to 250. The conductance level distributions are overlapped to allow the feasible programming of all 32 levels within the maximum reachable conductance g^{MAX}.

C. Drift Compensation

Drift-compensation is the main target of the described AIMC unit and its key element consists in the use of a reference PCM cell g_{REF} for the ramp generation. Its level can be chosen: i) to maximize the V_{OUT} output swing, and ii) to compensate the drift effects on MAC operations.

The output voltage V_{OUT} can vary between $\pm V_{\mathrm{OUT}}^{\mathrm{MAX}}$, a limit determined by the design of the output integrator. The maximum output swing, $V_{\mathrm{OUT}}^{\mathrm{MAX}}$, enforces an upper bound on the maximum MAC operation, i.e., from (3) $V_{\mathrm{OUT}}^{\mathrm{MAX}} = \frac{k V_{\mathrm{IN}}^{\mathrm{MAX}}}{g_{\mathrm{REF}}} \left[\max_j \left(\sum_{i=1}^{n} g_{j,i}\right)\right] = \frac{k V_{\mathrm{IN}}^{\mathrm{MAX}}}{g_{\mathrm{REF}}} \left[n g^{\mathrm{MAX}}\right]$. Thus, one can obtain a minimum value for g_{REF}:

$$g_{\mathrm{REF}} \geq k \frac{V_{\mathrm{IN}}^{\mathrm{MAX}}}{V_{\mathrm{OUT}}^{\mathrm{MAX}}} \left[n g^{\mathrm{MAX}}\right] \tag{5}$$

where $V_{\mathrm{IN}}^{\mathrm{MAX}}$ is the analog value corresponding to the maximum input x^{MAX}. Condition (5) represents the worst-case constraint on g_{REF}, as it assumes the maximum programmable conductance g^{MAX} for each stored weight $g_{j,i}$ [35]. However, in practical implementations, where the $w_{j,i}$ values and consequently the $g_{j,i}$ of the whole array are known, the previous condition can be relaxed considering the maximum amount of conductance per WL:

$$g_{\mathrm{REF}} \geq k \frac{V_{\mathrm{IN}}^{\mathrm{MAX}}}{V_{\mathrm{OUT}}^{\mathrm{MAX}}} \left[\max_j \left(\sum_{i=1}^{n} g_{j,i}\right)\right] = g_{\mathrm{REF}}^{\mathrm{min}} \tag{6}$$

If the inequality in (6) is satisfied, all possible MAC operations are mapped within the available output swing (as shown in the black line in Fig. 5); otherwise, the output voltage may saturate (as represented by the purple line). Thanks to the compensation technique, the first condition is maintained over time, whereas the drift-induced random drop of MAC weights would translate into a sensible reduction of the output swing, as depicted by the yellow curve of Fig. 5, with consequent issues in any elaboration of the output.

In the proposed AIMC architecture, the value of the reference cell conductance is crucial for the effectiveness of the drift compensation, as PCM devices tend to assume drift coefficients with cell-to-cell variability, and a correlation to their initial conductance [35], [37], [38], both effects leading to an imperfect compensation of the drift exponents in (4). The optimal value of g_{REF}, which satisfies (6), has been found by simulating 10000 random MAC operations $\mathbf{z}$ (the exact same set of inputs and weights used in Section III-A). V_{OUT} has been computed according to the model (3) at time t_0 with the target values of cells programming (i.e., without drift),

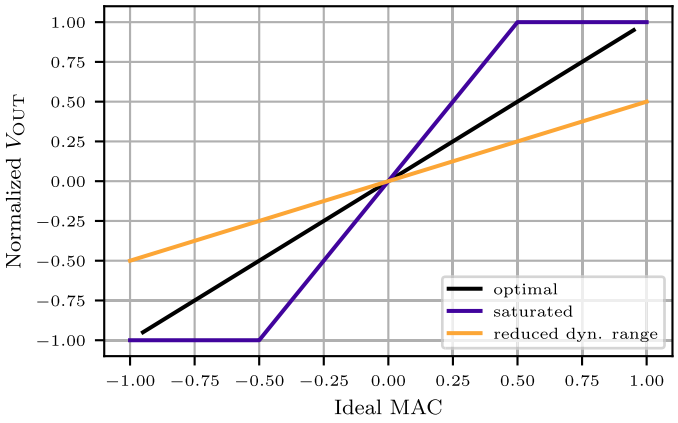

Fig. 5. Effects of different values for the reference conductance g_{REF} on the output swing of the MACs. i) optimal swing (Equation (6)); ii) saturation due to low g_{REF} level; iii) swing reduction induced by PCM cells drift with constant g_{REF}.

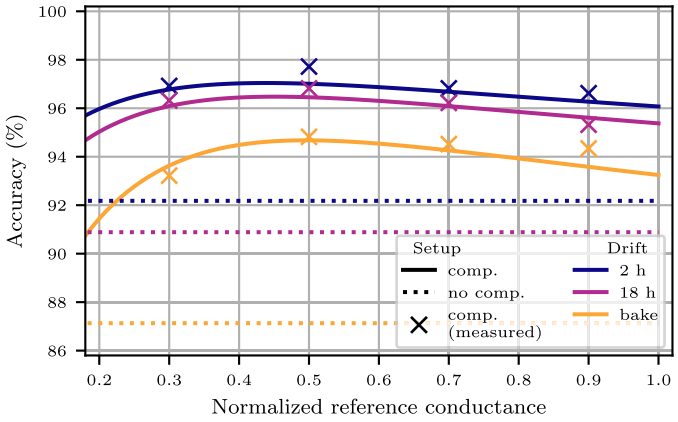

Fig. 6. MAC accuracy as a function of $\frac{g_{\mathrm{REF}}}{g^{\mathrm{MAX}}}$. Continuous lines refer to simulated results after 2- and 18-hours room temperature and 90°C bake drift. Dashed lines represent the MAC accuracy in the same conditions when no compensation is adopted. Crosses report the results of experimental evaluation of MAC accuracy for different g_{REF} levels.

obtaining the target MAC values $\widehat{\mathbf{z}}$; then, the effects of drift have been simulated in $\mathbf{z}(t)$ with (4), where the drift coefficient values have been taken from a previous work [38]. Fig. 6 depicts the MAC accuracy, already defined as $(1 - \sigma_\varepsilon)$, as a function of the $\frac{g_{\mathrm{REF}}}{g^{\mathrm{MAX}}}$ value. Different curves refers to three considered time intervals, i.e. 2 hours and 18 hours at room temperature, and after 24-hours 90°C bake. To simulate the condition where no compensation is adopted, the reference conductance has been kept constant in accordance with [26], letting thus MAC operations depend on drift. It is evident that the more effective interval of values for g_{REF} is between $\sim [0.4 - 0.6]\, g^{\mathrm{MAX}}$. The accuracy gain with respect to the uncompensated case in the three considered scenarios, when drift compensation is implemented with $\frac{g_{\mathrm{REF}}}{g^{\mathrm{MAX}}} = 0.5$, is 4.79, 5.38 and 7.81%.

As a final check, the same operations have been executed on the test chip, with $\frac{g_{\mathrm{REF}}}{g^{\mathrm{MAX}}} = 0.3, 0.5, 0.7$ and 0.9. The experimental results are coherent with the numerical simulations; in particular, the optimal level of $\frac{g_{\mathrm{REF}}}{g^{\mathrm{MAX}}}$ for drift compensation is equal to 0.5. The full set of MAC operations with $\frac{g_{\mathrm{REF}}}{g^{\mathrm{MAX}}} = 0.5$ is reported in Fig. 7, where the

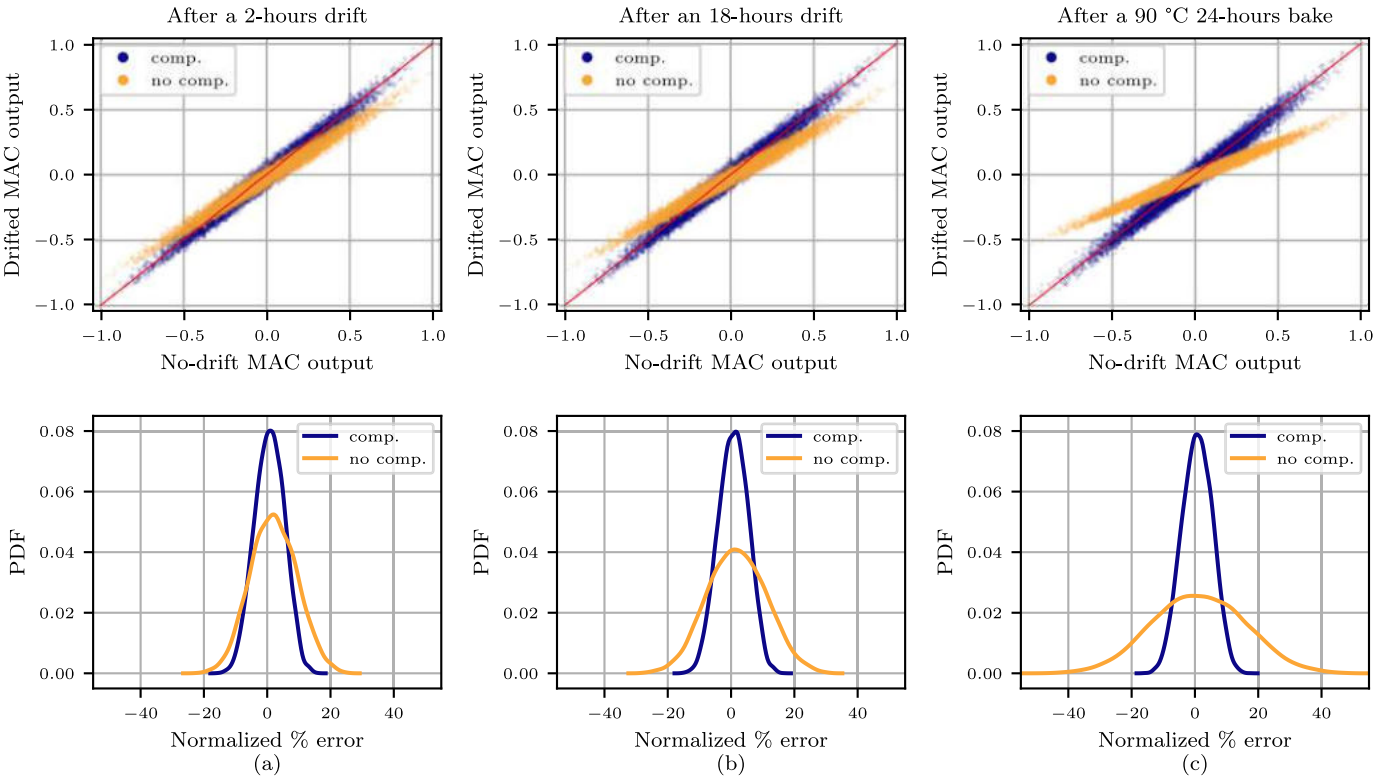

Fig. 7. (Top) Comparison of the MAC output for compensated and uncompensated cells (a) after 2 hours from programming, (b) after 18 hours and (c) after a 24-hours bake at 90 °C. (Bottom) Distribution of the MAC error ε.

measured $\mathbf{z}(t)$ are plotted as a function of the ideal MAC $\mathbf{z}^{id}$, in the three considered time instants. The results are also compared with the same operations performed with no compensation. In this case, the reference conductance g_{REF} is implemented with an integrated resistance; thus, being the ramp reference current I_{REF} constant in time, no drift compensation is applied. The distribution of the MAC error $\boldsymbol{\varepsilon} = \mathbf{z}^{id} - \mathbf{z}(t)$ is also reported in the same figure both with and without drift compensation. MAC accuracy becomes quite constant over time when compensation is adopted (97.7% after 2 hours and 96.8% after 14 hours at room temperature), even after a drift-induced 24-hours 90°C bake (94.8%) [17]; otherwise, its standard deviation $\sigma(\varepsilon)$ tends to increase with a consequent decrease of MAC accuracy over time (92.2%, 90.3% and 81.9%, respectively). It is evident that when no compensation is adopted, the output swing is reduced, as previously discussed.

IV. Modeling the Conductance Variability

Validating the device and circuit performance in a (simulated) application requires a numerical model for the device properties, namely the variability of the programmed conductance under the effect of the iterative programming, and the conductance drift, both with and without hardware compensation. To this end, PCM cells were characterized by executing a MAC operation for each $g_{j,i}$. To isolate a single cell, among the 12 that determine a MAC operation, one external input V_k has been applied at a time, setting the others to 0. The AIMC output, as expressed in (3), then reads:

$$V_{\text{OUT}_j} = \frac{g_{j,k}}{g_{\text{REF}}} V_k \,, \tag{7}$$

and depends on the single cell $g_{j,k}$ behavior only. The only nonzero input, V_k, was chosen equal to its maximum value $V_{\text{IN}}^{\text{MAX}}$ for increased accuracy.

Each individual level l can be reasonably approximated by a normal probability density $\mathcal{N}(\mu_p^{(l)}, \sigma_p^{(l)})$, whose standard deviations are depicted as crosses in Fig. 8 against the mean normalized conductance. From the data, a model for conductances affected by programming variability can be defined as $g_0 + \Delta g_p(g_0)$, where g_0 is the nominal conductance and $\Delta g_p(g_0)$ a zero-mean gaussian perturbation. To obtain the standard deviation of the Δg_p term a continuous function has been fit to the data, using the equation

$$\sigma_p(g) = \sigma_0 + \sigma_1 \tanh(g/\gamma_0) \,. \tag{8}$$

The parameters σ_0, σ_1 and γ_0 have been found by a nonlinear least squares fit using the Levenberg-Marquardt algorithm. As negative weights are implemented mapping their magnitude and sign onto different devices, and assuming that only errors on the former can be observed at the output, the model is extended towards negative g values by setting $\sigma_p(g) = \sigma_p(-g)$. This decision was grounded in the observation of a similar behavior for positive and negative weights in Fig. 2.

The standard deviation of cells spread at the end of the programming phase (time t_0) is reported in Fig. 8(a) as a function of the mean programmed conductance of each target level. As expected, all levels have been programmed with a

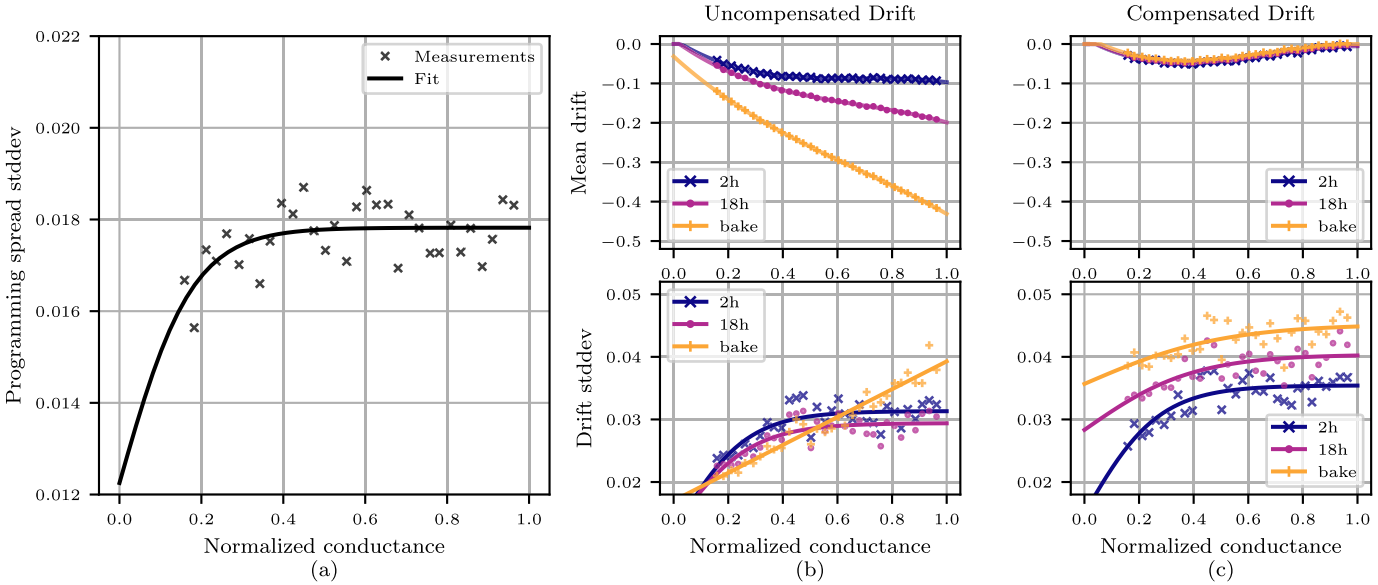

Fig. 8. (a) Standard deviation of the spread resulting from the iterative programming procedure, as a function of the average conductance of each programmed level. (b) Mean and standard deviation of the drift-induced conductance variation for cells without compensation and (c) with compensation.

spread standard deviation under the programming tolerance $\delta g = \pm 0.025 g^{\mathrm{MAX}}$. This tolerance is arbitrarily tunable, and directly impacts on the energy required for the programming phase.

Conductance drift has been then observed, both for compensated and uncompensated cells, in the same settings described in Section III-C, i.e., after 2 hours, 18 hours and after a 24 hours bake at 90 °C. The mean $\mu_d^{(l)}$ and standard deviation $\sigma_d^{(l)}$ of the conductance variation $\Delta g_d = g(t_1) - g(t_0)$ observed in each programmed level are shown in Fig. 8(b) and (c). Note how the hardware compensation scheme reduces the mean component of the drift by up to one order of magnitude (for the cells which underwent a bake), while the spread of the level is (slightly) increased. This is caused by the reference cell conductance g_{REF} being affected by its own variability, thus introducing an additional perturbation in the PCM-implemented levels. The standard deviation data has been fitted by model (8). Conversely, a polynomial of order 3 has been used for the error in the mean value of the programmed level, with a saturation applied so that it does not become positive for sufficiently low conductance values. The resulting functions $\mu_d(g)$ and $\sigma_d(g)$ are the solid lines in Fig. 8. Though measurements are scarce in the region around 0.1, the trend predicted by the model is in line with what is observed in the device characterizations performed in other works [18].

The curve describing the standard deviation of an uncompensated drift in Fig. 8(b) after the 24-hours bake results in an almost straight line. The intuitive explanation is that conductances observed after the bake are more densely packed in the lower half of the conductance domain, as showed in Fig. 9. They end up in the linear-growth region for the 2-hours and 18-hours models. Moreover, as larger conductances experience a more significant drift, they also spread out more, leading to large standard deviations. When

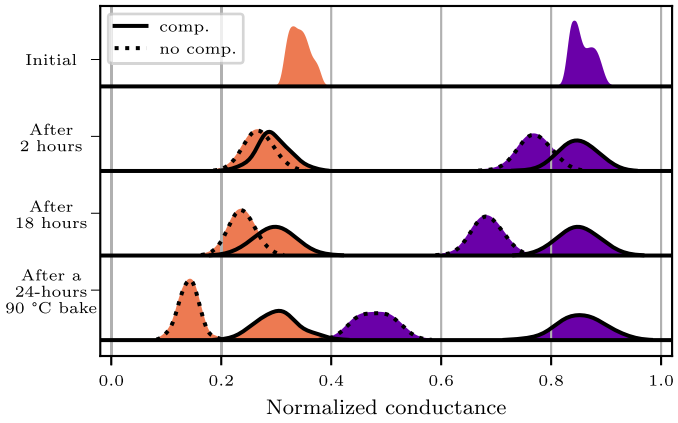

Fig. 9. Evolution over time of two batches of PCM devices, programmed to a target normalized conductance of 0.35 and 0.85. Both compensated and uncompensated cells are shown.

such large deviations are mapped to the initial target, the small post-drift domain occupied by the conductances is expanded to fill the entire horizontal domain, hence the yellow curve in the bottom plot of Fig. 8(b) surpassing the other two and having a linear trend.

As a final note, the models derived for the drift are not continuous over time, i.e. their description only refers to the specified test conditions. Furthermore, the models are extended towards negative weights by assuming the standard deviation is an even function, i.e., $\sigma_d(g) = \sigma_d(-g)$ and the mean as an odd one, i.e., $\mu_d(g) = -\mu_d(-g)$. This choice ensures that in any case drifts makes the cells more resistive as time goes by.

V. PCM-Aware DNN Training and Evaluation

To evaluate the performance of the proposed variability mitigation strategies on an actual application, a classification task on the well know CIFAR-10 dataset has been selected as a testbench [39]. Two popular neural networks have

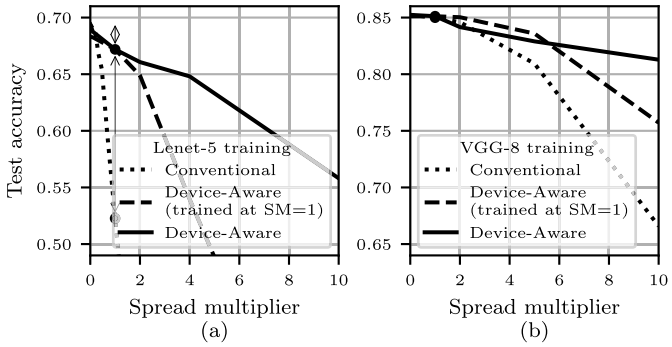

Fig. 10. Accuracy of the trained networks versus the programming spread scaling coefficient, both for the conventional (dotted line) and device-aware (DA) trainings: continuous lines refers to training and inference performed with the same spread multiplier (SM); dashed lines to training at SM = 1 and inference at different values of SM. (a) Lenet-5 and (b) VGG-8 DNNs.

been used, the Lenet-5 [40] and the VGG-8 [13], having significantly different complexities, with $\sim 8 \times 10^5$ and $\sim 4 \times 10^7$ trainable parameters, respectively. Their implementation has been suitably modified so that each synapse would emulate a PCM device, with the possibility of enabling conductance programming variability and drift at will.

With reference to a typical dense layer, the description of the j-th neuron output is $h_j = f(b_j + \sum_i w_{j,i} x_i)$, with inputs x_i, weights $w_{j,i}$, bias terms b_j and nonlinear activation $f(\cdot)$. A PCM-based layer driven by time-encoded inputs would instead be represented by:

$$h_j = f\left(b_j + \sum_i k \frac{g_{j,i}}{g_{\text{REF}}} V_i\right), \tag{9}$$

where equation (3) has replaced the MAC in the original formulation. This same reasoning can be trivially extended to convolutional layers and allows the definition of a fully PCM-based DNN.

If programming noise and drift are being introduced, the elementary synapse conductance becomes

$$g_{j,i} = g = g_0 + \Delta g_p(g_0) + \Delta g_d(g_0, \Delta t), \tag{10}$$

where $\Delta g_p(g_0)$ is the programming-induced variability, having distribution $\mathcal{N}\left(0, \sigma_p(g_0)\right)$ and $\Delta g_d(g_0, \Delta t)$ models the drift by drawing from a $\mathcal{N}(\mu_d(g_0, \Delta t), \sigma_d(g_0, \Delta t))$ distribution, using the models depicted in Fig. 8.

Both neural networks have been trained with the Adam optimizer [41], using the following parameters: exponential decay rate for the 1st and 2nd moments equal to 0.9 and 0.99, and learning rate equal to 10^{-2} for the Lenet-5 network and 10^{-3} for the VGG-8 one. Whilst training, the learning rates have been halved whenever the process would reach a plateau for a predefined amount of epochs.

Let us first observe how the two DNNs, trained without any weight variability, perform when the Δg_p term is introduced only at inference time. To widen the scope of the analysis, the injected perturbation is scaled by a multiplying factor. One reason to do it could be to relax the tolerance $\delta_g = \pm 0.025 g^{\text{MAX}}$ of the programming algorithm described in Section III-B, allowing it to converge in a lower number of iterations, speeding up the initial setup of the memory or a possible refresh of its values. The dotted curves in Fig. 10 highlight the subitaneous loss of performance as soon as noise is injected in the Lenet-5 DNN. The larger VGG-8 network, other than having a higher accuracy, is also more resilient towards the injected perturbation. This is thought to be the effect of the additional redundancy introduced by the larger number of weights, as in [42]. The datapoint corresponding to a spread multiplier (SM) of 1 has been highlighted, as it corresponds to the performance observed under the current programming parameters.

To make the network aware of the programming spread affecting its weights, a training methodology inspired by the *fake quantization* procedure [43] has been employed. This is a known methodology for the construction of NNs robust against synapse variability, and has been used extensively in the Literature [18], [44]. It requires, at train time, the addition of a perturbation before the weights are actually applied to the inputs. This obviously affects the network result, hence the starting point of the backpropagation algorithm [45]. The weight-update process then computes the derivative with respect to the original, nominal weights. Empirical evidence shows that this makes the network more resilient to weight variations. The original technique was devised for the purpose of making the network robust towards weight quantization. In that case, the properties of the injected variability would have been dependent on the number of allowed levels. For the PCM-based layers, instead, the injected perturbation models the programming-induced variability, i.e., the Δg_p term in (10). Results in Fig. 10 plotted as solid lines refer to DNNs trained and evaluated with an identical spread multiplier. The performance gain is much more pronounced for the smaller Lenet-5 than the larger VGG-8, so much so that the former becomes implementable also on the currently available technology. At a multiplier of 1, the Lenet-5 shows a 2.2% drop (69.4% down to 67.2%) in accuracy compared to the ideal, unperturbed, setup and a 15% increase (52.2% to 67.2%) with respect to the conventionally-trained DNN with weight perturbation injected at evaluation-time. This result, in conjuction with recent observations on the issues with the IR drop in large PCM arrays [46], highlights the value of the device-aware training technique to construct small and robust DNNs.

Meaningful observations follow from the behavior described by the dashed lines in Fig. 10. They represent the DNNs accuracy as a function of the increasing spread multiplier applied in the inference phase, while the device-aware training is performed with a constant multiplier of 1. Performance is improved with respect to the conventional training, allowing the network to tolerate higher spreads on the network coefficients. A direct consequence is the possibility of relaxing the requirements of the programming technique, without retraining the network, and with benefits in terms of programming speed and energy efficiency. At the same time, additional nonidealities, e.g., quantization of pre- and post-activation signals or the presence of parasitic elements along the conductive paths, to mention but a few, could be already

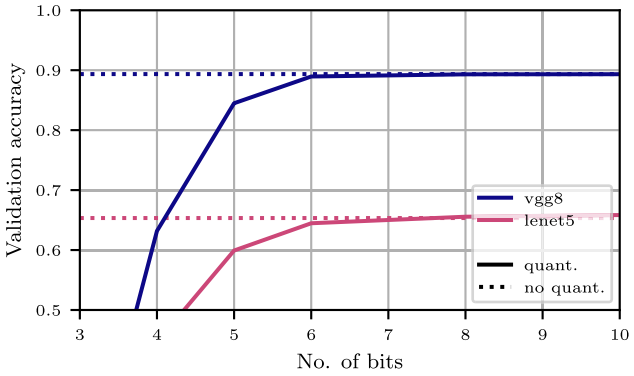

Fig. 11. Classification accuracy when quantizing the signals applied to and read from every layer, for NNs trained to exclusively address PCM programming spread using a multiplier (SM) of 1.

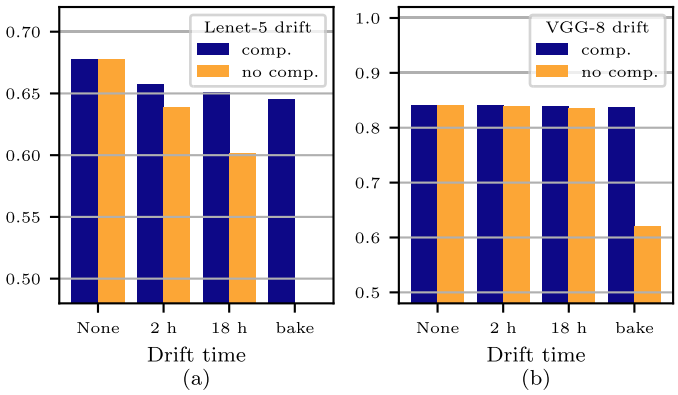

Fig. 12. Accuracy achieved when drift is applied to the DNN weights at inference time, both with and without compensation. (a) Lenet-5 and (b) VGG-8 DNNs.

managed by a network not trained specifically to address those issues, up to a certain level.

Indeed it has been observed how device-aware training techniques do not need to accurately describe the variability of interest, because of an inherent ability of the training to lead to networks robust against effect different from the perturbations used in training [18], [44]. As an example, Fig. 11 shows the classification accuracy of the two networks trained at SM=1 to address only PCM programming error, but evaluated with the introduction of quantized activations between each layer. Results prove that both networks can tolerate up to 6 bits of quantization with a performance degradation below 1%, while 5 bits introduce a loss around 5% points. More severe perturbations should be explicitly addressed during the training procedure [47]. In any case, the same perturbation-injection principle used in this work could be used to address signal quantization (the original purpose of the technique) or even the presence of parasitic elements in the analog array [44].

Having a network that can tolerate programming variability, the final step is to observe its robustness against weight drift. Both networks, trained with a spread multiplier of 1 (i.e., with programming tolerace $\delta g = \pm 0.025 g^{\mathrm{MAX}}$), have been re-evaluated by introducing the drift component of the conductance Δg_d at inference time. From Fig. 12 it is clear how the presence of the hardware compensation allows the accuracy to be retained over time. The accuracy gain after the 24-hours 90 °C bake is 36% for the Lenet-5 (even though the corresponding point for the uncompensated evaluation falls outside the range of the plot) and 22% for the VGG-8 DNN. While the drop with respect to the no-drift condition is 3% and 0.2%. Still, the benefit is larger for the smaller network. However, even the VGG-8 one, which would lose significant accuracy after the 24-hours bake, would be able to preserve its original performance with the introduction of the compensation technique.

VI. Conclusion

In this paper a combined hardware and software technique is employed to enhance Deep Neural Networks (DNNs) performance in inference tasks using measurements from an Analog In-memory Computing (AIMC) prototype based on a Phase Change Memory (PCM) IP. Empirical results show that the employed test-chip Multiply-and-Accumulate (MAC) accuracy is kept at around 95% over time, thus validating the proposed hardware compensation scheme. The spread and retention of the programmed conductances states have been characterized and modeled, including the effects of the hardware drift-compensation technique. The results have been used in a classification task on the CIFAR-10 dataset, where a device-aware training procedure was employed to make the DNNs resilient to weight variability. The tests show that the proposed combined techniques allows a 15% increase in accuracy for the Lenet-5 network compared to the conventionally trained one, with a marginal drop with respect to the ideal reference setup. Drift compensation enables the networks to retain accuracy over time and is especially beneficial for smaller DNNs, recovering up to 36% in accuracy compared to the uncompensated drift.

Acknowledgment

The authors would like to thank Chantal Auricchio and Laura Capecchi from STMicroelectronics Italy for their fundamental contribution to the testchip design and development.

References

[1] H. Amrouch et al., "Brain-inspired computing: Adventure from beyond CMOS technologies to beyond von Neumann architectures ICCAD special session paper," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2021, pp. 1–9, doi: 10.1109/ICCAD51958.2021.9643488.

[2] M. Le Gallo et al., "Mixed-precision in-memory computing," *Nature Electron.*, vol. 1, no. 4, pp. 246–253, 2018, doi: 10.1038/s41928-018-0054-8.

[3] N. Verma et al., "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Mag.* , vol. 11, no. 3, pp. 43–55, Summer 2019, doi: 10.1109/MSSC.2019.2922889.

[4] S. Gao, F. Yang, L. Zhao, and Y. Zhao, "Current research status and future prospect of the in-memory computing," in *Proc. IEEE 14th Int. Conf. ASIC (ASICON)*, Oct. 2021, pp. 1–4, doi: 10.1109/ASICON52560.2021.9620412.

[5] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: Analog computing," *Proc. IEEE*, vol. 107, no. 1, pp. 108–122, Oct. 2019, doi: 10.1109/JPROC.2018.2871057.

[6] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving matrix equations in one step with cross-point resistive arrays," *Proc. Nat. Acad. Sci. USA*, vol. 116, no. 10, pp. 4123–4128, 2019, doi: 10.1073/pnas.1815682116.

[7] X. Chen, T. Song, and Y. Han, "RRAM-based analog in-memory computing: Invited paper," in *Proc. IEEE/ACM Int. Symp. Nanosc. Archit. (NANOARCH)*, Nov. 2021, pp. 1–6, doi: 10.1109/NANOARCH53687.2021.9642235.

[8] G. W. Burr et al., "Recent progress in phase-change memory technology," *IEEE J. Emerg. Sel. Topics Power Electron.*, vol. 6, no. 2, pp. 146–162, Jun. 2016, doi: 10.1109/JETCAS.2016.2547718.

[9] J. Hartmann, P. Cappelletti, N. Chawla, F. Arnaud, and A. Cathelin, "Artificial intelligence: Why moving it to the edge?" in *Proc. IEEE 47th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2021, pp. 1–6, doi: 10.1109/ESSCIRC53450.2021.9567817.

[10] A. Athmanathan, M. Stanisavljevic, N. Papandreou, H. Pozidis, and E. Eleftheriou, "Multilevel-cell phase-change memory: A viable technology," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 1, pp. 87–100, Mar. 2016, doi: 10.1109/JETCAS.2016.2528598.

[11] R. Khaddam-Aljameh et al., "HERMES core—A 14 nm CMOS and PCM-based in-memory compute core using an array of 300 ps/LSB linearized CCO-based ADCs and local digital processing," in *Proc. Symp. VLSI Technol.*, 2021, pp. 1–2.

[12] R. Khaddam-Aljameh et al., "Hermes-core—A 1.59-TOPS/mm^2 PCM on 14-nm CMOS in-memory compute core using 300-ps/LSB linearized CCO-based ADCs," *IEEE J. Solid-State Circuits*, vol. 57, no. 4, pp. 1027–1038, Apr. 2022, doi: 10.1109/JSSC.2022.3140414.

[13] X. Sun et al., "PCM-based analog compute-in-memory: Impact of device non-idealities on inference accuracy," *IEEE Trans. Electron Devices*, vol. 68, no. 11, pp. 5585–5591, Nov. 2021, doi: 10.1109/TED.2021.3113300.

[14] D. Ielmini and G. Pedretti, "Device and circuit architectures for in-memory computing," *Adv. Intell. Syst.*, vol. 2, no. 7, Jul. 2020, Art. no. 2000040, doi: 10.1002/aisy.202000040.

[15] P. Narayanan et al., "Fully on-chip MAC at 14 nm enabled by accurate row-wise programming of PCM-based weights and parallel vector-transport in duration-format," *IEEE Trans. Electron Devices*, vol. 68, no. 12, pp. 6629–6636, Dec. 2021, doi: 10.1109/TED.2021.3115993.

[16] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 3, pp. 570–579, Sep. 2019, doi: 10.1109/JETCAS.2019.2933148.

[17] F. G. Volpe, A. Cabrini, M. Pasotti, and G. Torelli, "Drift induced rigid current shift in Ge-rich GST phase change memories in low resistance state," in *Proc. 26th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Nov. 2019, pp. 418–421, doi: 10.1109/ICECS46596.2019.8964986.

[18] V. Joshi et al., "Accurate deep neural network inference using computational phase-change memory," *Nature Commun.*, vol. 11, no. 1, p. 2473, May 2020, doi: 10.1038/s41467-020-16108-9.

[19] C. Mackin, M. Rasch, and A. Chen, "Optimised weight programming for analogue memory-based deep neural networks," *Nature Commun.*, vol. 13, p. 3765, Jun. 2022, doi: 10.1038/s41467-022-31405-1.

[20] M. Baldo et al., "Interaction between forming pulse and integration process flow in ePCM," in *Proc. 17th Conf. Ph. D Res. Microelectron. Electron. (PRIME)*, Jun. 2022, pp. 145–148, doi: 10.1109/PRIME55000.2022.9816795.

[21] M. Baldo et al., "Modeling environment for Ge-rich GST phase change memory cells," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2022, pp. 1–4, doi: 10.1109/IMW52921.2022.9779290.

[22] I. Munoz-Martin, S. Bianchi, O. Melnic, A. G. Bonfanti, and D. Ielmini, "A drift-resilient hardware implementation of neural accelerators based on phase change memory devices," *IEEE Trans. Electron Devices*, vol. 68, no. 12, pp. 6076–6081, Dec. 2021, doi: 10.1109/TED.2021.3118996.

[23] 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, doi: 10.1109/TED.2021.3089987.

[24] A. Chen et al., "Enabling high-performance DNN inference accelerators using non-volatile analog memory," in *Proc. 4th IEEE Electron Devices Technol. Manuf. Conf. (EDTM)*, Penang, Malaysia, Apr. 2020, pp. 1–4, doi: 10.1109/EDTM47692.2020.9117896.

[25] 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. Rel. Phys. Symp. (IRPS)*. Monterey, CA, USA, Mar. 2021, pp. 1–6, doi: 10.1109/IRPS46558.2021.9405191.

[26] A. Antolini et al., "An embedded PCM peripheral unit adding analog MAC in memory computing feature addressing non linearity and time drift compensation," in *Proc. IEEE 48th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2022, pp. 1–4, doi: 10.1109/ESSCIRC55480.2022.9911447.

[27] M. Pasotti et al., "A 32-KB ePCM for real-time data processing in automotive and smart power applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 7, pp. 2114–2125, Jul. 2018, doi: 10.1109/JSSC.2018.2828805.

[28] D. Ielmini, A. L. Lacaita, and D. Mantegazza, "Recovery and drift dynamics of resistance and threshold voltages in phase-change memories," *IEEE Trans. Electron Devices*, vol. 54, no. 2, pp. 308–315, Feb. 2007, doi: 10.1109/TED.2006.888752.

[29] M. Dazzi, A. Sebastian, P. A. Francese, T. Parnell, L. Benini, and E. Eleftheriou, "5 parallel prism: A topology for pipelined implementations of convolutional neural networks using computational memory," 2019, *arXiv:1906.03474*.

[30] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, and D. B. Strukov, "High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4782–4790, Oct. 2018, doi: 10.1109/TNNLS.2017.2778940.

[31] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *IEEE J. Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, Jun. 2020, doi: 10.1109/JSSC.2019.2963616.

[32] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020, doi: 10.1038/s41586-020-1942-4.

[33] 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, doi: 10.1109/ISSCC42613.2021.9365788.

[34] T. Nirschl et al., "Write strategies for 2 and 4-bit multi-level phase-change memory," in *IEDM Tech. Dig.*, 2007, pp. 461–464, doi: 10.1109/IEDM.2007.4418973.

[35] A. Antolini et al., "Characterization and programming algorithm of phase change memory cells for analog in-memory computing," *Materials*, vol. 14, no. 7, p. 1624, 2021, doi: 10.3390/ma14071624.

[36] N. Papandreou et al., "Programming algorithms for multilevel phase-change memory," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2011, pp. 329–332, doi: 10.1109/ISCAS.2011.5937569.

[37] M. Boniardi and D. Ielmini, "Physical origin of the resistance drift exponent in amorphous phase change materials," *Appl. Phys. Lett.*, vol. 98, no. 24, Jun. 2011, Art. no. 243506, doi: 10.1063/1.3599559.

[38] A. Antolini, A. Lico, E. F. Scarselli, M. Carissimi, and M. Pasotti, "Phase-change memory cells characterization in an analog in-memory computing perspective," in *Proc. 17th Conf. Ph. D Res. Microelectron. Electron. (PRIME)*, Jun. 2022, pp. 233–236, doi: 10.1109/PRIME55000.2022.9816788.

[39] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9716741

[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[42] 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, doi: 10.1109/IEDM19573.2019.8993482.

[43] V. Peluso and A. Calimera, "Energy-driven precision scaling for fixed-point ConvNets," in *Proc. IFIP/IEEE Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Oct. 2018, pp. 113–118, doi: 10.1109/VLSI-SoC.2018.8644902.

[44] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th Annu. Design Autom. Conf. (DAC)*, New York, NY, USA, Jun. 2019, pp. 1–6, doi: 10.1145/3316781.3317870.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[46] D. Ielmini, N. Lepri, P. Mannocci, and A. Glukhov, "Status and challenges of in-memory computing for neural accelerators," in *Proc. Int. Symp. VLSI Technol., Syst. Appl. (VLSI-TSA)*, Apr. 2022, pp. 1–2, doi: 10.1109/VLSI-TSA54299.2022.9770972.

[47] A. Bhattacharjee, L. Bhatnagar, and P. Panda, "Examining and mitigating the impact of crossbar non-idealities for accurate implementation of sparse deep neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 1119–1122, doi: 10.23919/DATE54114.2022.9774736.

Alessio Antolini (Graduate Student Member, IEEE) received the master's degree in electronic engineering from the University of Bologna, Italy, in March 2019, where he is currently pursuing the Ph.D. degree in electronics with the Electrical, Electronic and Information Engineering Department (DEI) "Guglielmo Marconi." In November 2019, he joined the "Ercole De Castro" Advanced Research Centre on Electronic Systems for Information and Communication Technologies, where his activities are focused on the physical characterization of phase-change memory (PCM) devices and on the design of integrated circuits for analog in-memory (AIMC) architectures.

Carmine Paolino (Graduate Student Member, IEEE) received the double bachelor's degree from Tongji University, Shanghai, China, and Politecnico di Torino, Italy, and the M.Sc. degree in analog and power electronics engineering from the Politecnico di Torino, Italy, in 2019, where he is currently pursuing the Ph.D. degree in electronics with the Electrical, Electronics and Communication Engineering Department (DET). In 2022, he was a Visiting Ph.D. Student with the Nano Laboratory Group, Tufts University, Medford, MA, USA. His research interests include mixed-signal integrated circuits, device modeling, and analog in-memory computing architectures.

Francesco Zavalloni received the Dr. (Eng.) degree (Hons.) in electronic engineering from the University of Bologna, Bologna, Italy, in 2021, where he is currently pursuing the Ph.D. degree with the Advanced Research Center on Electronic System (ARCES), under the ET-IT Ph.D. Programme. His research interests include analog in-memory computing, PCM-based programming algorithms, neural networks, and modeling PCM-based system behavior.

Andrea Lico (Graduate Student Member, IEEE) received the M.Sc. degree in electronics engineering from the University of Bologna, Italy, in 2021, where he is currently pursuing the Ph.D. degree in electronics with the Electrical, Electronic and Information Engineering Department (DEI). He joined the Advanced Research Center on Electronic Systems (ARCES), Bologna, in 2021, as a Research Fellow. His research interests include the design of analog integrated circuit for analog in-memory computing (AIMC) architectures based on phase change memory (PCM).

Eleonora Franchi Scarselli (Member, IEEE) received the M.S. degree in electrical engineering and the Ph.D. degree in electrical engineering and computer science from the University of Bologna, Bologna, Italy, in 1986 and 1992, respectively. From 1994 to 2004, she was a Research Assistant with the Faculty of Engineering, University of Bologna, where she has been an Associate Professor since 2005. She is currently teaching the design of digital integrated circuits. She is also a member of the Scientific Committee of the joint STMicroelectronics-ARCES Laboratory. Her main research interests include architecture and circuits for low-power sensing and actuating IoT nodes and for in-memory computing based on phase-change memory.

Mauro Mangia (Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering and the Ph.D. degree in information technology from the University of Bologna, Bologna, Italy, in 2005, 2009, and 2013, respectively. He was a Visiting Ph.D. Student with the École Polytechnique Fédérale de Lausanne in 2009 and 2012. He is currently an Assistant Professor with the Department of Electrical, Electronic and Information Engineering, University of Bologna, where he is also a member of ARCES, Statistical Signal Processing Group. His research interests include nonlinear systems, machine learning, compressed sensing, anomaly detection, the Internet of Things, big data analytics, and optimization.

He was a recipient of the 2013 IEEE CAS Society Guillemin-Cauer Award and the 2019 IEEE BioCAS Transactions Best Paper Award. He received the Best Student Paper Award at ISCAS2011. He was the Web and Social Media Chair for ISCAS2018.

Fabio Pareschi (Senior Member, IEEE) received the Dr. (Eng.) degree (Hons.) in electronic engineering from the University of Ferrara, Ferrara, Italy, in 2001, and the Ph.D. degree in information technology from the University of Bologna, Bologna, Italy, in 2007, under the European Doctorate Project (EDITH). He is currently an Associate Professor with the Department of Electronic and Telecommunication, Politecnico di Torino, Turin, Italy. He is also a Faculty Member of ARCES, University of Bologna. His research interests include analog and mixed-mode electronic circuit design, statistical signal processing, compressed sensing, DC/DC converters, random number generation and testing, and electromagnetic compatibility. He was a recipient of the 2019 IEEE BioCAS Transactions Best Paper Award. He also received the Best Paper Award at ECCTD 2005 and the Best Student Paper Award at EMC Zurich 2005 and IEEE EMCCompo 2019. He served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: EXPRESS BRIEFS from 2010 to 2013. He is currently an Associate Editor for the IEEE OPEN JOURNAL OF CIRCUITS AND SYSTEMS.

Gianluca Setti (Fellow, IEEE) received the Dr. (Eng.) degree (Hons.) and the Ph.D. degree in electronic engineering from the University of Bologna in 1992 and 1997, respectively.

From 1997 to 2017, he was with the Department of Engineering, University of Ferrara, Italy, as an Assistant Professor (1998–2000), an Associate Professor (2001–2008), and a Professor (2009–2017) in circuit theory and analog electronics. From 2017 to 2022, he was a Professor in electronics, signal and data processing at the Department of Electronics and Telecommunications (DET), Politecnico di Torino, Italy. Since November 2022, he has been the Dean of the Computer, Electrical, Mathematical Sciences and Engineering (CEMSE) at the King Abdullah University of Science and Technology (KAUST), Saudi Arabia. He held several positions as a Visiting Professor/the Scientist at EPFL (2002 and 2005), UCSD (2004), IBM (2004 and 2007), and the University of Washington (2008 and 2010). He is also a permanent (in-kind) Faculty Member of ARCES, University of Bologna. His research interests include nonlinear circuits, recurrent neural networks, electromagnetic compatibility, compressive sensing and statistical signal processing, biomedical circuits and systems, power electronics, design and implementation of IoT nodes, circuits and systems for machine learning, and applications of AI techniques for anomaly detection and predictive maintenance.

Dr. Setti received the 2013 IEEE CAS Society Meritorious Service Award, he was a co-recipient of the 2004 IEEE CAS Society Darlington Award, the 2013 IEEE CAS Society Guillemin-Cauer Award, the 2019 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS Best Paper Award, the Best Paper Award at ECCTD2005, and the Best Student Paper Award at EMCZurich2005 and ISCAS2011. He also received the 1998 Caianiello Prize for the Best Italian Ph.D. thesis on Neural Networks. He served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS (1999–2002 and 2002–2004) and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS (2004–2007), as the Deputy-Editor-in-Chief for the *IEEE Circuits and Systems Magazine* (2004–2007), and the Editor-in-Chief for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS (2006–2007) and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS (2008–2009). He also served on the Editorial Board of IEEE ACCESS (2013–2015) and the PROCEEDINGS OF THE IEEE (2015–2018). Since 2019, he has been serving as the first non U.S. Editor-in-Chief of the PROCEEDINGS OF THE IEEE and the flagship journal of the institute. He was a Distinguished Lecturer of the IEEE CAS Society (2004–2005 and 2013–2014) of the IEEE CAS Society, a member of the CASS Board of Governors (2005–2008), and served as the 2010 CAS Society President. In 2012, he was the Chair of the IEEE Strategic Planning Committee of the Publication Services and Products Board (PSPB-SPC). In 2013 and 2014, he was the first non North-American Vice President of the IEEE for Publication Services and Products. He served in the Program Committee of many conferences and was, in particular, the Special Sessions Co-Chair of ISCAS2005 (Kobe) and ISCAS2006 (Kos), the Technical Program Co-Chair of NDES2000 (Catania), ISCAS2007 (New Orleans), ISCAS2008 (Seattle), ICECS2012 (Seville), and BioCAS2013 (Rotterdam), and the General Co-Chair of NOLTA2006 (Bologna) and ISCAS2018 (Florence). He is a co-editor of the book *Chaotic Electronics in Telecommunications* (CRC Press, Boca Raton, 2000), *Circuits and Systems for Future Generation of Wireless Communications* (Springer, 2009), and *Design and Analysis of Biomolecular Circuits* (Springer, 2011), the coauthor of the book *Adapted Compressed Sensing for Effective Hardware Implementations* (2018) and one of the Guest Editor of the May 2002 Special Issue of the IEEE Proceedings on "Applications of Non-linear Dynamics to Electronic and Information Engineering."

Riccardo Rovatti (Fellow, IEEE) received the M.S. degree in electronic engineering and the Ph.D. degree in electronics, computer science, and telecommunications from the University of Bologna, Italy, in 1992 and 1996, respectively. He is currently a Full Professor in electronics with the University of Bologna. He has authored more than 300 technical contributions to international conferences and journals and two volumes. His research interests include mathematical and applicative aspects of statistical signal processing, on machine learning for signal processing, and on the application of statistics to nonlinear dynamical systems.

He was a Distinguished Lecturer of the IEEE CAS Society (2017–2018). He is a fellow of IEEE for his contribution to nonlinear and statistical signal processing applied to electronic systems. He was a recipient of the 2004 IEEE CAS Society Darlington Award, the 2013 IEEE CAS Society Guillemin-Cauer Award, and the 2019 IEEE BioCAS Transactions Best Paper Award. He received the Best Paper Award at ECCTD 2005 and the Best Student Paper Award at the EMC Zurich 2005 and ISCAS 2011.

Mattia Luigi Torres received the M.Sc. degree in electronics engineering from the Politecnico di Milano in 2020. He is currently a HW Design Engineer with the Smart Power TRD Group, STMicroelectronics. His research interests include PCM memory design and testing.

Marcella Carissimi received the M.Sc. degree in electronic engineering from the Politecnico of Milan. She joined STMicroelectronics in 2001. Since beginning, she focused on the design of flash memory and cost effective NVM solution. Between 2015 and 2022, she led the project of embedded PCM memory design for BCD technology for consumer product in 90nm. Her recent research interests include PCM memory design for automotive product and in AIMC with PCM memory on advance node.

Marco Pasotti received the degree in electronic engineering from the Università degli Studi di Pavia, Italy, in 1991. In 1994, he joined STMicroelectronics, Agrate Brianza, Milan, Italy, collaborating to the design of digital and mixed analog/digital IC for image acquisition and processing. In 1996, he was engaged in the design of flash memory for analog applications, multilevel storage, and applications to multimedia products. Since 2005, he has been driving the team designing cost effective eNVM solutions and lately embedded flash and PCM memories for consumer applications. His current research interests include high performances embeddable NVM memories, analog in memory computing, and new technologies for non-volatile memories and all related application aspects. He is currently a senior member of the technical staff.

Open Access funding provided by 'University of Bologna' within the CRUI CARE Agreement