Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN
INGEGNERIA ELETTRONICA, TELECOMUNICAZIONI
E TECNOLOGIE DELL'INFORMAZIONE

Ciclo XXXV

Settore Concorsuale: 09/E3 – ELETTRONICA

Settore Scientifico Disciplinare: ING-INF/01 – ELETTRONICA

PCM-based in-memory computing: architectures, circuits and applications

---

Candidato: Alessio Antolini

Coordinatore:
Prof. Aldo Romani

Supervisore:
Prof. Antonio Gnudi

Cosupervisore:
Prof.ssa Eleonora Franchi Scarselli

Esame finale anno 2023

"How far to go
I cannot say.
How many more
Will journey this way?"

"Storms in Africa", Enya - 1988

Abstract

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Ph.D. in Electronics, Telecommunications, and Information Technologies Engineering

PCM-based in-memory computing: architectures, circuits and applications

Alessio Antolini

Analog In-memory Computing (AIMC) has been proposed in the context of Beyond-Von Neumann architectures as a valid strategy to reduce internal data transfers energy consumption and latency, and to improve compute efficiency. The aim of AIMC is to perform computations within the memory unit, typically leveraging the physical features of memory devices. Among resistive Non-volatile Memories (NVMs), Phase-change Memory (PCM) has become a promising technology due to its intrinsic capability to store multilevel data. Hence, PCM technology is currently investigated to enhance the possibilities and the applications of AIMC. This thesis aims at exploring the potential of new PCM-based architectures as in-memory computational accelerators. In a first step, a preliminar experimental characterization of PCM devices has been carried out in an AIMC perspective. PCM cells non-idealities, such as time-drift, noise, and non-linearity have been studied to develop a dedicated multilevel programming algorithm. Measurement-based simulations have been then employed to evaluate the feasibility of PCM-based operations in the fields of Deep Neural Networks (DNNs) and Structural Health Monitoring (SHM). Moreover, a first testchip has been designed and tested to evaluate the hardware implementation of Multiply-and-Accumulate (MAC) operations employing PCM cells. This prototype experimentally demonstrates the possibility to reach a 95% MAC accuracy with a circuit-level compensation of cells time drift and non-linearity. Finally, empirical circuit behavior models have been included in simulations to assess the use of this technology in specific DNN applications, and to enhance the potentiality of this innovative computation approach.

Acknowledgements

Innanzitutto sono estremamente grato ai miei supervisori, la Prof.ssa Eleonora Franchi Scarselli e il Prof. Antonio Gnudi, per il loro supporto, consiglio e guida durante l'intero percorso di dottorato e nello sviluppo della tesi. Vorrei anche ringraziare i nostri partner di STMicroelectronics Marcella Carissimi, Mattia Luigi Torres, Marco Pasotti, Chantal Auricchio e Laura Capecchi per il loro fondamentale contributo alla progettazione, sviluppo e caratterizzazione del testchip. Fondamentale l'apporto dei colleghi Andrea Lico e Francesco Zavalloni, senza i quali molti dei risultati di questo lavoro non sarebbero stati possibili. Un grazie anche agli altri colleghi Matteo D'Addato, Luca Perilli e Alessia Elgani, con cui, seppur non lavorando sullo stesso progetto, ho comunque condiviso avventure (e sventure) di questi anni. Ringraziamenti sentiti vanno ai collaboratori universitari che hanno arricchito questa ricerca, ovvero il Prof. Riccardo Rovatti e il Prof. Mauro Mangia dell'Università di Bologna, insieme al Prof. Fabio Pareschi del Politecnico di Torino, con il fondamentale e costruttivo contributo di Carmine Paolino. Stimolante e originale è stato il ruolo del collega Said Quqa e dei suoi supervisori Prof. Luca Landi e Prof. Pierpaolo Diotallevi, a cui devo sincera gratitudine.

Questo percorso non sarebbe però stato possibile senza il supporto di tutte quelle persone che fanno parte della mia vita (anche) da fuori. Grazie a Matteo, Said, Richard, Carlotta, Elisa e Stefania (e tanti altri). Grazie a Luca, che ha sempre creduto in me e che mi ha fatto crescere. Grazie a mio padre, che mi ha sempre supportato. Grazie a mia madre, questo lavoro è dedicato a te, sperando che ovunque tu sia, tu possa esserne orgogliosa.

Contents

List of Figures

List of Tables

List of Abbreviations

AIMC	**A**nalog **I**n-**M**emory **C**omputing
CMOS	**C**omplementary **M**etal **O**xide **S**ilicon Field Effect Transistor
DMA	**D**irect **M**emory **A**ccess
DNN	**D**eep **N**eural **N**etwork
DRAM	**D**ynamic **R**andom **A**ccess **M**emory
IMC	**I**n-**M**emory **C**omputing
MAC	**M**ultiply **A**nd **A**ccumulate
MRAM	**M**agnetic **R**andom **A**ccess **M**emory
PCM	**P**hase-**C**hange **M**emory
RMS	**R**oot **M**ean **S**quare
RSC	**R**eset **S**tair-**C**ase
RSP	**R**eset **S**ingle-**P**ulse
SMU	**S**ource **M**eter **U**nit
SRAM	**S**tatic **R**andom **A**ccess **M**emory
SSC	**S**et **S**tair-**C**ase
SSP	**S**et **S**ingle-**P**ulse
SHM	**S**tructural **H**ealth **M**onitornig

To my mother

Chapter 1

Introduction and context

This Chapter provides a description of the In-memory Computing (IMC) field, an alternative to conventional computational architectures, where some computation tasks take place directly in the memory unit. Then, a brief outline of the memory devices that can support this method, along with some examples of applications that could benefit from it, are provided.

1.1 The Von Neumann bottleneck

In the last decades, computing systems have been mainly built on the basis of the Von Neumann architecture, where the processing unit and the memory one are physically separated. In the execution of various computational tasks, large amounts of data need to be continuously transferred between processing and memory units, which entails significant costs in terms of latency and power. In particular, the latency, associated with data accessing from storage units, constitutes a performance bottleneck in a wide range of applications, particularly for the data-centric computational workloads related to artificial intelligence. This issue is known as the "Von Neumann bottleneck".

The performances of processors have been rapidly increasing, following the well-known Moore's law. The storage unit mainly used in modern computers is typically implemented with dynamic random access memory (DRAM), which is a high-density storage solution based on the charging and discharging of capacitors. The performance of this memory depends mainly on two aspects, namely: i) the speed of reading and writing, i.e., charging and discharging of the internal capacitors, and ii) the memory bandwidth of the interface between the devices. Although the charge and discharge rate of the capacitors has been increasing following Moore's law, it is still slower than the processing speed of the processors. Furthermore, the interface between memory and processor is typically implemented by dedicated mixed-signal circuits, and the increase in its bandwidth is mainly limited by the integrity of the signal in the interconnection paths. Consequently, DRAM performances improvements have been much slower than processors, and at the present, the performance of DRAM has become an huge bottleneck of overall computer performance, the so-called "memory wall". Moreover, the energy consumption related to data

movements represents another significant challenge, as computing systems are now severely constrained from the energy standpoint. As an example, in a 45-nm Complementary Metal Oxide Semiconductor (CMOS) technology node, the energy cost of multiplying two numbers is orders of magnitude less than accessing from memory [1].

Some current approaches, such as employing hundreds of parallel processors (for example, as in Graphics Processing Units, GPUs) or Application-specific Integrated Circuits (ASICs), cannot efficiently overcome the data movement challenge. Therefore, the need to explore new architectures with an intrinsic alternative organization of memories and processing units is becoming increasingly evident.

1.2 In-memory computing

1.2.1 Context and aims

In-memory Computing (IMC) is an alternate design approach where certain computational tasks are performed within the memory unit itself, organized as a computational memory unit. Figure 1.1 schematically illustrates the differences in the computation mechanism between a conventional computing system and a in-memory computing one. In the first case, in the top of Figure 1.1, an operation is performed on the data and data must be conveyed to the processing unit, resulting in significant costs in terms of latency and energy. In the case of in-memory calculation, (bottom), the entire operation is performed within the computational memory unit, thus avoiding the need to move data into the processing unit. The computational activities are performed within the confines of the memory matrix and its peripheral circuits, without deciphering the contents of the individual memory elements. This approach is generally achieved by exploiting the physical attributes of the memory devices, their array-level organization, the peripheral circuitry as well as the control logic.

The advantages in power consumption in IMC architectures arises mostly from the massive parallelism afforded by a dense array of millions of memory devices performing computation. It is also likely that by introducing physical coupling between the memory devices, we can further reduce the computational time complexity [2], and this is an attractive aspect for all those applications that require simple but repeated operations on a large set of data, such as matrix operations or convolutions. By blurring the boundary between the processing unit and memory unit, it is possible to achieve significant improvements in computational efficiency. However, this comes at the expense of the generality offered by the conventional approach, in which the memory and processing units are functionally distinct from each other. In fact, unlike generic processors, which can perform any type of calculation, in the IMC approach it is possible to perform only a limited set of operations. Furthermore, the in-memory calculation can only offer limited precision due to the analog nature

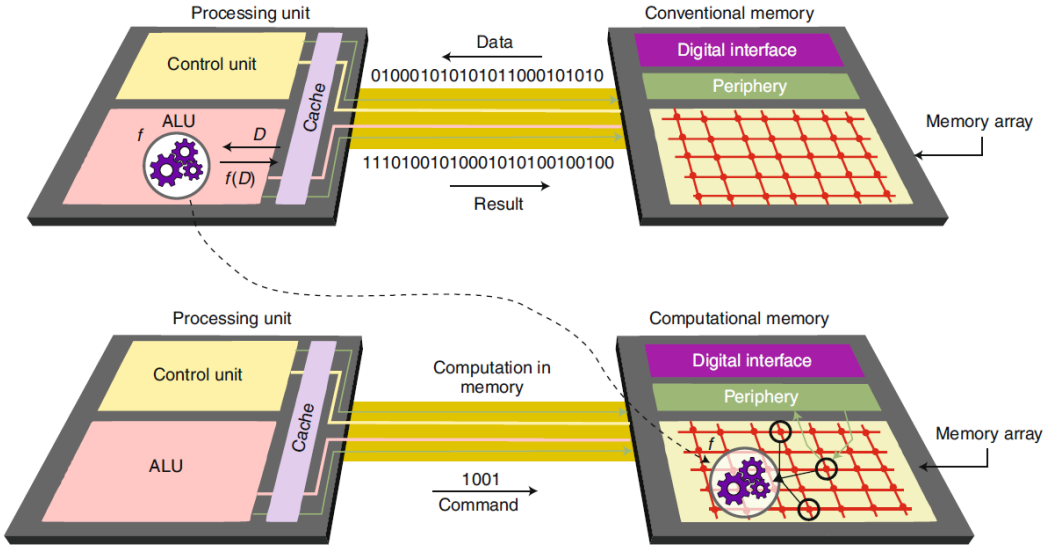

FIGURE 1.1: Difference between the execution of a generic computation in a conventional architecture (top), and with the In-memory Computing paradigm (bottom). Adapted from [1].

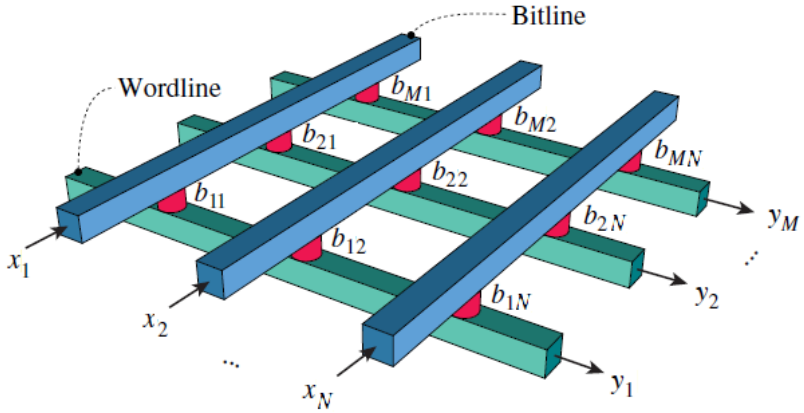

FIGURE 1.2: Example of a physical array to perform MVMs. Adapted from [3].

of the operations performed within the memory, as opposed to conventional digital calculation which grants arbitrarily high precision.

1.2.2 Analog In-memory computing

At the heart of the several computation algorithms are Matrix-Vector Multiplication (MVM) functions. For purely digital computation, these operations can be reduced to floating-point or fixed-point operations with an appropriate accuracy requirement. Alternatively, analog computing elements can be used to perform the matrix operations. Analog In-memory Computing (AIMC) for matrix operations exploit the possibility to map a 2-D matrix into a physical array (as depicted in Figure 1.2) with an appropriate number of rows and columns in accordance with the abstract

mathematical operand. At the intersection of each row and column there is a memory element with conductance $b_{i,j}$, that can represent a generic element of the matrix $\mathbf{B}$ involved in the computation. The components of a voltage vector $\mathbf{x} = x_{i=1,\dots,N}$ are applied to the N rows, and the currents at the n columns $\mathbf{y} = y_{j=1,\dots,M}$ are then collected. Exploiting Ohm's and Kirchhoff's, the expression of the collected currents are:

$$\mathbf{y} = \mathbf{B} \cdot \mathbf{x} = \begin{bmatrix} y_1 = \sum_{j=1}^{N} b_{1,j} x_j \\ \vdots \\ y_M = \sum_{j=1}^{N} b_{M,j} x_j \end{bmatrix} \tag{1.1}$$

which is equivalent to a matrix-vector multiplication.

The use of arrays of conductive elements for matrix multiplication is not new; it was proposed many years ago [4], [5]. With renewed interest in deep learning, it gained attention again as a possible solution to speed the required computations [1], [6] up. To maintain the benefits noted above, this would mean that the weight data are stored in a physical array, and that all operations are performed locally with the weights in place. The natural choice for such arrays come from memory technologies. Ideally, the requirements for a memory device to be employed for AIMC are: i) storage and retain of weights; ii) nondestructive readout mechanism; and iii) possibility to read and write the entire memory array in one single operation. While i) and ii) are conceivable, iii) is not feasible in conventional memories which are optimized for random sequential access of size-limited words. Thus, conventional memory elements must be arranged in an array architecture that differs form the architecture of conventional memory, and the employed architecture strongly depends on the type of memory devices being employed in the computation.

1.3 Memory devices

Primary techniques used to store information have been based on the presence or absence of charge, as occurs in dynamic random access memory (DRAM), static random access memory (SRAM) and flash memory. A SRAM cell consists of two CMOS inverters connected back to back. The charge is confined within the barriers formed by FET channels and by gate insulators. The stored charge retention is small and an external source constantly replenishes the lost charge. SRAM has almost unlimited cycling endurance and sub-nanosecond read and write access times. In SRAM the information is stored in the form of electric charge, with almost unlimited cycling endurance and sub-nanosecond read and write access times [1], [7]. A DRAM cell comprises a capacitor that serves as the storage node, which is connected in series to a FET. The storage node of a flash memory cell is coupled to the gate of a FET. A range of in-memory logic and arithmetic operations can be performed using both SRAM and DRAM. Capacitive charge redistribution serves as the foundation for many of them, in particular storing and sharing of charge across multiple storage nodes. In DRAMs, simultaneous reading of devices along multiple rows can be used

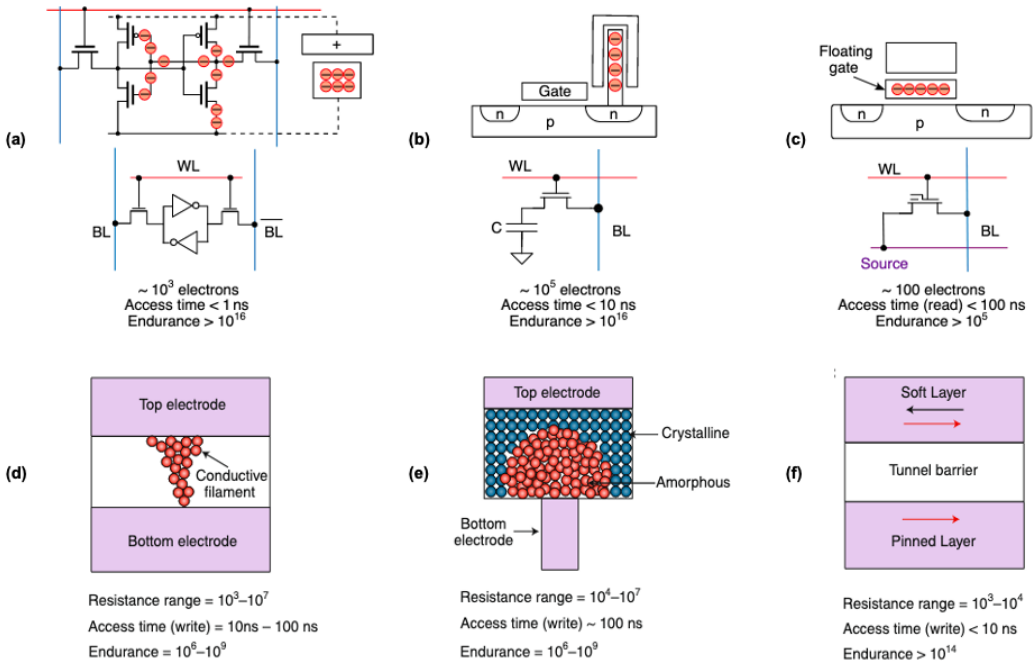

FIGURE 1.3: Summary of the most common memory devices: (a) SRAM; (b) DRAM; (c) Flash; (d) RRAM; (e) PCM; (f) MRAM. Adapted from [1].

to execute basic Boolean functions within the memory array [8], [9]. SRAM arrays can also be used for matrix-vector multiplication operations, [10]. If the elements of A and x are limited to signed binary values, the multiply operation is simplified to a combination of XNOR and accumulate functions. A SRAM cell can be also designed to execute XNOR operations within each memory cell [11]. In case the input is non-binary, one approach would be to employ capacitors in addition to the SRAM cells [12].

More recently, a novel class of memory devices has emerged, where information is stored in terms of differences in the atomic arrangements of the materials they are made of. Such differences manifest themselves as a change in resistance and, therefore, these devices are termed as resistive memory devices (or, for brevity, "memristive"). Among these, the most important are phase change memory (PCM), resistive random access memory (RRAM), and magnetic random access memory (MRAM). One of the attributes of memristive devices that can be exploited for computation is their non-volatile binary storage capability, thus, allowing logical operations to be implemented through the interaction between the voltage and resistance state variables [5]. In addition, their non-volatile storage capability, in particular, the ability to store a continuum of conductance values, facilitates the computation of analogue MVMs. Memristive devices also exhibit an accumulative behaviour [13], whereby the conductance of devices such as PCM and RRAM progressively increases or decreases with the application of an appropriate sequence of programming pulses. This non-volatile accumulative behaviour can be exploited in several applications [14].

In general, one of the main characteristics of a memory device is its access time, which corresponds to the speed with which information can be stored (written) and retrieved (read). Another key feature is its reliability, which refers to the number of times a memory device can be switched from one state to another. A summary of the most common charge-based and resistive memory devices is represented by Figure 1.3.

The work developed in this Thesis is based on Phase-change memories, whose technology will be exposed in Section 1.5.

1.4 Applications of AIMC

The in-memory implementation of matrix-vector multiplications, examined in the previous Paragraphs, can be used in a wide range of application domains, ranging from scientific computation, which requires computational high precision, to stochastic computation, which exploits imprecise and random processes.

AIMC can be used both to reduce the computational complexity of a problem, and to reduce the amount of data accessed by performing computations within memory arrays. Data-centric applications in machine learning and scientific processing take full advantage of the reduced amount of memory accesses. In this Section, some examples are given to show how in-memory computing has been applied in various fields, such as scientific computing and artificial intelligence.

1.4.1 Algebra accelerators

The matrix-vector multiplication represents one of the most frequent operations in the field of scientific computing applications. However, although approximate solutions may be sufficient for many computational tasks in the field of artificial intelligence, the realization of an in-memory processing unit capable of effectively addressing the problems of scientific processing still remains a challenge [1]. This precision limitation can be solved to some extent, for example, with mixed-precision computing, an alternative approach to achieve high-precision processing, based on the combined use of in-memory processing and conventional processing. This approach is based on the fact that many calculation activities can be formulated as a sequence of two distinct parts. In the first part, an approximate solution is obtained. In the second part, the resulting error is calculated with high accuracy. Based on the calculated error, the approximate solution is perfected and then the first part is repeated. The first part typically has a high computational load, while the second part has a low computational load. By repeating this sequence several times, it is often possible to achieve a highly accurate solution. In mixed-precision memory calculation, the basic idea is to use a low-precision computational memory unit to obtain the approximate solution of the first part, and a high-precision processing unit to create the second one. In this way, it is possible to take advantage of the high

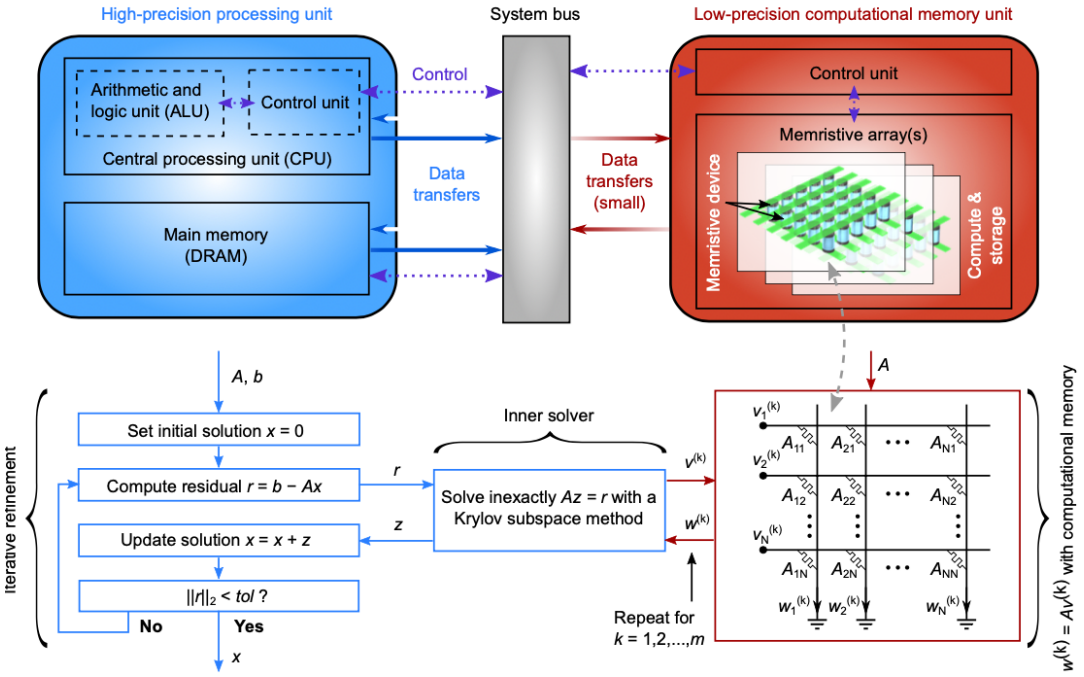

FIGURE 1.4: Mixed-precision in-memory computing example. Taken from [15].

area and energy efficiency of the computational memory unit, in which most of the calculation is performed, but still with high-precision results.

The implementation of this concept and the experimental demonstration of solving a system of linear equations using PCM devices was presented in [15] and schematically depicted in Figure 1.4. The basic principle here shown, is to exploit a fast but imprecise matrix-vector products execution. An approximate solution is obtained with in-memory computation with an iterative linear solver; then, this solution is refined exploiting the residual error, which is calculated with high accuracy through a conventional digital architecture. Experimental results have demonstrated that the linear system can be solved with an error of approximately $1.3 \cdot 10^{-15}$ by performing a large number of iterations. Thus, the final error appears to be limited by the precision of the high-precision processing unit. A significant performance gain in terms of consumption was also shown.

The main limitation of this technique is that the data must be stored in both computational memory and the memory of a high-precision digital processing unit, which increases the resources required to perform the task.

1.4.2 Signal processing

In the field of signal processing, compressed sensing and recovery is one of the applications that could benefit from matrix-vector multiplication performed in computational memory units. The goal behind of compressed sensing is to acquire a

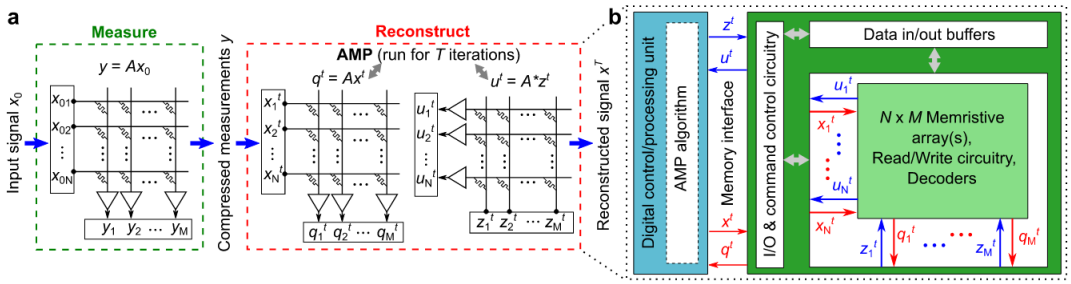

FIGURE 1.5: A $N \times M$ memristive crossbar encoding the measurement matrix A used to acquire the CS measurements and to realize the matrix-vector computations of the recovery algorithm. Taken from [16].

large signal at a sampling rate which is below the Nyquist frequency and subsequently reconstruct that signal accurately [17]–[19]. Unlike most other compression schemes, sampling and compression are performed simultaneously, with the signal being compressed as it is sampled. These techniques have widespread applications in medical imaging domains, in security systems and in camera sensors.

Compressed sensing consists in mapping a signal x of length N to a measurement vector y of length $M < N$. If this process is linear, then it can be modeled from a measurement matrix $\mathbf{M}$ of size $M \times N$. The idea is to store the measurement matrix in a computational memory unit in order to allow the execution of compression with a time complexity equal to $O(1)$. An approximate message passing (AMP) algorithm [16] can be used to retrieve the original signal from the compressed sampling vector y, using an iterative algorithm that involves multiple matrix-vector multiplications on the same measurement matrix and its transpose (Figure 1.5). In this way it is possible to use the same matrix that was encoded in the computational memory unit also for the reconstruction, reducing the complexity of the reconstruction from $O(MN)$ to $O(N)$.

A recent work related to the field of compressed sensing has been based on some results of this Thesis [20].

1.4.3 Artificial intelligence

A neural network consists of at least two layers of nonlinear neuronal units (neurons) interconnected by adjustable synaptic weights [21]. The propagation of data through the layers of the network involves a sequence of multiplications between matrices. The simplest neural network model is the feed-forward network, in which information can travel in only one processing direction. These types of networks can be single-layer, i.e., consisting only of input and output levels, or multi-layer with various hidden layers.

Another type of neural network is represented by recurrent networks, in which the output values of a higher-level layer are used as an input to a layer of a lower level. These interconnections between layers allow the system to create a memory

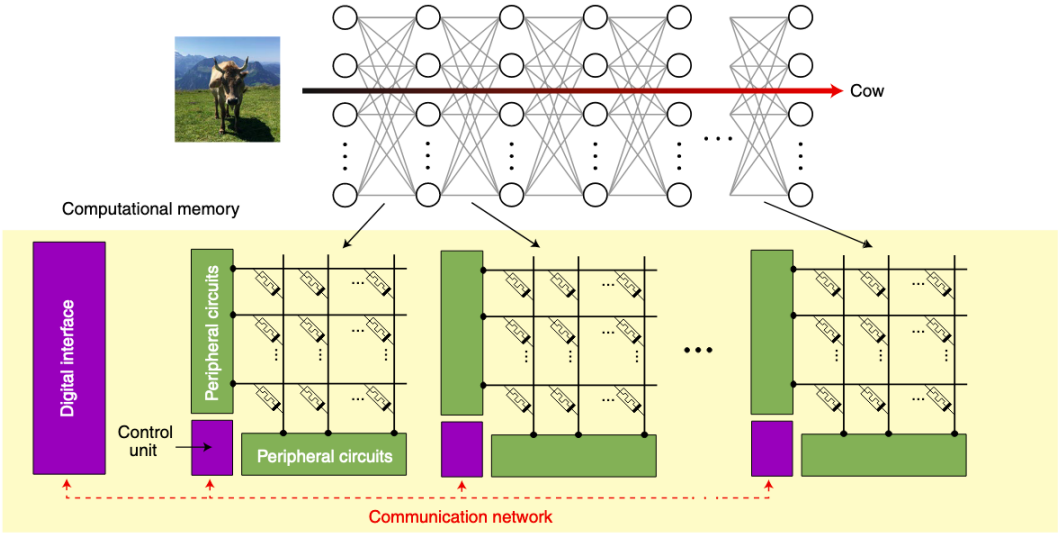

FIGURE 1.6: Top: Symbolic representation of a neural network. Bottom: Resistive array implementing a neural network layer. Taken from [21].

effect. Recurring networks are in fact used, for example, in speech recognition, translation, and handwriting recognition.

Modern neural networks (Deep neural networks, DNNs), can have more than a thousand layers. By adjusting the weights, with an optimization procedure which employs millions of examples, these networks can solve some problems remarkably well.

In addition to the multiplications between large and dense matrices, which are implicit in their functioning, DNNs are highly resistant to numerical inaccuracies, especially for direct inference applications. These features make DNNs particularly suitable for implementation on computational memory units, which can also implement non-binary networks thanks to their multilevel storage capability. A DNN can be mapped to multiple crossbar arrays of storage devices that communicate with each other. By exploiting the physical structure of the computational memory unit, a layer of a DNN can be implemented on (at least) one crossbar, in which the weights of the layer are stored in the state of charge or conductance of the memory devices at the cross points (Figure 1.6). The propagation of data through the layer is performed in a single step by entering the data in the rows of the crossbar and decrypting the results in the columns. The results are then passed through the non-linear function of the neuron, and then fed into the next level. The nonlinear function of the neuron is typically implemented at the periphery of memory arrays, using analog or digital circuits [21].

The calculations for DNNs includes both training, during which the network weights are optimized on a labeled dataset, and direct inference, where the trained network is used for classification, prediction or other tasks [22]. The efficiency of the matrix-vector multiplication, in terms of speed and energy consumption, achieved through in-memory processing is very relevant for inference-only applications, in

which data are propagated through the network on offline trained weights. In this scenario, the weights are typically trained using conventional hardware, based on graphics processing units (GPUs), and are subsequently programmed into the in-memory processing device that performs the inference. Due to the non-idealities introduced by the memory devices and the analog circuits present in the in-memory processing chip, it is often necessary to include customized techniques in the training algorithm to mitigate the effect of such non-idealities on the accuracy of the network [15]. The training procedure should be generic and as hardware independent as possible so that the network only needs to be trained once to be deployed across a multitude of different chips [21].

In-memory computing can also be exploited in the context of supervised DNN training, generally referred to as backpropagation. This type of training involves three stages: i) forward propagation of the labeled data across the network; ii) backward propagation of error gradients from the output to the network input, and iii) weight update based on the calculated gradients with respect to the weights of each layer. This procedure is repeated over a large dataset of labeled examples many times until the network achieves satisfactory performance. Due to the need to repeatedly show large datasets to neural networks with a high number of layers, this approach can take several days or weeks to train state-of-the-art networks with Von Neumann machines. The concept of mixed-precision in-memory processing, described in the previous Section, can be extended to the problem of training deep neural networks where a computational memory unit is used to perform the back and forth steps, while the variations of weight are calculated with high precision [1].

Some results of AIMC based on Phase-change memory cells employed in the DNN scenario will be provided in Chapters 3 and 4.

1.5 Phase-change memory technology

Phase-change memories (PCM) represent an emerging technology in the field of non-volatile memories. A PCM device typically consists of a small active volume of phase change material sandwiched between two electrodes (as depicted in Figure 1.7). The phase change material can be changed from a low conductivity state to a high conductivity state, and vice versa, by applying pulses of electric current. The data are stored using the different electrical resistivity between the two possible states (the high resistivity state can represent a logic "0", while the low resistivity state can represent a logic "1") and can be read by measuring the electrical resistance of the cell [13].

One of the key-features of PCM is that the data to be stored can be written in a few nanoseconds, with a high retention (typically tens of years at room temperature [24], [25]). This property allows the use of PCM cells for non-volatile storage,

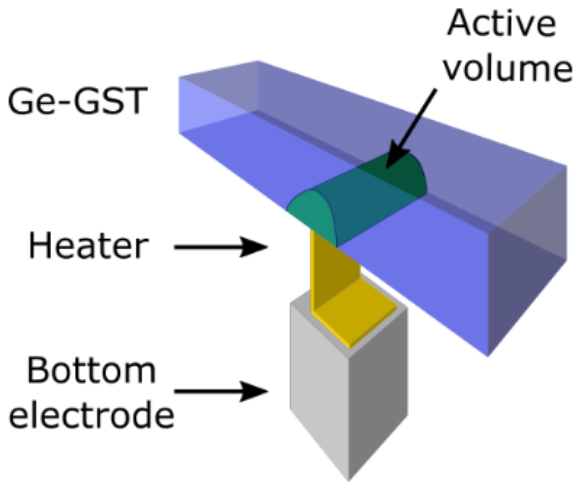

FIGURE 1.7: Schematic representation of a GST-type PCM cell. Taken from [23].

such as Flash memories and hard-disks, operating at almost the same speed as high-performance volatile memory such as DRAM memory [21], and being then recently marketed as storage memories for conventional computing systems.

Although the physics related to the functioning of PCM devices has been extensively studied since their discovery in the 1960s, there are still several open questions relating to their electrical, thermal and structural dynamics. In the following, a brief description of the operating principle will be provided, underlying as well the read and write operations of PCM devices. The characteristics that make these memories suitable for the AIMC context, and the main issues affecting their accuracy are then illustrated.

1.5.1 Working principle

In PCM, data are recorded by causing a phase change in the material within the memory device, that is to make it shift from a crystalline (ordered) phase to an amorphous (disordered) one, and vice versa. This transformation is accompanied by a sharp change in the electrical and optical properties of the material. The amorphous phase has a high electrical resistivity and a low optical reflectivity, while the crystalline phase has a low electrical resistivity (sometimes three or four orders of magnitude lower) and a high optical reflectivity. The optical properties of phase change materials have been widely used in optical data storage devices such as DVDs and Blu-Ray discs. The electrical storage of binary data principle, on the other hand, relies on the difference in resistivity between the two phases. Therefore, a write operation in a PCM cell involves the transition of state from the amorphous to the crystalline through the application of an appropriate electrical pulse. A reading operation typically involves a current-based readout the electrical resistance of the device, which allows to discern if the state of the cell is amorphous (high resistance,

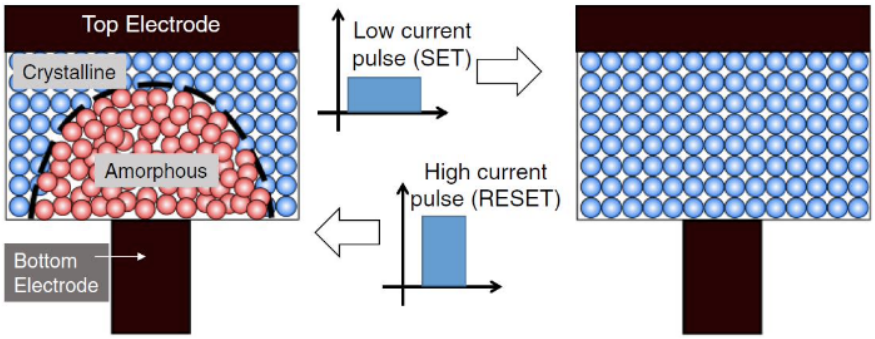

FIGURE 1.8: PCM cells programming principle. Taken from [26].

logical "0") or crystalline (low resistance, logical "1").

The amorphous phase of the phase change material (typically composed of an alloy of Germanium, Antimony and Tellurium, known as GST) is thermodynamically unstable, but the crystallization time at room temperature is very long [24]. However, by heating the amorphous material to a sufficiently high temperature, but below the melting temperature, it will crystallize rapidly. To transform the material back into the amorphous state, it must be heated above its melting temperature and then rapidly cooled. This rapid cooling will "freeze" the atomic structure in a disordered state.

In PCM cells heat is induced by Joule effect with the application of an electric current through the phase-change material [13]. The electrical pulse used to switch the device to the amorphous state is called the RESET pulse; the pulse used to switch the device to the crystalline state is called the SET pulse. A RESET pulse therefore refers to a current pulse that can melt a significant portion of the phase change material. When the pulse stops abruptly, the molten material remains in the amorphous state. In the resulting RESET state, the device will be in a high resistance state as the amorphous region blocks the lower electrode. When a SET pulse is applied to a PCM device in the RESET state, part of the amorphous region crystallizes. The temperature corresponding to the highest crystallization rate is typically 400°C, which is lower than the melting temperature (600°C). The programming principle of PCM devices is depicted in Figure 1.8 [26], while Figure 1.9 shows the shapes of SET and RESET pulses.

The resistance state reached after the application of a SET or RESET pulse can be deciphered by biasing the device with a low amplitude reading voltage, to avoid the perturbation the phase configuration [27]–[29].

1.5.2 Multilevel storage

A key-property of PCM devices is the possibility to reach a continuum of resistance values between the RESET and the SET states. This feature allows PCM cells to store

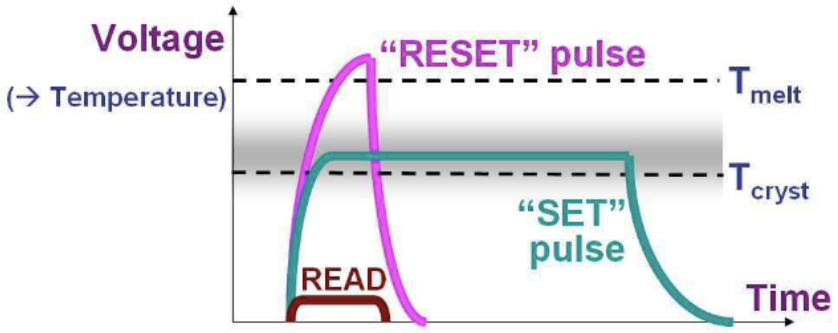

FIGURE 1.9: SET and RESET pulses. Taken from [1].

data in an analog fashion [13]. This possibility is generally obtained by creating intermediate phase configurations in the material by applying suitable partial RESET pulses. For example, Figure 1.10 shows a continuum of resistance levels achieved by applying RESET pulses of varying amplitude (known as partial-RESET programming [30]). The device is first programmed in a fully-crystalline state, after which a sequence of RESET pulses is applied with progressive-increased amplitude. After the application of each RESET pulse, the state of the device is read; the cell resistance, which depends on the size of the amorphous region, accordingly increases with increasing RESET amplitude.

The curve of Figure 1.10 is generally named programming curve, and shows the possibility to increase and decrease the cells resistance by modulating the programming current. Accordingly, it is possible to program a PCM device to a desired resistance value through iterative programming, thus applying several consecutive pulses. In iterative programming, after each programming pulse, a verification phase is performed by reading the resistance of the device. The programming current applied to the PCM device in the next iteration will then be adapted according to the error value between the desired resistance value and the read value. The algorithm runs until the programmed resistance value reaches a value within a predefined margin from the desired value.

Another technique to program PCM devices is represented by the dynamic crystallization, also named partial-SET programming. As shown in Figure 1.11, a progressive reduction of the size of the amorphous region (and therefore of the resistance of the device) can be induced by the subsequent application of SET pulses with the same amplitude.

Although it is possible to achieve a desired resistance value through iterative programming, there are significant temporal fluctuations associated with conductance values, that will be addressed in the next Section, and experimentally characterized in Chapter 2.

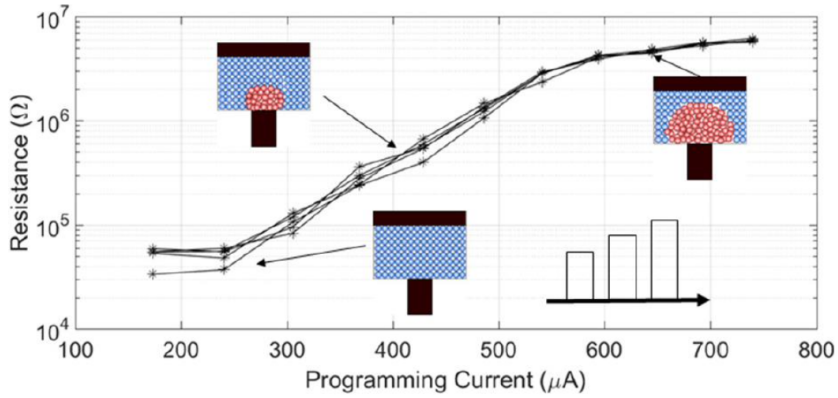

FIGURE 1.10: Partial-RESET programming curve. Taken from [26].

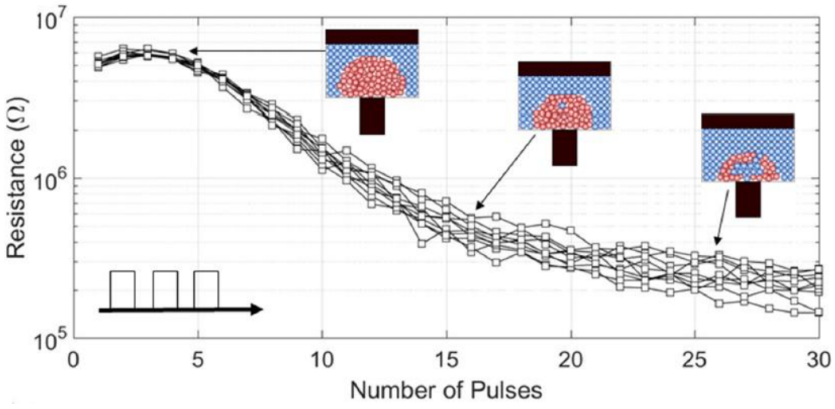

FIGURE 1.11: Partial-SET programming curve. Taken from [26].

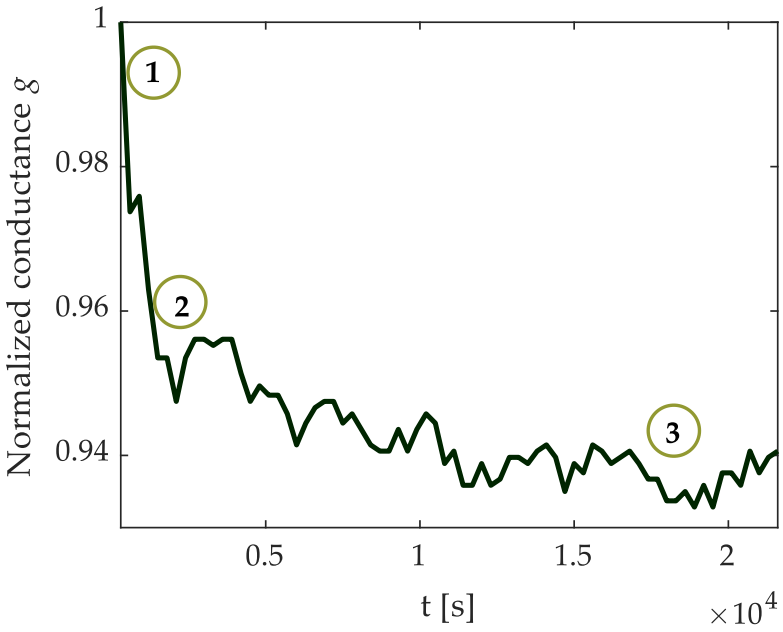

FIGURE 1.12: Measured 14-hours time behavior of a cell normalized conductance showing undesired phenomena: (1) uncertainty of initial value; (2) drift; (3) noise. Taken from [31].

1.5.3 Issues and challenges

The main PCM cells non-idealities, that are relevant when employed in analog computing contexts, are:

- Noise: low-frequency (flicker) noise affects cells behavior, as random electron traps are located in the cell lattice, especially in the amorphous region [32]. Noise is proposed to be generated by variation in the configuration of the amorphous-state traps structure.
- Time drift: cell conductance tends to decrease due to the time-decreasing density of traps of the hopping Pool-Frenkel conduction, typical of the amorphous phase of cells [33]–[36].
- Uncertainty of the programmed conductance level: different cells respond differently to the same programming pulses. Besides, the response of the same cell to subsequent programming cycles shows a large variability. This leads to dispersion and inaccuracy of the conductance levels [14], [28].

To illustrate the above points, the time-behavior of a typical PCM cell is shown in Figure 1.12, where the measured conductance, normalized to its initial value, is reported. Several studies have been carried out to motivate and model the behavior of PCM cells, expecially for what concerns the amorphous phase. Some reference works are [37]–[40], and an important survey on PCM device modeling is presented in [41]. Moreover, new phase-change devices are currently under development [42]. Although relevant research efforts and advances, some questions related to this device technology are still open.

1.6 State-of-the-art AIMC-based units

In this Section, a brief analysis of the most recent AIMC elaboration units is presented.

For what SRAM-based AIMC is concerned, a notable work is presented in [43], where a SRAM macro that computes ternary-MAC operations in binary/ternary DNNs with high energy efficiency and high accuracy. The size of the array is 256×64 and the prototype achieves energy efficiency of 40.3 TOPS/W for MAC operations and 88.8% test accuracy for a CIFAR-10 data set. An additional contribution comes from [10], where the architecture supports analog/binary input activation first layer and binary hidden layers, with batch normalization and input–output buffering circuitry to enable cascading, if desired, for realizing different DNN layers. The energy efficiency is comparable to the previous one, while the array dimension is 2.4 Mb. Both prototypes are realized in a 65-nm CMOS technology node.

An example of AIMC unit exploiting a flash memory is illustrated in [44]. The prototype targets a 28×28 binary-input, ten-output, three-layer neuromorphic network based on arrays of highly optimized embedded nonvolatile floating-gate cells,

redesigned from a commercial 180-nm NOR flash memory. The network has shown a 94.7% classification fidelity, with a 10-TOPS/W energy efficiency.

RRAM devices are exploited in [45] to obtain a high-performance and uniform memristor crossbar array for the implementation of CNNs, which integrates eight 2048-cells memristor arrays to improve parallel-computing efficiency up to a measured 21.9 TOPS/W energy efficiency.

In the field of PCM technology, a recent work [46] shows a 256×256 AIMC core designed and fabricated in a 14-nm CMOS technology. It allows to perform outputs affine scaling and non-linear operations. The measured energy efficiency is 10.5 TOPS/W for a CIFAR-10 classification task.

A summary of the presented works is reported in Table 1.1.

1.7 A brief overview on PCM-based AIMC

In this Section, some of the most recent works and results in the field of PCM-based AIMC are commented.

Circuit solutions are analyzed in [47] and [48]. 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 [49] 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 [50] drift is addressed by renormalizing the drifted MVM output by modeling the median conductance decay and rescaling the argument of the nonlinear activation function

TABLE 1.1: Summary of some state-of-the-art testchips for AIMC.

Reference	[43]	[10]	[44]	[45]	[46]
Employed memory devices	SRAM	SRAM	Flash	RRAM	PCM
CMOS Technology node	65 nm	65 nm	180 nm	55 nm	28 nm
Classification accuracy	88.8%	83.3%	94.7%	88.5%	96.2%
Energy efficiency [TOPS/W]	40.3	658	10	21.9	10.5

following each layer to ensure that the entire nonlinearity domain is excited as expected for non-drifting weights. In [51] 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 [52], a considerable area overhead associated to the AIMC unit [47], [48], reliance on accurate device models [50] or the periodic recalibration of the system [51]. By applying multiple techniques simultaneously the requirements on each of them can be relaxed, with potential reduction of the incurred cost.

1.8 Overview of the thesis

This thesis aims at exploring the potential of new PCM-based architectures as in-memory computational accelerators. The reference technology has been provided by STMicroelectronics, through the joint lab with the ARCES Center of the University of Bologna.

Several aspects will be addressed in order to achieve the desired goal. First of all, current PCM writing cycles are optimized for use as binary memories. Even if a binary PCM is compatible with many applications involving also MVM operations, the real advantage of in-memory computing will be unfolded only if truly analog or at least multilevel resistive values will be achievable. This is a non-trivial step, which requires a new carefully dedicated programming algorithm. Besides, additional problems expected in multilevel PCMs are the spread and time-drift of the programmed resistance values, causing a large variability, as well as the non-linearity of the I-V characteristics of the memory cells.

In some applications, such as DNN, these problems can be mitigated by a spread- and non-linearity-aware learning scheme, i.e. during the phase of weight determination. In a first phase of this work, arrays of conventional binary PCMs will be operated and characterized under non-conventional operating conditions, similar to the ones required for in- memory MVM. The focus will be to estimate and model through suitable compact models the non-linear characteristics of the PCM resistances, as well as their variability. This will allow to simulate MVM operations under realistic conditions. In addition, this will allow to include realistic MVM models into specific high-level software for DNN description and training.

As a result of this first phase, a first small-size array architecture for in-memory MVM will be proposed, designed and laid-out in the given technology. A key point will be the design of the analog blocks (drivers, converters, voltage and current reference generators) necessary for the non-conventional operation of the array. A deep experimental characterization and validation of the architecture will be carried out and discussed.

This thesis is organized as follows:

- In **Chapter 2**, a thorough characterization of PCM cells is presented, aimed at evaluating and optimizing their performance as enabling devices for analog in-memory computing applications.
- In **Chapter 3**, the use of PCM cells in two different applications is simulated, with the aim of quantifying the impact of PCM cells non-idealities when employed to perform Multiply and Accumulate operations.
- **Chapter 4** presents an integrated peripheral unit interfaced to an embedded Phase-change Memory macrocell, with the aim of adding analog in-memory computing feature without any modifications to the internal structure of the memory array. Experimental characterizations are carried out to validate the testchip, and to simulate its employment in a deep neural network scenario.

This research activity have been carried out with a fundamental support from STMicroelectronics Italy, who first provided the evaluation board and the memory samples used in Chapter 2; several designers contributed to the development of the AIMC testchip presented in Chapter 4.

Part of the results of this work have been obtained thanks to a tight collaboration with other academic research groups; in particular, results related to the deep neural networks have been reached with the Signal Processing research group of both University of Bologna (Prof. Mauro Mangia and Prof. Riccardo Rovatti) and Politecnico of Torino (Dott. Carmine Paolino, Prof. Fabio Pareschi and Prof. Gianluca Setti). A collaboration with the Structural Engineering group of the University of Bologna resulted in the analyses in the field of structural health monitoring (Dott. Said Quqa and Prof. Luca Landi). My contributions focused on the PCM characterization, together with the collection of the experimental data for the proposed applications. A relevant part of my Ph.D. activity was also involved in the testchip design and testing. This research activity has been carried out with my colleagues Dott. Andrea Lico and Dott. Francesco Zavalloni.

Chapter 2

PCM cells characterization for analog in-memory computing

In this Chapter, a thorough characterization of phase-change memory (PCM) cells is carried out aimed at evaluating and optimizing their performance as enabling devices for analog in-memory computing (AIMC) applications. Exploiting the features of programming pulses, strategies to reduce undesired phenomena that afflict PCM cells and are particularly harmful in analog computations, such as low-frequency noise, time drift and cell-to-cell variability of the conductance, are discussed. The test vehicle is an embedded PCM (ePCM) provided by STMicroelectronics and designed in 90-nm smart power BCD technology with a Ge-rich Ge-Sb-Te (GST) alloy for automotive applications. Based upon the results of the characterization of a large number of cells, an iterative algorithm is proposed to allow multi-level cell conductance programming and its performances for AIMC applications are discussed. An analysis of the effects of time-temperature effect on cells in terms of drift and noise concludes the Chapter.

Some of the material reported in this Chapter is reused from [31] (open access), and from [53], in agreement with MDPI and IEEE copyright on theses and dissertations.

2.1 Experimental setup

2.1.1 PCM Testchip

We performed the experimental activity on an embedded PCM (ePCM) test chip designed and manufactured by STMicroelectronics in 90-nm smart power BCD technology featuring a specifically optimized Ge-rich Ge-Sb-Te (GST) alloy. The chip is intended for digital storage in automotive applications. The ePCM elementary cell is based on an nMOS selector and occupies 0.19 μm^2 of silicon area [29]. A 256-KB macrocell was included in the test chip in 8 independent instances in order to increase the total number of cells in a single chip. In addition to the 8 ePCM macrocells, the chip also includes a built-in self-test (BIST) block, several configuration registers, a reference generator block, and the circuitry that manages the input–output interface [54].

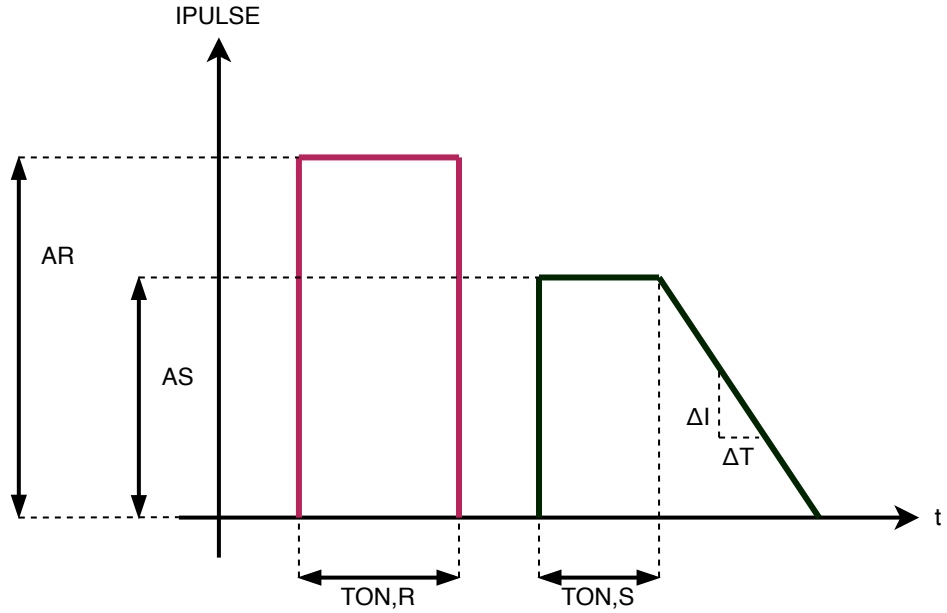

FIGURE 2.1: SET and RESET pulses and their editable parameters.

2.1.2 Implemented testing routines

A PCM evaluation board (properly designed for testing purposes) was employed and customized. This board allows one to configure current pulses applied to cells, as voltage and current regulators are integrated on the test chip. Furthermore, it is possible to measure the current of single or multiple cells thanks to an analog chip-board interface and a dedicated I-V conversion chain. Every programming or measurement process is achieved with a GUI interface, which is available on a personal computer and customizable.

Several improvements to the GUI have been developed, in order to implement dedicated testing routines. In particular, iterated-measures environments to characterize PCM in terms of drift and noise have been exploited. Furthermore, an interface to perform precise current-measurements through a Source Meter Unit (SMU) has been created.

Finally, the evaluation board was equipped with analog to digital converters that allow for the measured current to be stored and elaborated. A photo of the experimental setup is reported in Figure 2.3, whereas a representation of the GUI interface is shown in 2.2.

2.1.3 Programming pulses parameters

Cell transition between SET state and RESET state is accomplished with the application of a corresponding current pulse [14], [24], [27], which causes a significant portion of the cell to be heated, in order to modify its internal structure:

- a SET pulse is a trapezoidal current pulse, composed of an initial melting phase, followed by a slow crystallization phase;

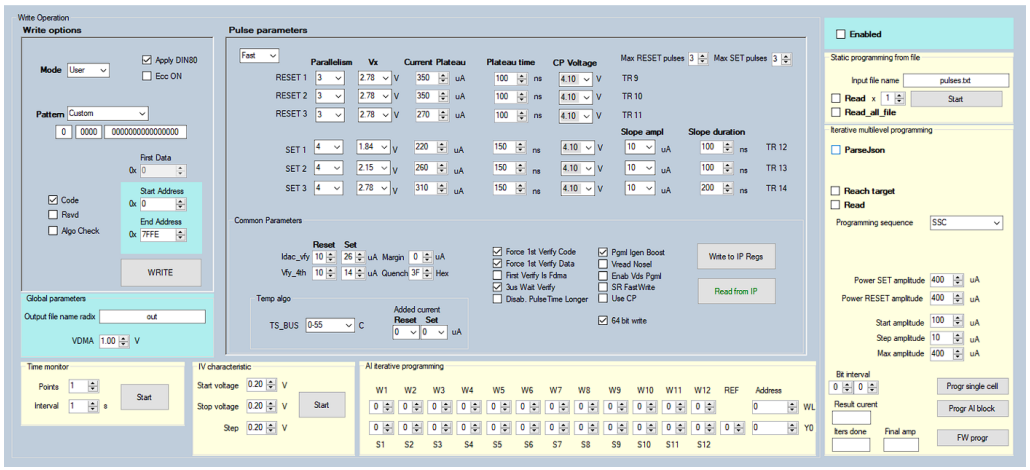

FIGURE 2.2: Snapshot of the developed GUI for PCM cells characterization.

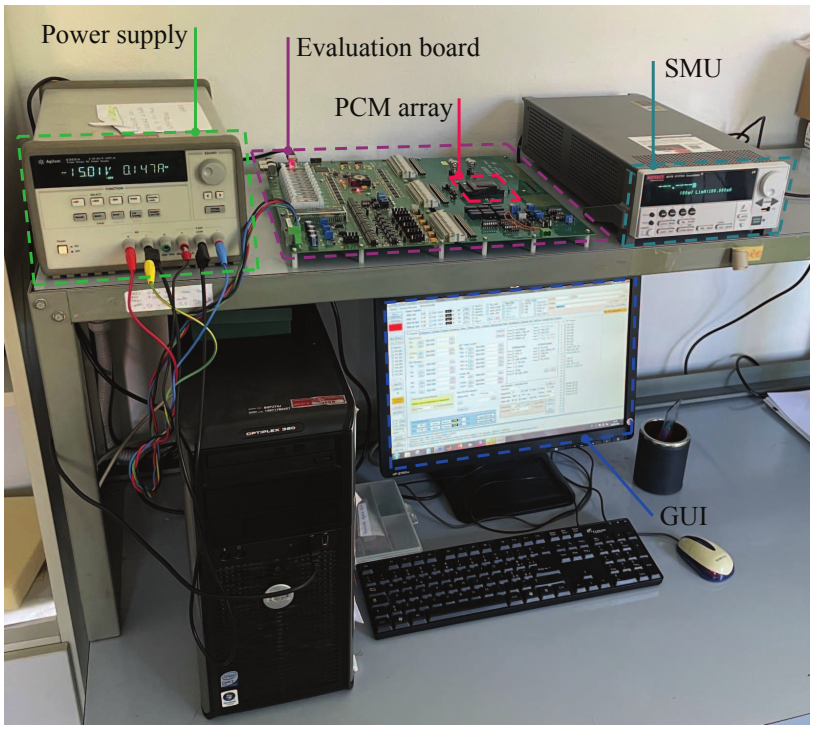

FIGURE 2.3: Experimental setup for PCM cells testing.

- a RESET pulse consists in a higher current flow and it is applied in order to melt the central portion of the cell. The molten material quenches into the amorphous phase, producing a cell in the high-resistance state.

The possibility to set the cell in a wide range of intermediate conductance states is achieved through an adequate control of different configurations of the crystalline and amorphous phases inside the active chalcogenide volume: in other terms, the cell resistance value depends on the shape and the volume of the two phases. The main aim of our set of measurements was to investigate the impact of the different pulse parameters and the associated programming sequences on cells noise, drift, and conductance variability. The pulse parameters that are editable through the evaluation board are indicated in Figure 2.1, namely:

- the SET pulse can be modulated in amplitude (A_S), width of the flat portion ($T_{ON,S}$), and decaying slope ($\Delta I/\Delta T$);
- the RESET pulse can be modulated in amplitude (A_R) and width $T_{ON,R}$.

The editable minimum, maximum, and step values of each parameter are reported in Table 2.1.

2.1.4 Readout voltage choice

The available hardware allows current measurements through the application to one or more cells of a readout voltage V_R, ranging from 0 to V_R^{MAX}. The measured average $i(v)$ characteristic of a group of PCM cells is depicted in Figure 2.4, where i is the cell current normalized to its maximum value, and v is defined as V_R/V_R^{MAX}. The average normalized conductance $g = i/v$ is nearly constant when V_R falls within $[0 - 0.4]V_R^{MAX}$; above $V_R = 0.5V_R^{MAX}$, the voltage applied to cells differs from V_R due to voltage drops of the transistors in the test chip readout circuitry. Therefore, due to test chip implementation, for the operation described in (2), $V_{k=1,...,N}$ will be limited within the range $[0 - 0.4]V_R^{MAX}$. All measurements described hereafter are performed in the middle of that interval, namely, $V_R = 0.25V_R^{MAX} \doteq V_X$.

TABLE 2.1: Configurable parameters of SET and RESET pulses.

Parameter	Minimum	Maximum	Resolution	Order of magnitude
$T_{ON,S}$	$T_{ON,S0}$	$2T_{ON,S0}$	$T_{ON,S0}/2$	100 ns
ΔI	ΔI_0	$2\Delta I_0$	ΔI_0	10 μA
ΔT	ΔT_0	$2\Delta T_0$	$\Delta T_0/2$	10 ns
A_R	A_{R0}	$6A_{R0}$	$A_{R0}/10$	10 μA
$T_{ON,R}$	$T_{ON,R0}$	$2T_{ON,R0}$	$T_{ON,R0}/10$	10 ns

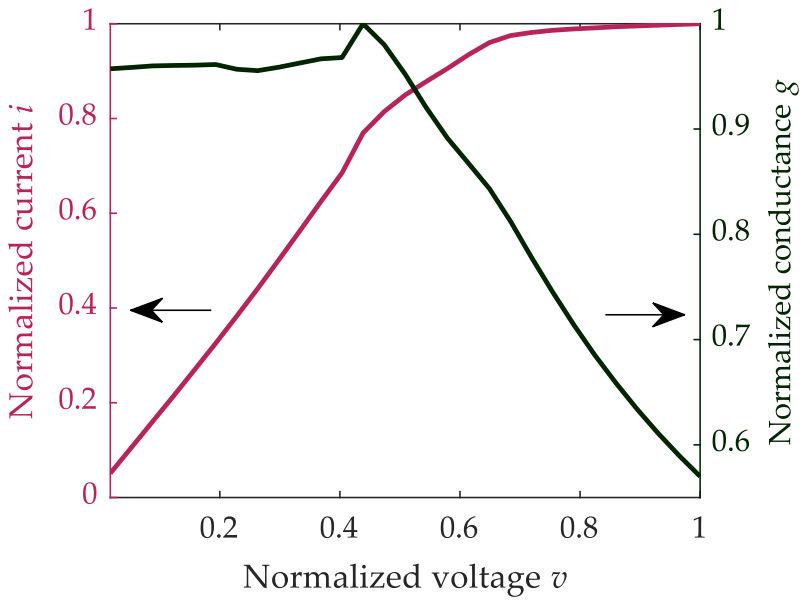

FIGURE 2.4: Left axes: typical normalized I-V characteristic obtained by averaging the currents of 5120 cells. Right axes: normalized cells mean conductance $g = i/v$.

2.2 PCM cells characterization using single-SET pulses

In this Section, a characterization in terms of drift and noise is carried out. Cells were programmed through a single SET pulse. The following analyses were performed considering 5120 cells. Henceforth, conductances G are normalized to cell maximum conductance G_{MAX}, and their currents I to $I^{\mathrm{MAX}} = G^{\mathrm{MAX}} V_{\mathrm{X}}$, turning in cells normalized conductance $g = G/G_{\mathrm{MAX}}$ and normalized current $i = I/I^{\mathrm{MAX}}$, respectively. All the measurements, unless otherwise specified, were performed at room temperature.

2.2.1 Noise

As previously observed, lattice imperfections and traps contribute to generate low-frequency noise, which affects the analog computation process [33], [55]–[57]. Tests were performed in the following way: first, a start RESET pulse with $A_{\mathrm{R}} = 3A_{\mathrm{R0}}$ and $T_{\mathrm{ON,R}} = T_{\mathrm{ON,R0}}$ was applied to erase the previous state, followed by a SET pulse with $T_{\mathrm{ON,S}} = 2T_{\mathrm{ON,S}}$, $\Delta I = \Delta I_0, \Delta T = \Delta T_0$. Four different values of A_{S} were considered: A_{S0}, $1.5A_{\mathrm{S0}}$, $2A_{\mathrm{S0}}$, and $3A_{\mathrm{S0}}$. To limit the time drift contribution, we performed measurements 12 h after the application of the SET pulse. Then, $S_{\mathrm{TOT}} = 188$ current samples were collected for each cell at time intervals of 5 min t_i. We evaluated the noise parameter $N_{\%j}$ of the j-th cell as:

$$N_{\%,j} = \frac{100}{\overline{g_j}} \sqrt{\frac{1}{S_{\mathrm{TOT}} - 1} \sum_{i=1}^{S_{\mathrm{TOT}}} \left[g_j(t_i) - \overline{g_j}^2 \right]} \tag{2.1}$$

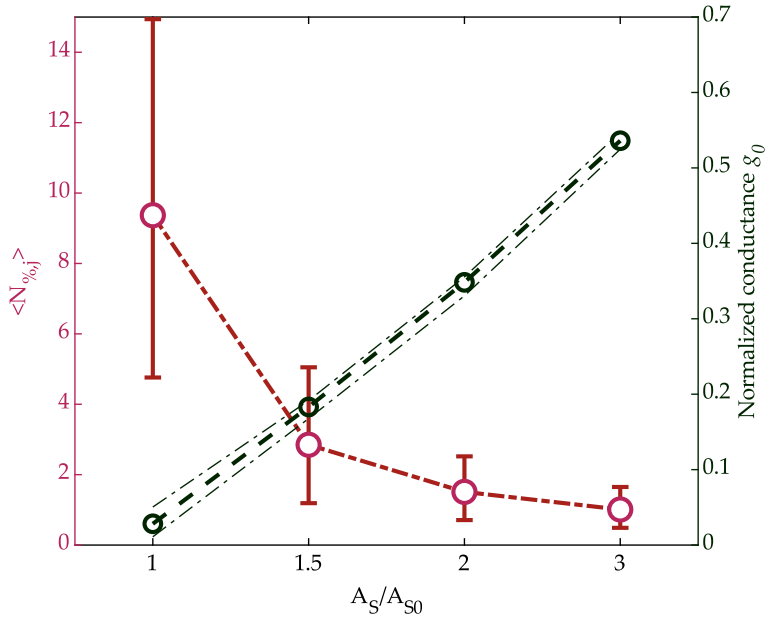

FIGURE 2.5: Left: ensemble average over all the tested cells of $N_{\%,j}$ defined in 2.1 vs. SET pulse amplitude. Right: normalized conductance averaged on both time and cells.

where $g_j(t_i)$ is the j-th cell normalized conductance at time t_i, and $\overline{g_j}$ is the time average of $g_j(t_i)$. The ensemble average $\langle N_{\%,j} \rangle$ over all the tested cells is shown in Figure 2.5 with red circles as a function of the amplitude A_S, together with the indication of the 10% and 90% limits of the distribution. On the right vertical axis the cell conductance averaged on both $S_{TOT} = 188$ time samples t_i and the 5120 measured cells is also shown. The conductance is proportional to the SET amplitude, as expected, since a higher amplitude implies the crystallization of a wider cell volume. This leads to a reduction of noise, as its origin is mainly correlated to the lattice disordered structure of the amorphous phase [30], [56], [57].

We then investigated the possibility of noise reduction by means of summing the current contributions of adjacent cells programmed in the same SET state. Measurements have been performed with groups of 2, 4 or 8 Adjacent Working Cells (AWC). To do that, previous measurements have been repeated on a set of AWC×5120 cells, and N% has been evaluated as in 2.1 but replacing $g(t_i)$ with the average of AWC cells for each sample time. Results are shown in Figure 2.6 (left) as a function of AWC for different pulse amplitudes. If noise of different cells were totally uncorrelated, the curves would depend on AWC as $1/\sqrt{\text{AWC}}$ (reported in the figure as solid lines). As AWC > 1 for a given pulse amplitude results in an increase of power consumption, it is interesting to compare the cases AWC $= 1$ and AWC > 1 for the same normalized total current consumption. In Figure 2.6 (right) the ensemble average noise $\langle N_{\%,j} \rangle$ is reported as a function of the normalized total current for different AWC. It is clear that the AWC > 1 strategy is not convenient when power consumption is considered. In other words, for a given total current, a single cell achieves

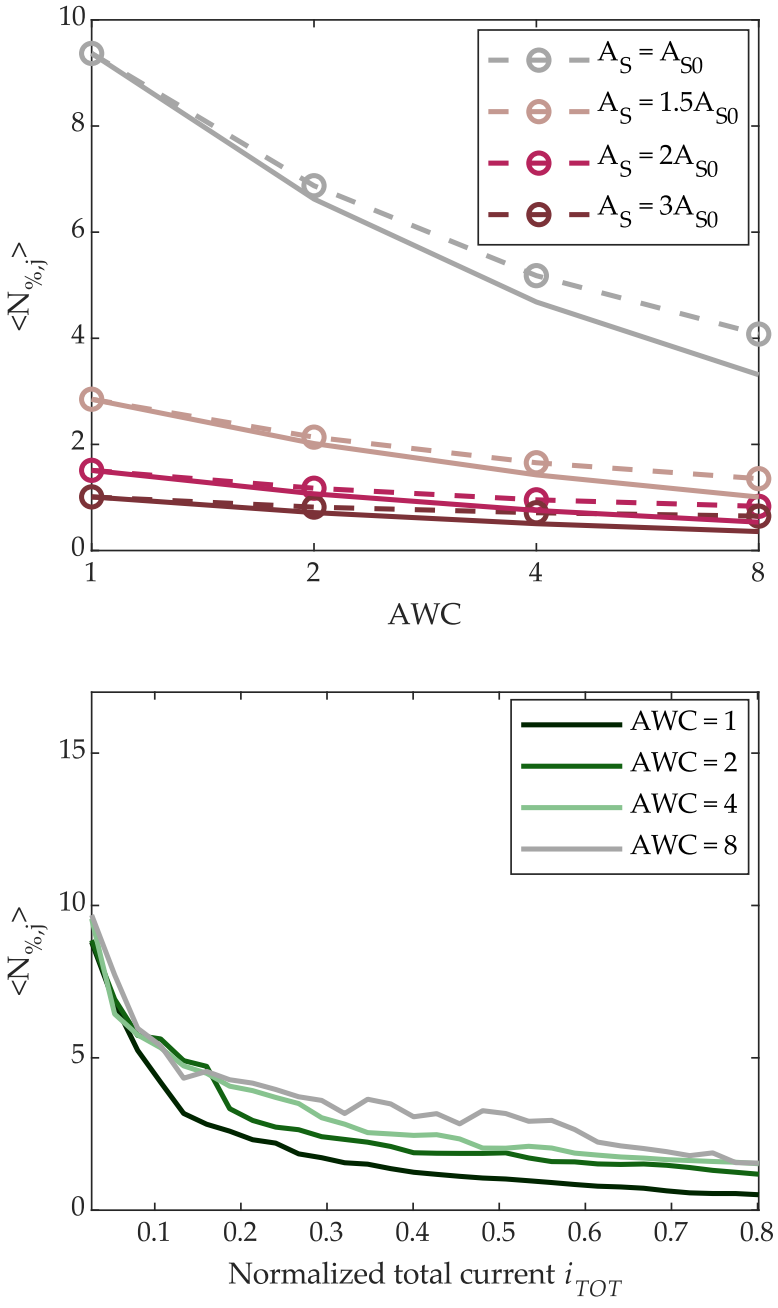

FIGURE 2.6: (Left) Dotted lines: measured ensemble average $\langle N_{\%,j} \rangle$ of $N_{\%,j}$ defined in 2.1 vs. AWC for different SET pulse amplitudes. Solid lines: theoretical $1/\sqrt{\text{AWC}}$ noise behavior. (Right) $\langle N_{\%,j} \rangle$ vs. normalized total current for different AWC values.

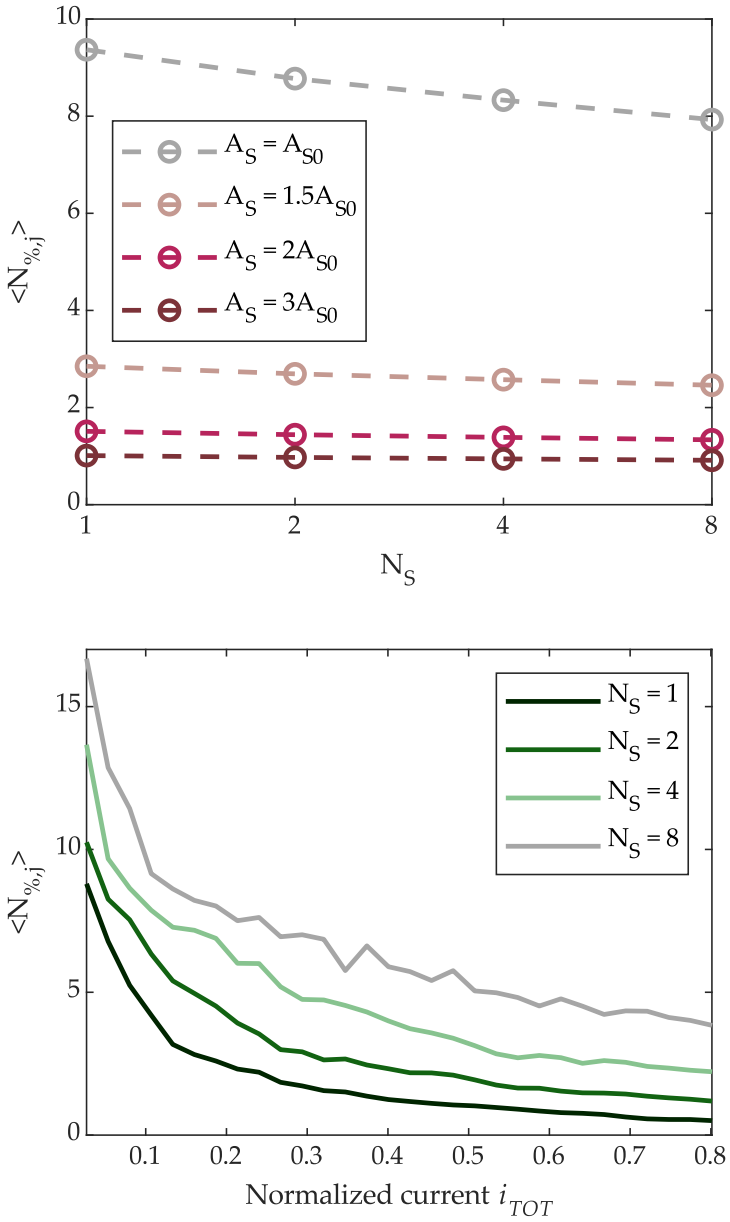

FIGURE 2.7: (Left) Measured ensemble average $\langle N_{\%,j} \rangle$ of $N_{\%,j}$ defined in 2.1 vs. number NS of samples in the averaging window for different SET amplitude pulses. (Right) $N_{\%,j}$ vs. normalized total current for different NS values.

more noise reduction than several cells in parallel with lower conductance. For these reasons, the characterizations presented hereafter are performed with AWC = 1. Finally, we explored the possibility to reduce noise through a time average operation. To this purpose, the previous measurements have been repeated and $N_{\%}$ has been calculated replacing in Figure 2.1 each $g(t_i)$ with the average over NS consecutive samples equally separated in time by $\Delta t = 5\text{min/NS}$, with NS = 1, 2, 4 or 8. Results are shown in Figure 2.7 (left), where a slight reduction of noise is visible, in particular in the A_{S0}-SET case. In analogy with the AWC strategy, it is necessary to consider the additional power consumption introduced by the NS-oversampling operation. N% as a function of the normalized total current is shown in Figure 2.7 (right) for the different values NS. It is seen that time average is not effective to reduce noise for a given total current. This can be understood taking into account the flicker nature of PCM cells noise [33], [55], [57], as time average operation is equivalent to a low-pass filter in the frequency domain.

A dependence of $\langle N_{\%,j} \rangle$ on SET pulse amplitude, AWC number and time average, similar to the ones discussed in Figures 2.5, 2.6, 2.7, is obtained varying $T_{\text{ON,S}}$, $\Delta\text{I}/\Delta\text{T}$. To conclude, the most efficient strategy to reduce noise is to use a single cell with a higher conductance for each matrix element.

2.2.2 Time drift

Short term drift manifests itself as a slow but steady increase of the resistivity of the amorphous material. The conductance $g(t)$ drift has been shown to follow a power law [33]:

$$g(t) = g_0 \left(\frac{t}{t_0} \right)^{-\alpha} \tag{2.2}$$

where g_0 is the initial conductance at arbitrary time t_0, and α is the drift coefficient, which is positive and cell-to-cell variable.

In this work, instead of exploiting such power-law model, drift is evaluated in terms of relative conductance decrease $D_{\%,j}$ of the j-th cell as:

$$D_{\%,j}(t_i) = 100 \frac{g_{j,0} - g_j(t_i)}{g_{j,0}} \tag{2.3}$$

where $g_j(t_i)$ is the j-th cell normalized conductance at time t_i and $g_{j,0}$ its value measured 1 ms after the pulse application. We first investigated the effect of SET-pulse amplitude on D%. To do that, we programmed 5120 cells in the same way explained in the previous paragraph, then, we monitored them for T = 14 hours. The average $\langle D_{\%,j}(T) \rangle$ over all the tested cells as a function of the SET-amplitude is shown in Figure 2.8 (left) with red bullets as a function of the amplitude A_S, and the indication of the 10% and 90% limits of the distribution are also shown. On the right vertical axis the cell normalized mean conductance is plotted. Results show that the increase of SET-amplitude reduces cells drift below 8% for $A_S = 3A_{S0}$.

An additional result is reported in Figure 2.8 (right), where D% for each cell is plotted vs. g_0 for different pulse amplitudes. It can be observed that cells with the same initial conductance g_0 have a lower drift when g_0 has been reached by applying a higher SET pulse.

2.3 PCM cell characterization using multiple pulses

In this Section we investigate the use of specific sequences of multiple current pulses to tune the cell conductance as close as possible to the desired level, while limiting noise, drift and variability.

2.3.1 Conductance tunability

Cells reaction following the application of both a SET or a RESET pulse shows an uncertainty due to random amorphization and crystallization phenomena. The programming space is defined by the characteristic programming curve, which quantifies the change of the cell (normalized) conductance as a function of the programming pulse cur-rent. In the literature, two approaches have been proposed in order to program the cell resistance to an intermediate level: (a) partial-SET programming [58] and (b) partial-RESET programming [56], [57]. In the first approach, the cell is first brought into the RESET state, and then, a partial-SET programming pulse is applied so as to partially crystallize the ac-tive volume. In partial-RESET programming, the cell is first brought into the SET state, and then, a partial-RESET pulse is applied in order to partially amorphize the active volume. Based on these two approaches, we experimented four different programming strategies and derived the corresponding programming curves. The adopted programming se-quences are illustrated in Figure 2.9: (a) RESET single pulse programming (RSP); (b) RESET staircase programming (RSC); (c) SET single pulse programming (SSP); (d) SET staircase (SSC) programming.

In the RSP case (Figure 2.9 (a), first a SET pulse with $A_S = 5A_{S0}, T_{ON,S} = 2T_{ON,S0}, \Delta I = \Delta I_0, \Delta T = \Delta T_0$ is applied, followed by a single partial-RESET pulse having a predetermined amplitude AR and width TON,R, and then, after 1 ms, a readout operation is performed. The above sequence is repeated with increasing values of A_R between A_{R0} and $4A_{R0}$ with steps of $\sim A_{R0}/10$. In the RSC case (Figure 2.9 (b)), a single start SET pulse with the same parameters mentioned above is applied only at the beginning, followed by a partial-RESET sequence identical to the one in the RSP case, with readout operations performed after each specific RESET pulse.

Results of RSP and RSC are illustrated in Figure 2.10 (a) and 2.10 (b), respectively, where the mean conductance of NC = 5120 cells is plotted as a function of A_R for different values of $T_{ON,R}(T_{ON,R0}, 1, 5T_{ON,R0}, 2T_{ON,R0})$. The behavior of cells in RSP mode shows an initial increase of conductance, due to the fact that small amplitude RESET pulses tend to be similar to a SET pulse. Then, when $A_R > 2A_{R0}$, cells

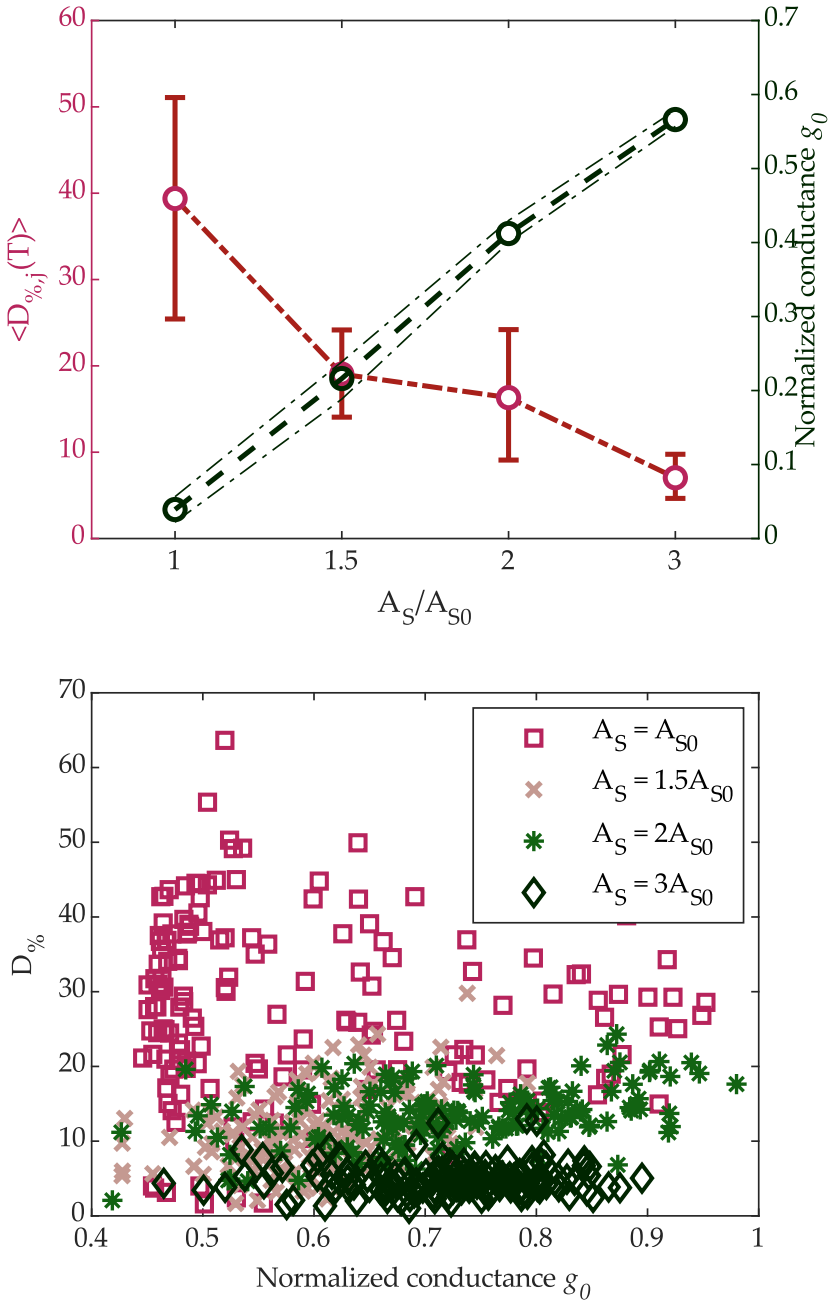

FIGURE 2.8: (Left) Ensemble average of D% defined in 2.3 with T = 14 h vs. SET pulse amplitude; Right: mean value of the normalized conductance measured after the application of SET pulse. (Right) D% defined in 2.3 vs. normalized initial conductance g_0 for different SET pulses amplitudes. Measures have been taken over a set of 960 cells.

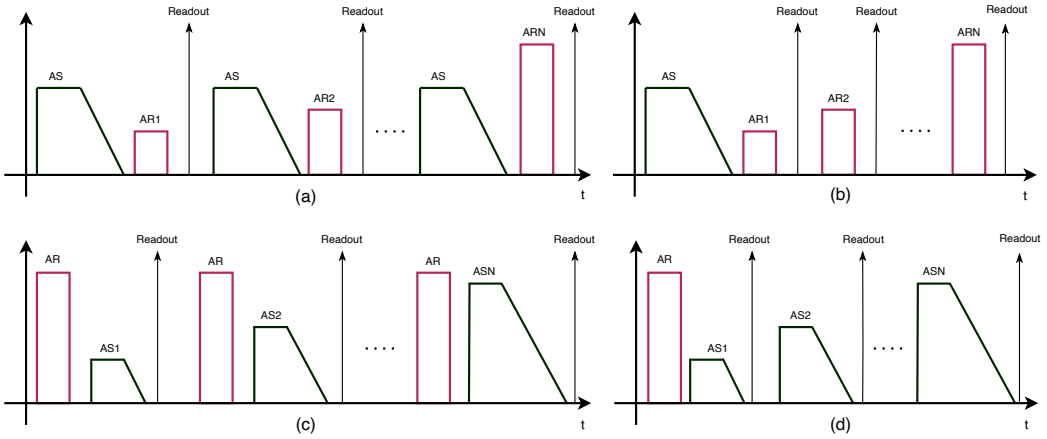

FIGURE 2.9: Analyzed programming sequences: (a) RESET single pulse (RSP); (b) RESET staircase (RSC); (c) SET single pulse (SSP); (d) SET staircase (SSC).

conductance begins to decrease. This initial in-crease of the conductance value is absent in RSC mode. In both families of programming curves, the mean normalized conductance g slightly depends on $T_{ON,R}$, whose value tends to increase the mean conductance of cells, as the RESET pulse is longer and tends to be more similar to a SET one. Furthermore, the programming curves for RSP or RSC are quite similar when $A_R > 2A_{R0}$ both being characterized by an abrupt decrease to a full RESET state. For what concerns partial-SET programming, in the SSP case (Figure 2.9 (c)) a start RESET pulse with $A_R = 3A_{R0}$ and $T_{ON,R} = 2T_{ON,R0}$ is applied, followed by a single partial-SET pulse and a readout operation. The sequence is repeated with A_S varying from A_{S0} to $4A_{S0}$ in steps of $\sim A_{S0}/10$. Adopted values of $T_{ON,S}$ are $T_{ON,S0}, 1.5T_{ON,S}$ and $2T_{ON,S0}$. We chose $\Delta I = \Delta I_0$ and $\Delta T = \Delta T_0$ for all measurements. The SSC case (Figure 2.9 (d)) is similar, but the start RESET pulse is applied only at the beginning.

As before, the mean conductance of 5120 cells has been monitored. Results are reported in Figure 2.11 (a) and 2.11 (b) for the SSP and SSC cases, respectively. In these cases the conductance is not significantly influenced by the value of $T_{ON,S}$, except for the lowest value of $T_{ON,S}$ in the SSP case. On the other hand, as opposed to the partial-RESET strategy, differences between the two sequences are indeed more visible: the SSC conductance tends to increase faster, reaching values above 90% of G^{MAX} with a lower SET amplitude ($A_S = 2.2A_{S0}$), whereas the SSP conductance reaches the same level only with a $3A_{S0} - 3.5A_{S0}$ SET pulse.

Comparing partial-RESET and partial-SET strategies, we can point out that RSP and RSC lead to abrupt programming curves, whereas partial-SET programming allows a smoother control of the conductance by means of the SET amplitude. Thus, in view of a good conductance controllability, the partial-SET approach is preferable.

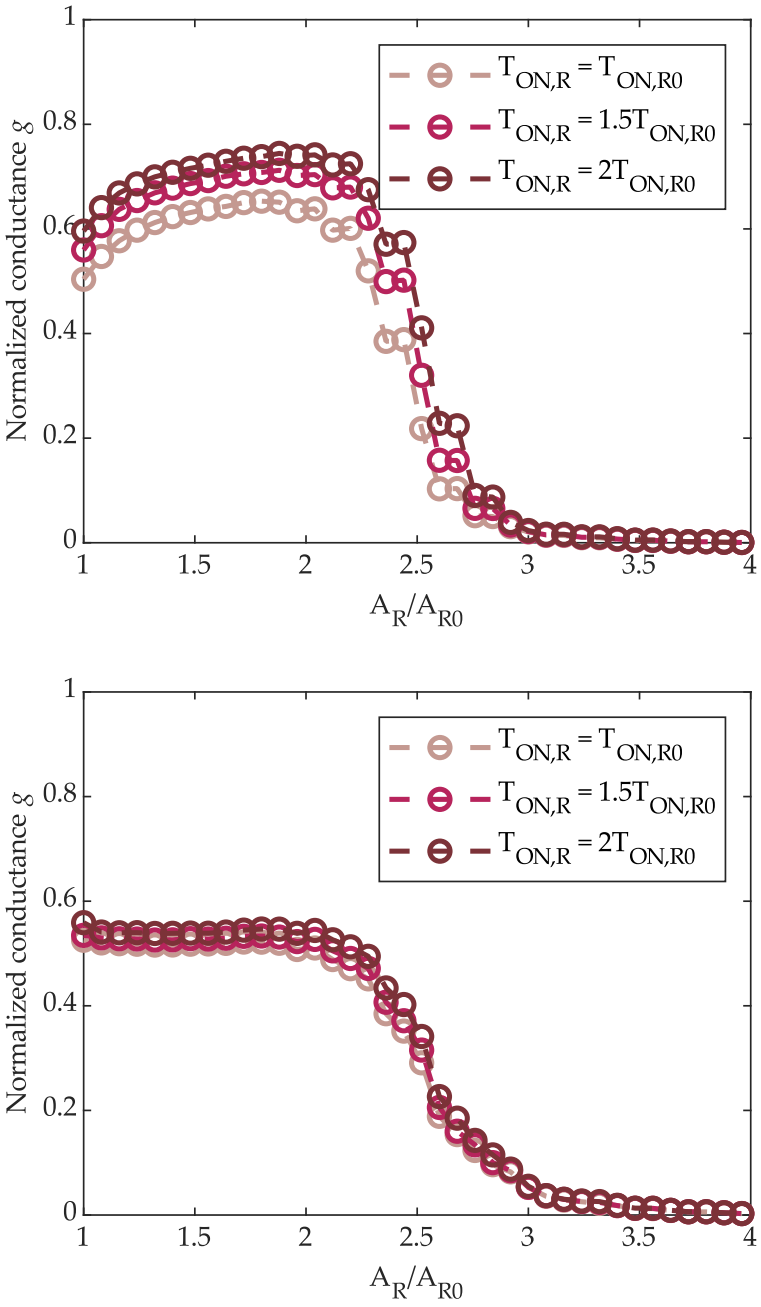

FIGURE 2.10: (a) RSP programming curves as a function of RESET pulse amplitude, with different $T_{ON,R}$ values. The generic $g(A_{R,i})$ represents cells normalized mean conductance after the application of a start SET pulse and a RESET pulse with amplitude $A_{R,i}$. (b) RSC programming curves as a function of RESET pulse amplitude, with different $T_{ON,R}$ values. The generic $g(A_{R,i})$ represents cells normalized mean conductance after the application of a start SET pulse and a sequence of RESET pulses with amplitude from A_{R0} to $A_{R,i}$.

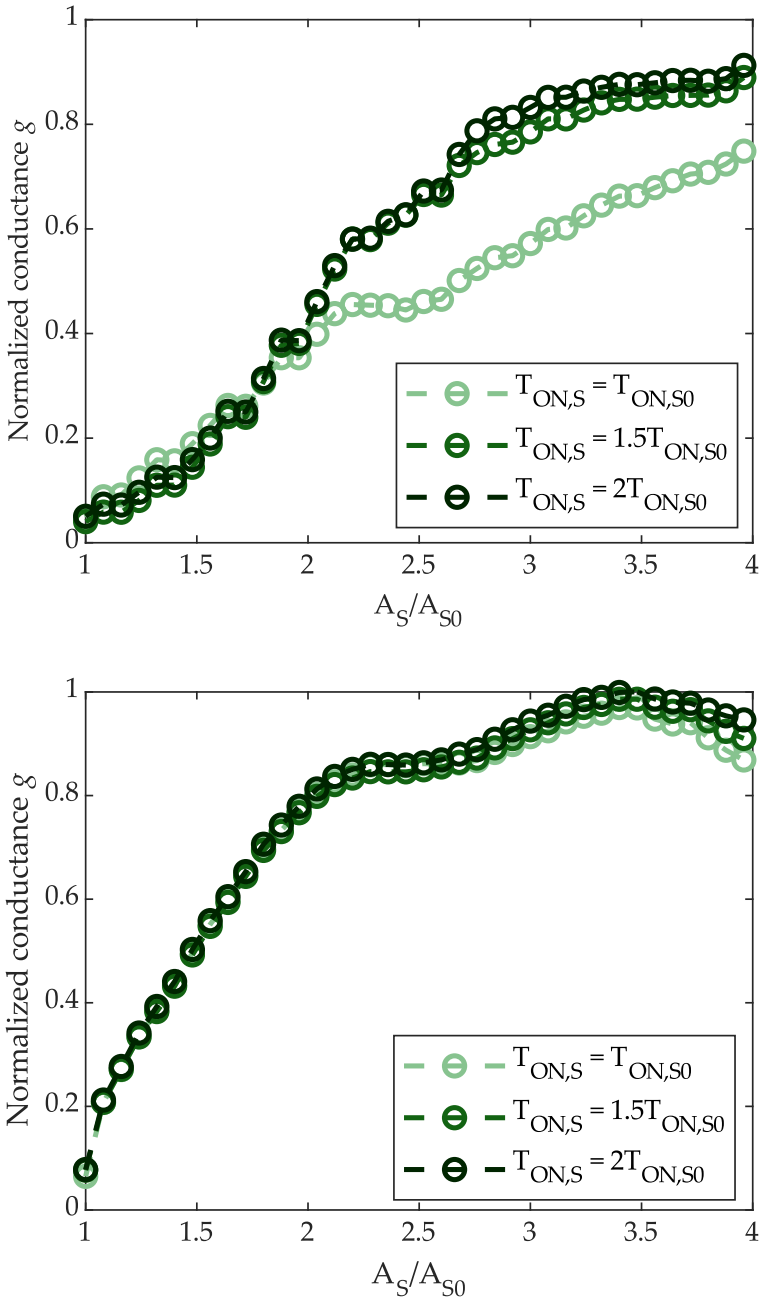

FIGURE 2.11: (a) SSP programming curves as a function of SET pulse amplitude, with different $T_{ON,S}$ values. The generic $g(A_{S,i})$ represents cells normalized mean conductance after the application of a start RESET pulse and a SET pulse with amplitude $A_{R,i}$. (b) RSC programming curves as a function of SET pulse amplitude, with different $T_{ON,S}$ values. The generic $g(A_{S,i})$ represents cells normalized mean conductance after the application of a start RESET pulse and a sequence of SET pulses with amplitude from A_{S0} to $A_{S,i}$.

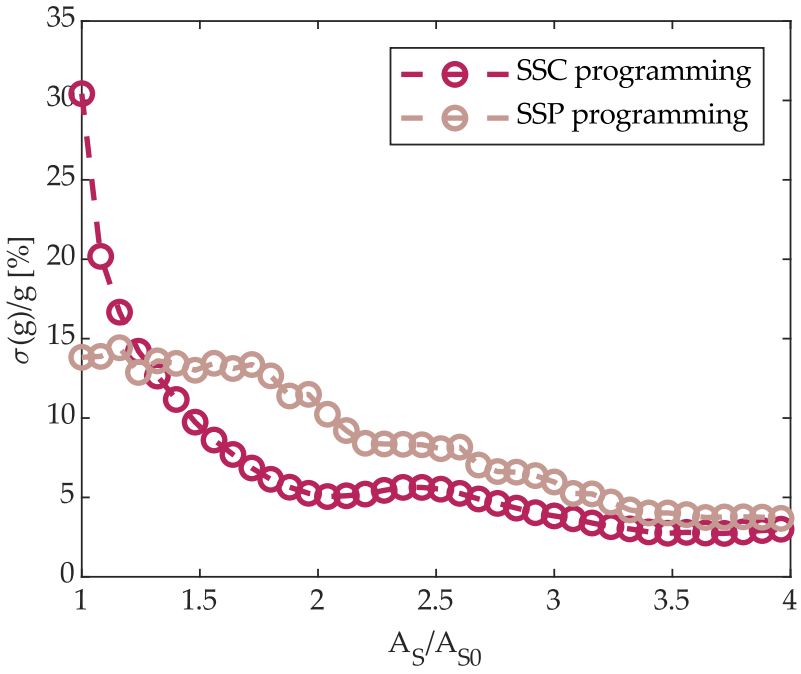

FIGURE 2.12: Normalized standard deviation $\sigma(g)/g$ defined in 2.4 as a function of SET pulse amplitude for both SSP and SSC programming.

We also investigated the conductance spread induced by partial-SET programming evaluating the normalized conductance dispersion $\sigma(g)/g$ at each SET amplitude step $A_{S,i}$ defined as:

$$\frac{\sigma(g)}{g}(A_{S,i}) = \frac{100}{\langle g_j(A_{S,i})\rangle}\sqrt{\frac{1}{NC-1}\sum_{j=1}^{NC}\left[g_j(A_{S,i}) - \langle g_j(A_{S,i})\rangle\right]^2} \tag{2.4}$$

where the mean $\langle g_j(A_{S,i})\rangle$ is calculated over the full set of NC = 5120 cells after the application of the $A_{S,i}$-amplitude SET pulse. Results depicted in Figure 2.12 show that SSC programming leads to a lower spread when $A_S > 1.4A_{S0}$. Additionally, SSP programming turns out to be more power hungry, as it requires a greater amount of RESET applied pulses than the SSC one to reach the same value of g.

We finally investigated the effect of the amplitude of the start RESET pulse on the SSC programming curve. Results are shown in 2.13, where g vs. A_S/A_{S0} for $T_{ON,S} = 1.5T_{ON,S0}$ is plotted for $AR = 3A_{R0}, 4A_{R0}$ or $5A_{R0}$. It is seen that the conductance tends to increase more slowly for larger A_R. In turn, larger SET pulse amplitudes are required to reach the same conductance level when A_R is larger. Therefore, the choice of the start RESET pulse amplitude plays an important role in the programming curve, and this property will be exploited in the next Paragraph. To sum up, SSC programming seems to be the most convenient programming strategy, as it allows both good conductance control and spread reduction.

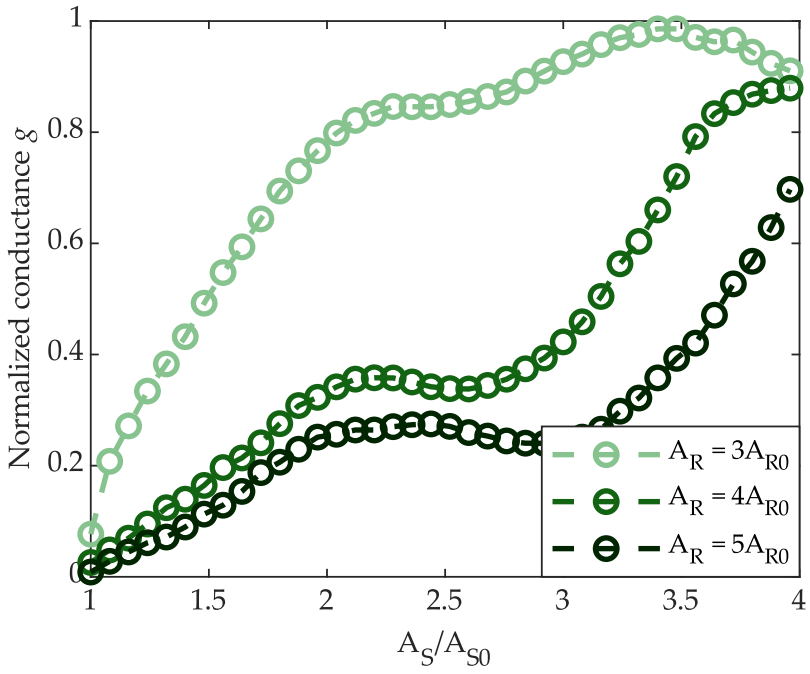

FIGURE 2.13: SSC programming curves as a function of SET pulse amplitude, with different values of the start RESET pulse amplitude.

2.3.2 Drift-induced dispersion

The cell-to-cell conductance spread, which is initially determined by the finite resolution of the programming algorithm (see next Section), tends to increase with time due to the cell-to-cell spread of the drift process described by the parameter D% defined in (4). To investigate such drift spread we have characterized the D% distribution, with the aim of optimizing the programming parameters in order to reduce its standard deviation $\sigma(D\%)$. To this purpose, 5120 cells have been programmed with an SSC strategy. After that, cells conductances have been measured firstly after 14 hours at room temperature (around 25°C), and then after having heated the whole test chip to 150°C for 48 hours in a controlled climate chamber, to emulate the maximum drift achievable by cells [59].

Figure 2.14 (a) shows the values of the measured normalized cells conductances as a function of their initial normalized conductance g_0 after the first and the second time interval. Among the resulting conductivities, a set of four increasing normalized conductivity values (g_0 = 1/6, 1/3, 1/2, 2/3) has been chosen. Figure 2.14 (b) reports the distribution of D% for such values of initial conductivity $g_0 \pm 10\%$, where the top and the bottom plot refers to the first and the second measure, respectively. Results show that after 14 hours the mean value of D% is quite independent of initial conductance value g_0, while its dispersion tends to decrease for higher values of g_0. After 48-hours bake, both the mean value and dispersion of D% are increased with respect to the first measure, and tend to decrease for higher values of g_0, as can be observed also from 2.14 (a).

Previous results on D% in have shown that a drift reduction is achievable using SET pulses of higher amplitude (see Figure 2.13). So, as observed at the end of Section 4.1, we can use a higher-amplitude start RESET pulse in the SSC sequence to reach the same desired conductance with higher partial-SET pulses. Thus, we repeated the D% dispersion analysis by increasing the start RESET pulse amplitude to $5A_{R0}$, instead of the $3_{AR,0}$ used for the results of Figure 2.14 (a) and Figure 2.14 (b). Moreover, as suggested in [27], an additional $5A_{S0}$ start SET pulse was applied before the start RESET pulse, with the aim of obtaining a more uniform cell initialization. The improvements induced by these choices are clearly visible in Figure 2.15 (a) and Figure 2.15 (b), to be compared with Figure 2.14 (a) and 2.14 (b), for each g_0: the average value of D% is strongly reduced and the dispersion of D% is quite reduced.

For the sake of completeness, we also performed measurements by varying the duration of the start SET pulse ($T_{ON,S}, 1.5T_{ON,S}$ and $2T_{ON,S}$), as well as those of the start RESET pulse ($T_{ON,R}, 1.5T_{ON,R}$ and $2T_{ON,R}$), but results did not significantly differ from those reported here. The impact of high-amplitude SET pulses on endurance is not a severe constraint from the AIMC applications where a large amount of write cycle is not required.

2.4 A programming algorithm for AIMC

In this Section, leveraging the characterizations described in the previous Sections, an iterative programming algorithm is defined, able to set the cell conductance close to a desired value. The algorithm is outlined in Figure 2.16. Once the conductance target interval has been defined, specifying the mean value and relative tolerance, the cell is first stimulated with the start SET and RESET pulses, as suggested by the results of the analysis dis-cussed in the previous Section. Then the partial-SET SSC sequence begins with a minimum SET amplitude A_{MIN}. After a predefined time, interval T_{WAIT}, the cell current is read. If it falls within the target interval, the sequence is terminated. If the conductance is lower than the required limit, the cell is stimulated with a new SET pulse, with increased amplitude by a programmable step ΔA (see Figure 2.17, sample cells 1 and 3). If instead the conductance is above the upper limit, the whole process is restarted from the initial SET and RESET pulses (see Figure 17, sample cell 2). A maximum number of iterations $ITER_{MAX}$ is defined: if the algorithm exceeds $ITER_{MAX}$, the cell is declared not programmed and will not be used in the final AIMC array. Figure 17 shows the programming sequences relative to 5 sample cells, where the target has been defined as $0.5G_{MAX} \pm 10\%$ tolerance. It must be noted that the definition of this tolerance sets the maximum initial cell spread $\sigma(g)/g$ defined in 2.4. A_{MIN} has been set to $1.5A_{S0}$, ΔA to $A_{S0}/20$, T_{WAIT} to 1 ms and $ITER_{MAX}$ = 100. In the same way we programmed groups of NPC = 128 cells with target $g_0 = 1/6, 1/3, 1/2$ and $2/3$, respectively. Table 2.2 summarizes the minimum, maximum and average number of partial SET pulses required to program

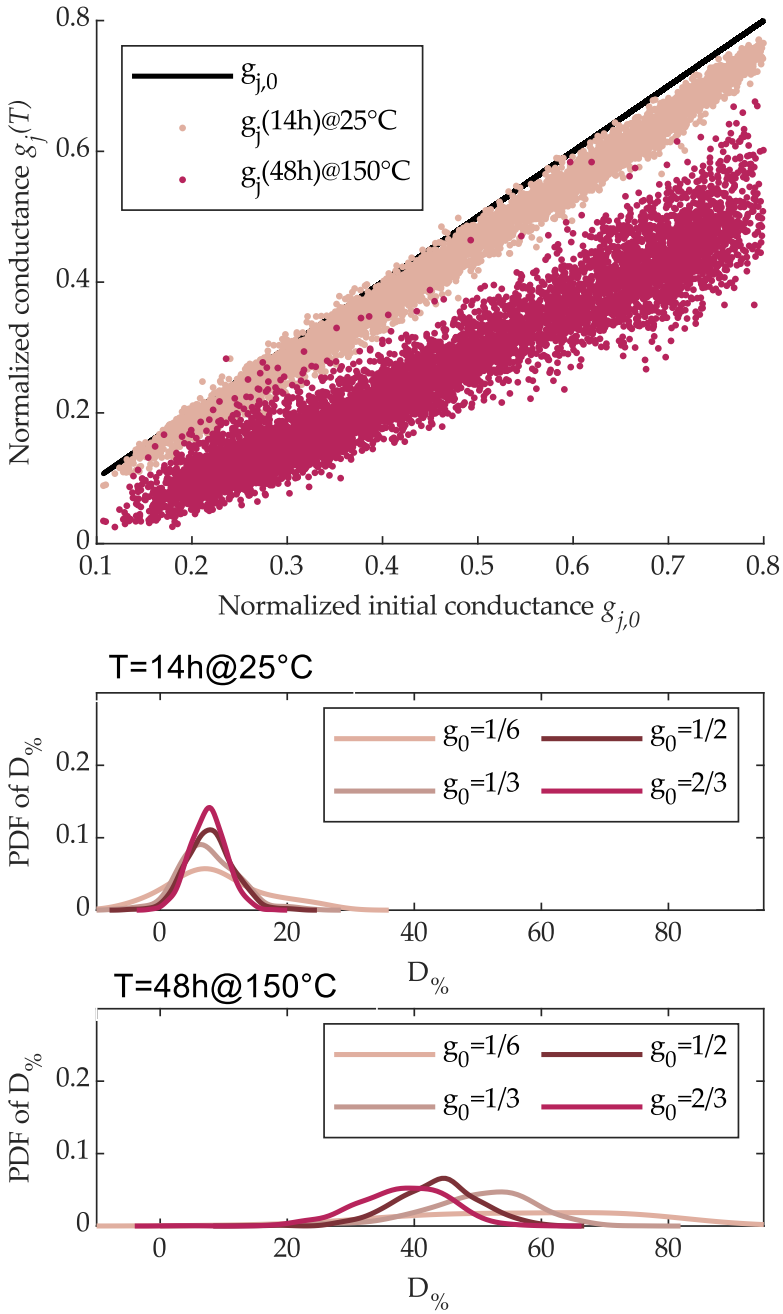

FIGURE 2.14: Effects of SSC sequence with $A_R = 3A_{R0}$. (a) Cells conductance as a function of the initial normalized conductance after 14 hours at room temperature, and after 48 hours at 150°C. (b) Probability distribution of D% obtained with the SSC programming sequence. Different curves refer to different target conductances with $\pm$ 10% tolerance.

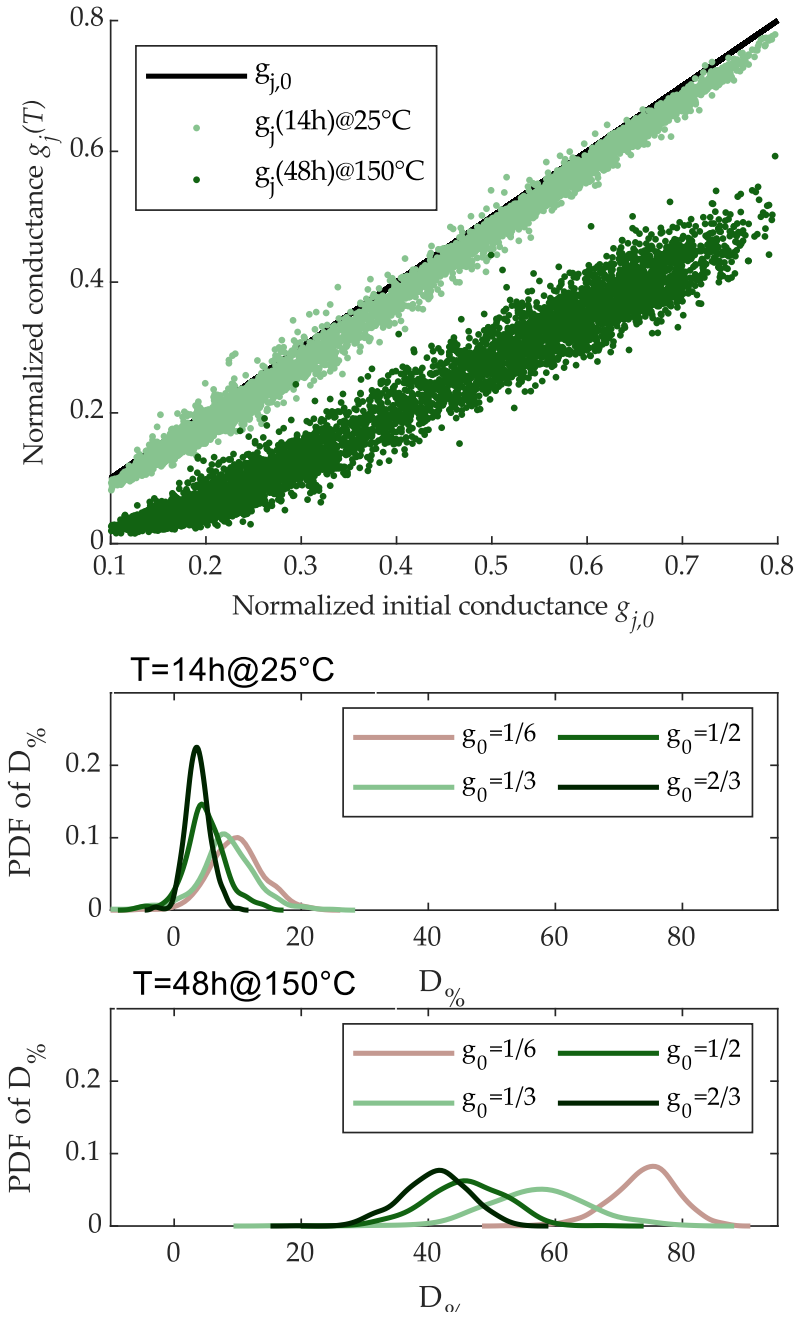

FIGURE 2.15: Effects of SSC sequence with the addition of an initial $5A_{S0}$ SET pulse and $AR = 5A_{R0}$. (a) Cells conductance as a function of the initial normalized conductance after 14 hours at room temperature, and after 48 hours at 150°C. (b) Probability distribution of D% obtained with the SSC programming sequence. Different curves refer to different target conductances with $\pm$ 10% tolerance.

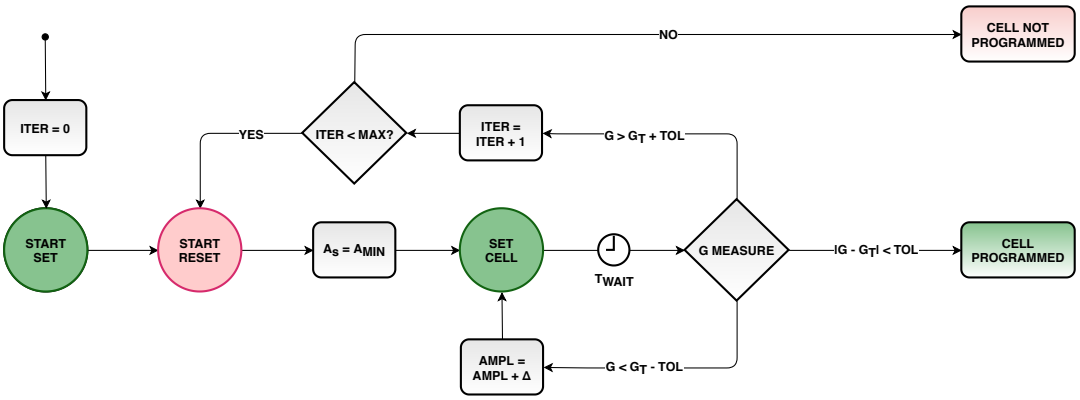

FIGURE 2.16: Proposed cells iterative programming algorithm. G indicates the measured cell conductance and G_T denotes the conductance target.

each cell, including possible restarted sequences. It can be noticed that the number of mean programming pulses increases with the conductance target, as we used the same A_{MIN} for every conductance goal. To improve the programming speed, A_{MIN} could be chosen in relation to the target level. Every cell was correctly programmed within the maximum 100 iterations.

Then, the programmed cells conductance has been monitored for $\sim$ 14 hours (160 samples with 5 minutes-steps), whose time evolution is depicted in 2.18. It must be noticed that 4 different levels of conductance are distinguishable in the whole observation time interval. For each programmed group of NPC = 128 cells we calculated the conductance spread defined as:

$$\frac{\sigma(g)}{g}(t_i) = \frac{100}{\langle g_j(t_i)\rangle}\sqrt{\frac{1}{NPC-1}\sum_{j=1}^{NPC}\left[g_j(t_i) - \langle g_j(t_i)\rangle\right]^2} \tag{2.5}$$

and results are reported in 2.19. The initial value is under 6% in all cases (5.08%, 5.17%, 3.16% and 2.42% for g_0 = 1/6, 1/3, 1/2 and 2/3, respectively) lower than the target tolerance $\pm$ 10%. Then, due to the random conductance drift, $\sigma(g)/g$ tends to increase in the first readout interval (5 minutes). After that time, spread does not change significantly, suggesting that the effect of drift is appreciable mostly in the first 5 minutes (or less). Moreover, cells with higher conductance show a lower and less variable spread, consistent with the previous analysis (see Figure 2.15).

Noise was evaluated through taking the last 120 samples occurring after 4 h from

TABLE 2.2: Required number of steps for cells programming.

Normalized target	Min n. of steps	Max n. of steps	Mean n. of steps
1/6	2	20	6
1/3	2	45	10
1/2	2	64	22
2/3	3	95	36

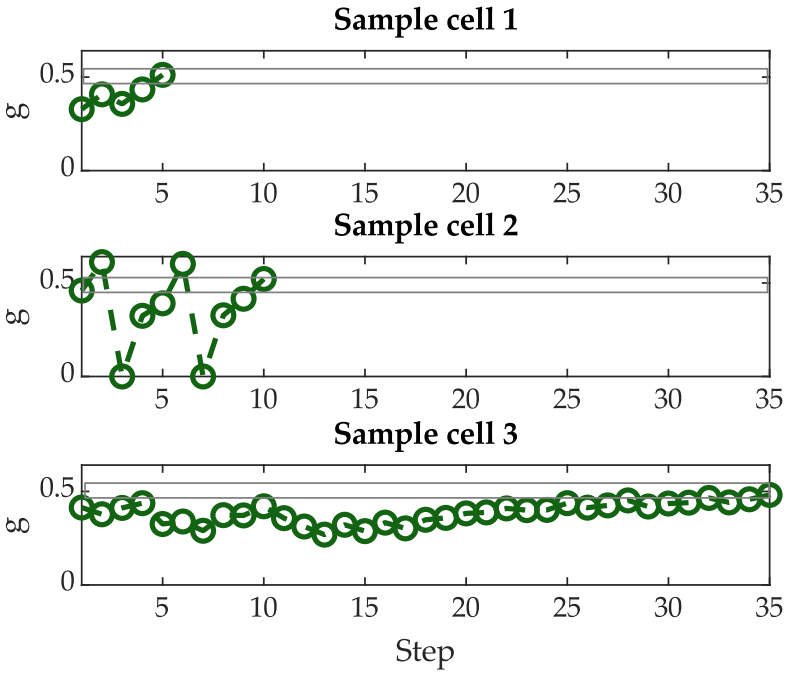

FIGURE 2.17: Typical evolution of the conductance of 3 sample cells during the programming sequence steps with the conductance target value set to $1/2 \pm 10\%$. (1) cell programmed in few steps and only one iteration; (2) cell programmed in 3 iterations; (3) cell programmed with a long sequence of steps. The horizontal lines show conductance target $\pm$ 10%.

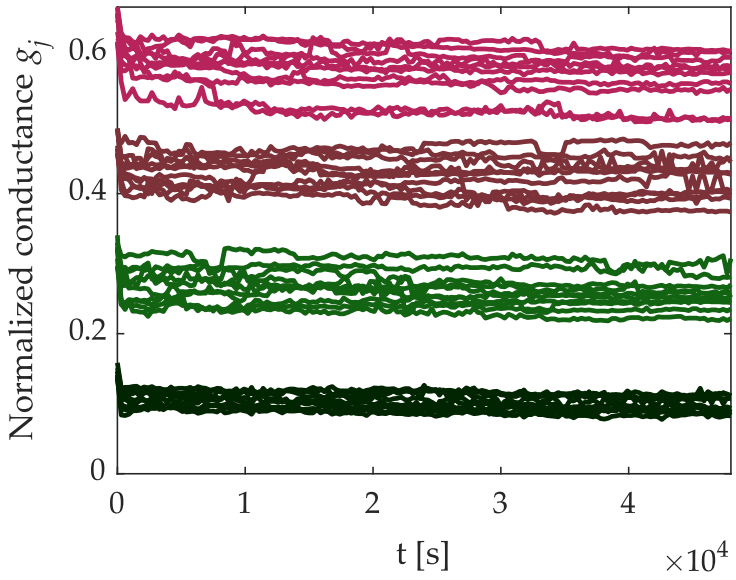

FIGURE 2.18: Programmed cells conductance behavior monitored for 14 hours. Only 10 cells each group are plotted. Initial conductance target values are 1/6, 1/3, 1/2, 2/3.

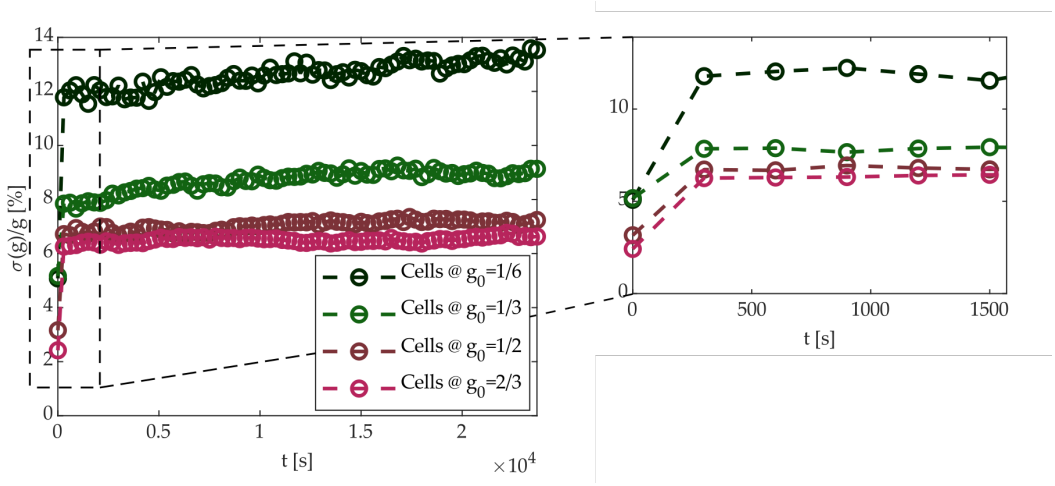

FIGURE 2.19: Cells conductance spread $\sigma(g)/g$ defined in 2.5 vs. time. A zoom on the first 6 measures is shown the effect of drift on the initial spread set by the proposed programming algorithm.

the application of the programming sequence to neglect initial strong drift effects. Results are shown in Figure 2.20 (left) with circles, where N% defined in 2.1 for each of the 512 cells is reported. Cells with the lowest conductance were characterized by N% in the 2–10% range (except for two cells); the lowest noise, less than 2%, was achieved by the cells with the highest conductance $g_0 = 2/3$.

Finally, D% defined in 2.3 is shown in Figure 2.20 (right) with circles. Results showed a decrease of conductance loss for higher-conductance, and D% was lower than 10% for all cells except for the ones with the lowest conductance levels. This is a key feature of SSC programming strategy combined with the adoption of start SET and start RESET pulses. Solid lines in both plots in Figure 2.20 report the ensemble average $\langle N_{\%,j} \rangle$ and $\langle D_{\%,j} \rangle$ over all the 512 tested cells with circles as a function of the conductance target, together with the indication of the 10% and 90% limits of the distributions.

2.5 Time-temperature combined effect analysis

In this final Section, we exploit the possibility to program PCM cells to a predefined target to study the effects of temperature on drift and noise. According to this purposes, n_c cells have been programmed to the normalized target conductances $\hat{g}_i$ with a normalized tolerance $\pm\delta g$. For each normalized target conductance $\hat{g}_i$, a set of n_c cell conductances is associated and then characterized in terms of spread, drift and noise, and different temperatures T have been included in this study. For each levels, the mean value of cells conductance $\mu_g(t_0)$ and their relative dispersion $\sigma_g(t_0)$ are defined. To this purpose, the considered cell conductances sets have been measured 24 hours after programming at room temperature $T = t_A$ (approximately 25°C), defining thus a new cell set, with its mean value $\mu_g(t_1)$ and relative dispersion $\sigma_g(t_1)$. Afterward, the test chip has been baked at $T = T_B = 90$°C. To monitor the

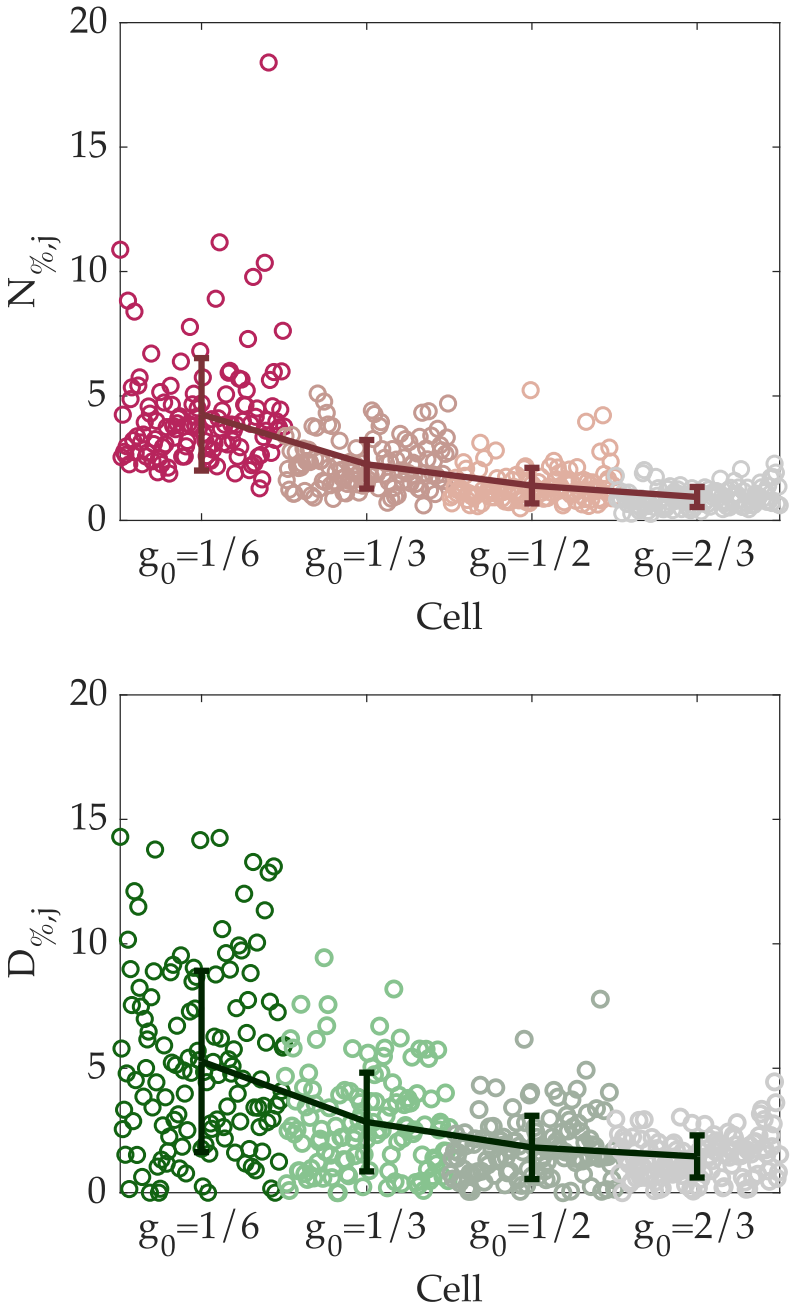

FIGURE 2.20: (Left) $N_{\%,j}$ defined in (3) of the 512 programmed cells. Circles represent noise of single cells. Error bars indicate noise mean value for the four conductance target levels, together with the 10% and 90% limits of the distribution. (Right) $D_{\%,j}$ defined in (4) of the 512 programmed cells. Circles represent drift of single cells. Error bars indicates noise mean value for the four conductance target levels, together with the 10% and 90% limits of the distribution.

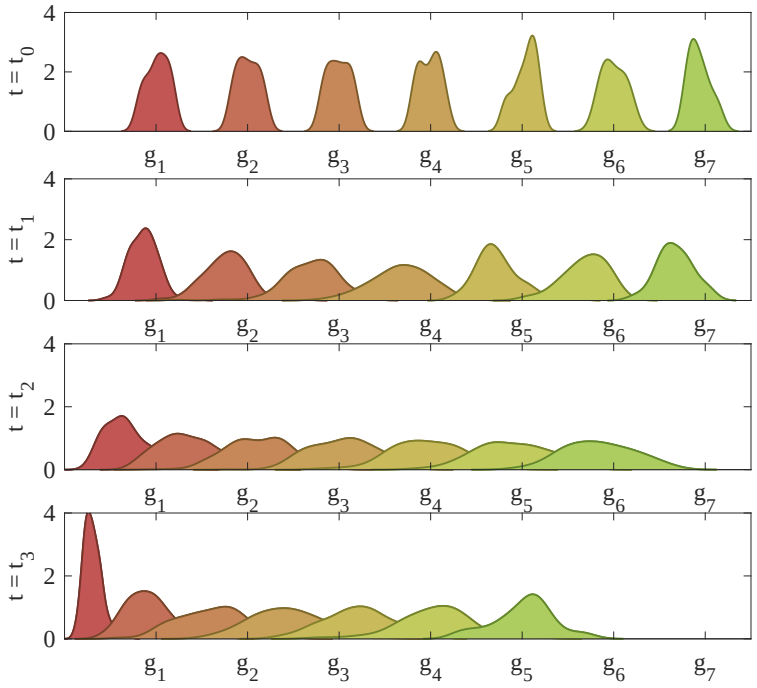

FIGURE 2.21: Probability density function of measured normalized cells conductances. The four plots are related to the distribution after programming (t_0), after 24 hours at $T_A = 25$°C (t_1), after 24 hours at $T_B = 90$°C (t_2) and after 24 hours at $T_C = 150$°C (t_3), respectively.

dynamics of each cell, the conductances were then measured at room temperature (to avoid leakage current increase due to high temperature) after 24 hours of bake; the same process has been repeated for $T = T_C = 150$°C, defining thus $\mu_g(t_2)$, $\sigma_g(t_2)$, $\mu_g(t_3)$ and $\sigma_g(t_3)$.

2.5.1 Evolution of cells distributions

In the first subplot of Figure 2.21, the probability density functions (pdfs) of the seven conductance levels are shown. In this case, the distributions of all cell sets are separated, their mean values are near the conductance targets $\hat{g}_i$, and their boundaries lays under the normalized target tolerance $\pm\delta g$. These conditions are implicitly granted by the adoption of the aforementioned single-cell iterative programming algorithm previously described. In the further subplots, reporting the pdfs at $T = T_A$, T_B and T_C, respectively, it can be easily observed that cells distributions tend to decrease their mean conductance μ_g, while their relative dispersion σ_g increases. As a result, the considered conductances distributions tend to overlap, as memory cells have lost their initial conductance under the combined effect of time and temperature due to random alterations to their internal structure. The values of $\mu_g(t_0)$ and $\mu_g(t_i)$ are shown in Figure 2.22 (left), while the right plot reports the values of $\sigma_g(t_0)$ and $\sigma_g(t_i)$. The mean values of cells sets tend to decrease uniformly with a slight dependence on the mean initial conductance, whereas, the cells sets dispersion increase is more evident for cells set with the lower value of target $\hat{g}_i$. Moreover, as

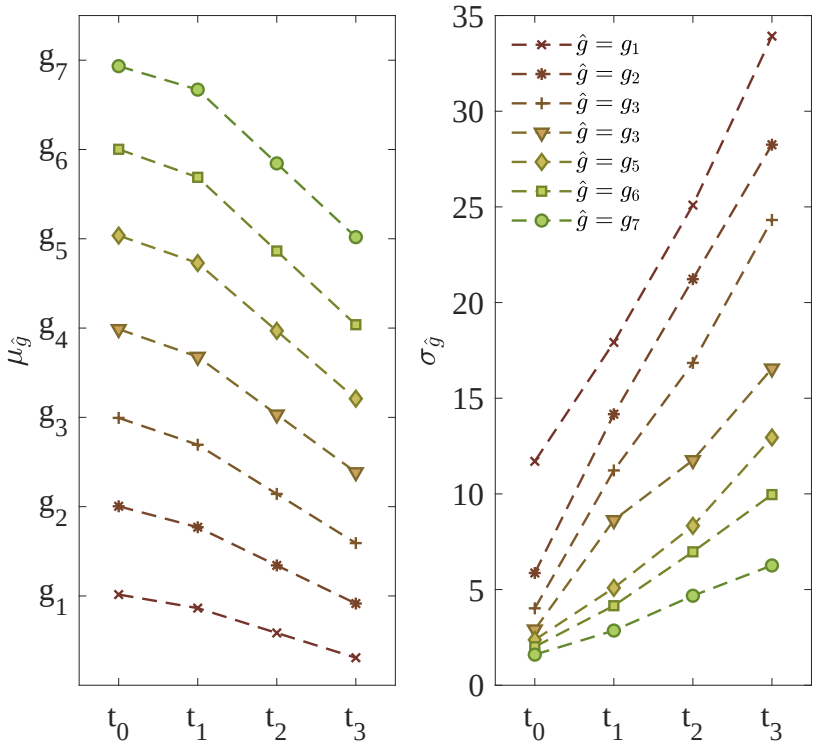

FIGURE 2.22: Left: measured mean normalized conductance μ_g after programming (t_0), and in the three conditions corresponding to t_1, t_2, t_3; different curves refer to the seven target conductances $\hat{g}_i$. Right: measured conductance relative dispersion σ_g in the same conditions.

the programming tolerance δg has been chosen equal for all $\hat{g}_i$, $\sigma_g(t_0)$ results to be inversely proportional to the target conductance.

In order to describe the behaviors of cells sets, the absolute variations of mean values $\Delta\mu_g(t_i) = \mu_g(t_0) - \mu_g(t_i)$ and dispersions $\Delta\sigma_g(t_i) = \sigma_g(t_0) - \sigma_g(t_i)$ of cells sets are plotted as dots in Figure 2.23 left and right, respectively. As previously shown, $\Delta\mu_g$ increases with time and bake temperature. Moreover, $\Delta\mu_g(t_1)$ is slightly dependent on the target conductance $\hat{g}_i$ and varies between 0.1 and 0.3; $\mu_g(t_2)$ instead is greater for the higher values of $\hat{g}_i$ and ranges from 0.3 to 1.1, while $\mu_g(t_3))$ varies from 0.5 to 1.9, and shows a strong dependence on $\hat{g}_i$. For what concerns the cells set dispersion, $\Delta\sigma_g$ increases when cells conductance target $\hat{g}_i$ is greater. In particular, $\Delta\sigma_g(t_1)$ varies from 5% to -0.5%, $\Delta\sigma_g(t_2)$ varies from -15% to -1%, and $\Delta\sigma_g(t_3)$ varies from -25% to -5%. All the measured values of $\Delta\mu_g$ and $\Delta\sigma_g$ can be fitted with 2^{nd}-order polynomial functions of conductance target $\hat{g}_i$, which are plotted in Figure 2.23 as dashed lines. The fitting functions of $\Delta\mu_g$ show a more incisive dependence on the 2^{nd}-order term $\hat{g}_i{}^2$, whereas $\Delta\sigma_g$ has a stronger dependence on the 1^{st}-order term $\hat{g}_i$.

2.5.2 Effects on drift coefficient

In this context, three additional measurements have been performed after $\Delta t = 12$ hours from t_1, t_2 and t_3, respectively. Comparing these measurements with their

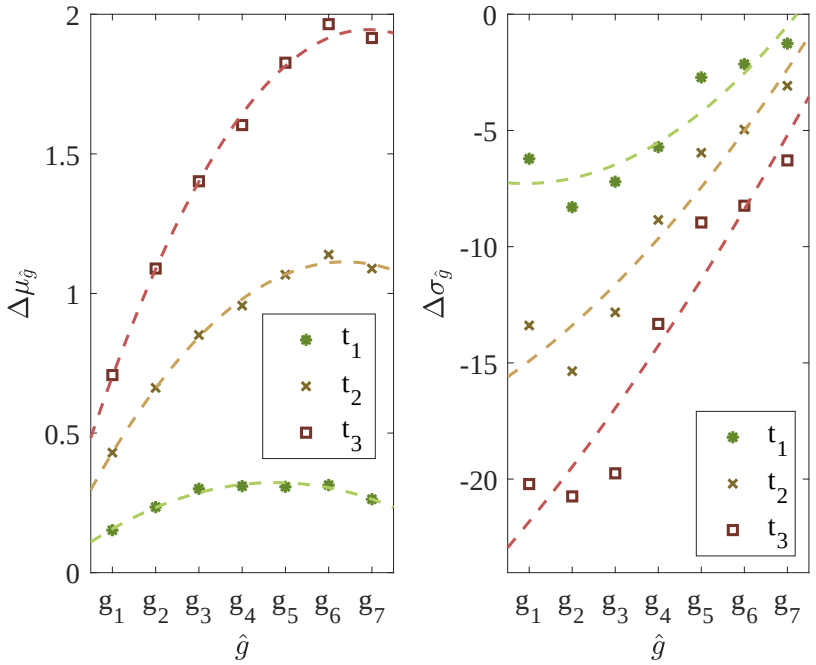

FIGURE 2.23: Left: measured absolute variations of mean values (dots) as a function of targets $\hat{g}_i$, and their fitting functions (dashed lines), in the three conditions corresponding to t_1, t_2, t_3. Right: measured absolute variations of dispersions (dots) as a function of targets g $\hat{g}_i$, and their fitting functions (dashed lines) in the same three conditions.

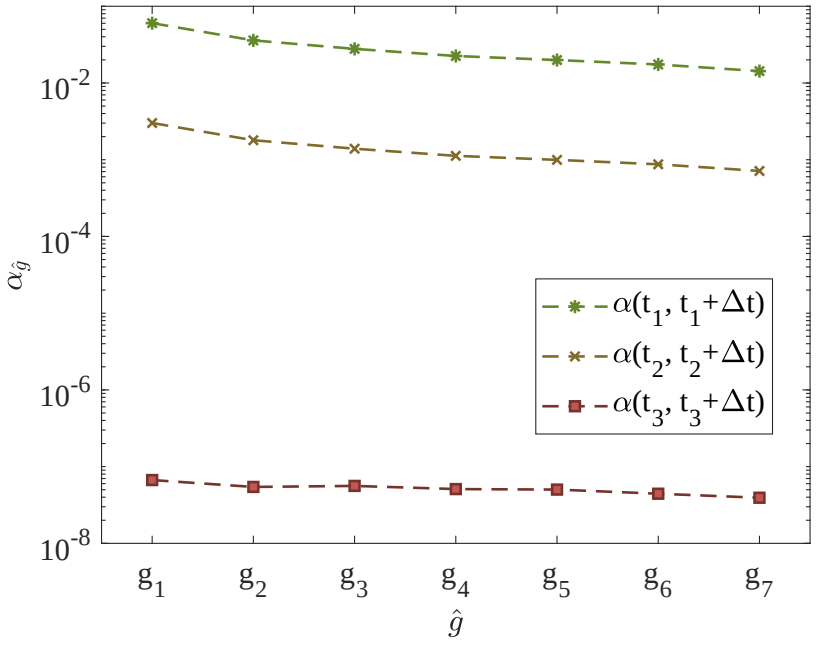

FIGURE 2.24: Mean drift coefficients of cells sets as a function of the targets $\hat{g}_i$ in the three conditions corresponding to t_1, t_2, t_3 with $\Delta t =$ 12 hours.

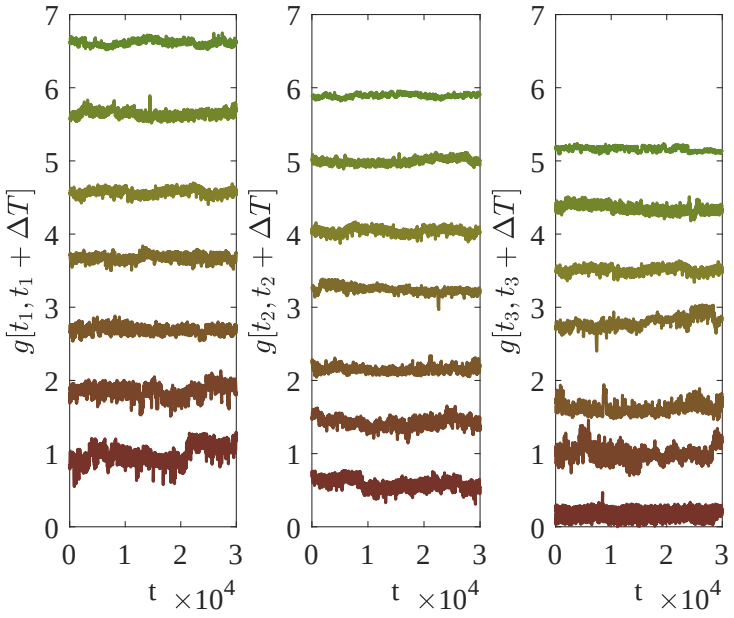

FIGURE 2.25: Example of noise measurement of seven sample cells at $[t_1, t_1 + \Delta t]$ (left), $[t_2, t_2 + \Delta t]$ (center), $[t_3, t_3 + \Delta t]$ (right), with Δt = 5 min.

corresponding of Figure 2.22 left, and inverting 2.2, drift coefficient has been then estimated. The influence of targets $\hat{g}_i$ and temperature on α are depicted in Figre 2.24, where the mean drift coefficients $\alpha_{\hat{g}_i}$, which are the mean drift coefficient of each cell set, are plotted in logarithmic scale as a function of the targets $\hat{g}_i$, and for the three different test conditions. Results show that the drift coefficient slightly depends on the target conductance value, and it is more significant for low $\hat{g}_i$. As α is greater than 0.01 when T =25°C, cells tend concordantly to weakly drift after $\Delta t = 12$ hours. When $T = 90$°C or 150°C, α is near to 0 in the time interval of Δt, concluding that, in these two latter conditions, time drift can be considered negligible, concluding that its effect becomes trifling in few hours when high temperature is applied.

2.5.3 Effects on noise

An analysis of cells noise concludes this Section. To characterize this aspect, cells conductances have been measured over a time interval $\Delta t = 5$ min, collecting n_s = 30000 samples for each cell. Then, the mean noise $N_{\%}$, defined accordingly with 2.1, is evaluated for each cell set. An example of noise measurement is reported in Figure 2.25, where seven sample-cells conductances are showed. Measures have been performed at $t_1 + \Delta t$, $t_2 + \Delta t$, $t_3 + \Delta t$. The mean $N_{\%}$ as a function of target conductances $\hat{g}_i$ is reported in Figure 2.26. Results show that noise is more relevant for lower values of $\hat{g}_i$, where it is about three times greater. Moreover, $N_{\%}$ does not significantly differ for different temperatures. Accordingly, noise in PCM elements

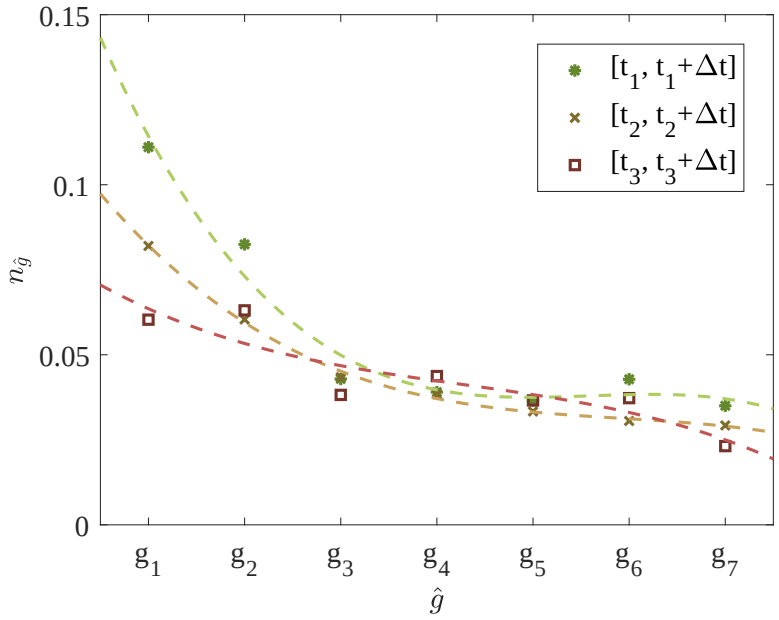

FIGURE 2.26: Mean noise of cells sets as a function of target conductances in the three conditions corresponding to t_1, t_2, t_3.

is related to amorphous phase of cells, which is more significant in low-conductance ones.

2.6 Conclusion

PCM cells non-idealities, i.e. low-frequency noise, time drift and conductance spread, lead to inaccuracies which affect the computation process accomplished by the memory array. Proper cell programming sequences to mitigate these undesired effects are proposed. In particular, higher applied SET-amplitude pulses lead to better performance in terms of noise. In addition, results have shown that, for a given target conductance, a single cell achieves more noise reduction than several cells in parallel each having lower conductance. Besides, drift is reduced when high SET-amplitude pulses are employed. The SSC-programming strategy ensures better results in terms of cells spread and initial conductance control. Moreover, the application of large start SET and RESET pulses at the beginning of the programming sequence achieves a better cells dispersion performance. Drift, dispersion and noise have been then analyzed in relation to memory elements programmed with a dedicated programming algorithm, showing their dependences on conductance targets and temperature. As an example of application of the above considerations, the results of programming 40 cells with 4 different conductance levels are shown. The cells conductances have been monitored up to 14 hours after the application of the programming procedure. For all memory cells the measured conductance spread is under 14% and the relative drift under 15%, the relative noise less than 9% for the 90% of cells.

Chapter 3

Evaluation of PCM-based AIMC operations for specific applications

On the basis of the results previously presented, the use of PCM cells in two different applications is simulated in this Chapter. Here the aim it to quantify the impact of PCM devices non-idealities when employed to perform Multiply and Accumulate (MAC) operations, which are the kernel of Matrix-vector Multiplications (MVMs). To this purpose, additional characterization procedures have been implemented. This Chapter also motivates the design of the AIMC unit presented in Chapter 4, whose aim is is to develop an embedded unit which adds analog in-memory computing (AIMC) features to an embedded PCM (ePCM) memory. These analyses have been carried out with the contributions of expert collaborators, which provided and implemented the application scenarios.

Some of the material reported in this Chapter is reused from [20], and from [3], in agreement with IEEE and ASCE copyright on theses and dissertations.

3.1 A basic approach for AIMC based on PCM cells

A circuit representing a basic idea for a PCM-based AIMC scheme is illustrated in Figure 3.1, and exploits the conductances stored in an embedded PCM array to perform MAC operations. Typically, embedded PCM memory arrays are organized in bit lines (BLs) and word lines (WLs), which are accessible through selectors made of NMOS transistors, as shown in dedicated works [27]–[29], [54]. With the selection of a single word line, memory cells can be accessed in parallel through the main bit line (MBL) nodes. Once a WL is selected, and a voltage V_i is applied to a single cell programmed to a conductance g_i, the current obtained is $I_i = g_i V_i$; the total current of the summation node is:

$$I_{OUT,j} = \sum_{i=1}^{n} g_{j,i} V_i \tag{3.1}$$

which corresponds to the product between a voltage vector $\mathbf{V} = [V_1, ..., V_n]$ and the conductance vector $\mathbf{G}_j = [g_{j,1}, ..., g_{j,n}]$ of the selected j-th word line. The main limit of this architecture is that it allows to compute a product between an input vector and a conductance vector of a single selected word line only per cycle. To

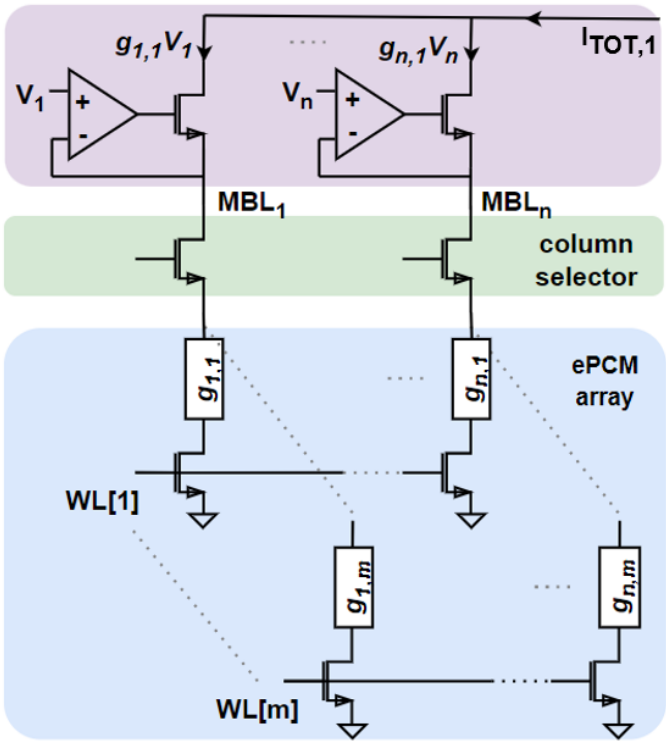

FIGURE 3.1: Basic architecture to execute MAC operations with no modifications to the structure of the PCM IP. In this example, the wordline WL_1 is supposed to be activated.

obtain a full matrix vector multiplication it is necessary to repeat the operation by selecting in sequence the following word lines. Furthermore, this architecture does not address the non-idealities of PCM (as previously analyzed in Chapter 2). In fact, noise and drift effects would directly influence the accuracy of MAC operations; moreover, the non-linear I-V characteristic of the PCM devices will become relevant in the computation, as each variable input is here applied on a PCM cell. Thus, each MAC coefficient g_i shows a dependence on both the input and on time:

$$g_i = g_i(V_i, t) \tag{3.2}$$

which is directly mapped on the MAC result.

In this Chapter, we analyze the impact of such non-idealities on specific applications. In particular, the I-V characteristic effect is modeled in a Deep Neural Network (DNN) application, whereas drift and noise are taken into accout in a filtering process for a Structural Health Monitoring (SHM) algorithm.

3.2 Neural networks

In this Section, we simulate the employment of PCM cells in AIMC operations performed with the previous architecture in the field of Deep Neural Networks (DNNs). To this purpose, we acquired and analyzed a set of I-V characteristics of PCM cells,

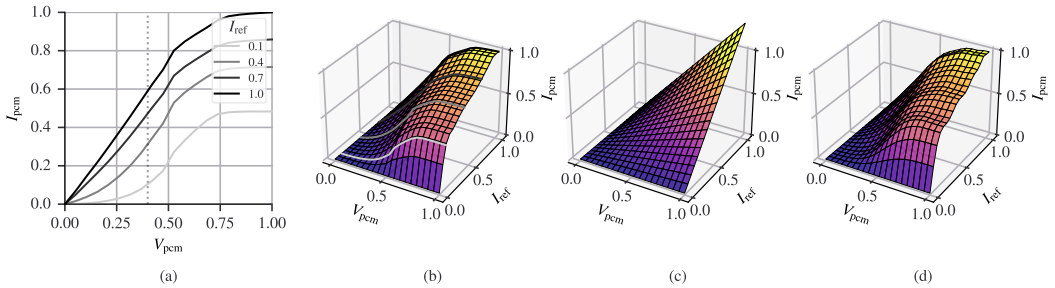

FIGURE 3.2: (a) Average, normalized I/V characteristics of PCM devices in four different conductance states. (b) Spline-based interpolation of the average, normalized PCM behaviours, highlighting the four states depicted in (a). (c) Low-order polynomial fitting of spline-generated data points. (d) High-order polynomial fit of the same data.

exploiting also the experimental setup previously exposed in Chapter 2, and we analyze the impact on non-linearities in the accuracy of two classification tasks performed with DNNs.

3.2.1 PCM Characterization and Numerical Modeling

We have performed measurements on the ePCM test chip designed and manufactured by STMicroelectronics in a 90-nm BCD technology previously employed in Chapter 2.

Device characterization begins with the dedicated programming step, where the PCM cells are brought into highly conductive SET states by means of a single current pulse. A higher pulse intensity determines a more conductive state. The RESET state, conversely is associated in this work to a null SET intensity. The current through each of the 5120 available cells has been measured while sweeping the voltage across each cell, for different values of the applied programming pulses.

The one-shot programming phase does not include any iterative feedback mechanism to ensure that the programmed cell state is indeed the expected one. As our goal requires the definition of nominal cell behaviours in different conductive states, then the intensity of the applied current pulse does not provide a good measure of the actual state. We have therefore classified the cell behaviour according to the features of the obtained $I(V)$ curves themselves, disregarding the intensity of the programming pulse. Indeed, similar $I(V)$ curves could be obtained by a cell programmed with a low-intensity pulse which in reality acts stronger than intended, or a high-intensity pulse whose result is particularly weak.

Therefore, typical behaviours of ideal PCM cells are obtained by observing all the curves at a fixed voltage V_{ref}, quantizing the current axis I around a set of reference currents $I_{ref} = \{I_{ref}^{(0)}, \ldots, I_{ref}^{(L-1)}\}$ and averaging all the cells belonging to the same quantization bin to obtain the typical behavior associated to that bin. A selection of curves obtained at different $I_{ref}^{(l)}$ values is shown in Fig. 3.2. More in detail,

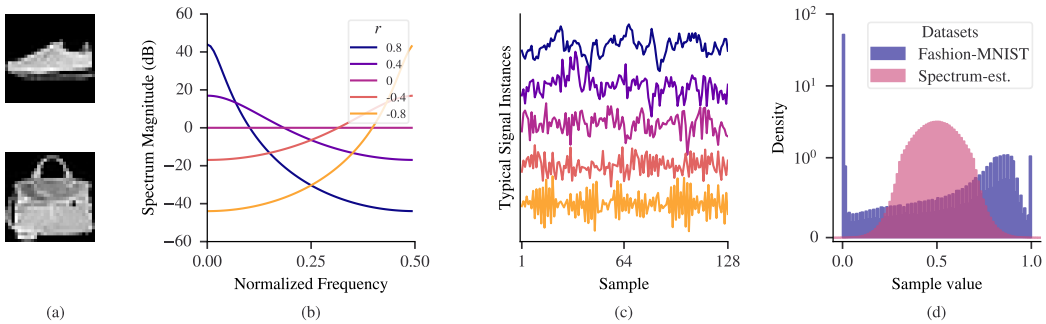

FIGURE 3.3: (a) Examples of Fashion-MNIST instances (b) Spectrums for a selection of r with corresponding representative instances of length 128 in 3.3. (c) Normalized histograms of input values for the two datasets being employed. A linear scale is used in the range $[0, 1]$ for the vertical axis.

we define the nominal behaviour of a PCM cell as the average of all collected I/V characteristics whose current is contained in the interval $(I_{ref}^{(l)} - \Delta I^{(l)}, I_{ref}^{(l)} + \Delta I^{(l)})$ at voltage V_{ref}, with $\Delta I^{(l)} / I_{ref}^{(l)} = 5\%$. Having V_{ref} fixed, we then identify the programming state with the value of measured current. Values have been normalized so that applied voltages V, programming states $I_{ref}^{(l)}$ and output currents I all lie in the $[0, 1]$ range, as shown in Fig. 3.2. In the following, we will only use this normalized data.

To obtain a numerical model of the type $I(V, I_{ref}^{(l)})$ we have interpolated the typical behaviours, extracted according to the above procedure, using a spline of order 3. The result is depicted in Fig. 3.2. This allows a reduced local complexity of the model, while still describing accurately the features of the underlying surface. The spline model will be here considered as our reference PCM model. At the same time, polynomial models of arbitrary order (in the range 3 to 27) have been fitted to the spline. Fig. 3.2 and Fig. 3.2 highlight the difference in approximation accuracy obtained with different polynomial degrees. The necessity for polynomial models will be clarified in the following section. Suffice it to say that the use of such models within neural networks programming libraries allows the automatic differentiator procedures to operate without concerns.

For convenience, the data, which is defined over positive values for both the applied voltage and the programming state, has been extended towards negative values along the programming axis, so that $I(V, -I_{ref}^{(l)}) = -I(V, I_{ref}^{(l)})$ with $V, I_{ref}^{(l)} > 0$. Even if negative values for the programming state are not physically meaningful, the actual hardware implementation can operate so that the contribution of a particular cell is negative on the output. A convenient side effect, is that the artificially-introduced symmetry makes the polynomial representation more well-behaved and reduces the number of significant coefficients, hence the computational burden.

3.2.2 Neural Training with PCM Layers

In a traditional dense layer, the core operation for the j-th neuron is $h_j = f(b_j + \sum_i w_{j,i} x_i)$. Aiming towards a circuital implementation where inputs are voltages, and they are weighted by conductances programmed in different states, the expression becomes $h_j = f(b_j + \sum_i I(x_i, w_{j,i}))$, where we neglect any additional term introduced by electrical noise, programming noise or even quantization of the inputs or the outputs.

Training a layer requires that $I(x_i, w_{j,i})$ is differentiable with respect to the weights [60]. Therefore, the synapses description is in this case of polynomial type. Potentially, a more physically-based model could be used as well, though as our measurement data includes significant effects from the access devices surrounding the PCM cells, we have preferred to have a unique model that could describe the behaviour of the entire circuital block over the full voltage domain.

Two case studies will be analyzed in the following: a classification task performed on the Fashion-MNIST dataset [61], and a regression problem, in which the network has to estimate a parameter describing the spectral content of randomly generated signal instances.

3.2.3 Results

In the following we will show numerical results on the training of neural networks in which one layer is PCM-based. In all setups, the performance of a neural network employing only conventional dense layers and having the same structure, is used as a reference. To train the PCM-based network, the PCM synapses are always described by their polynomial model, with an arbitrarily selected degree and by identifying $L = 10$ different reference currents. An initial performance metric is thus obtained, related exclusively to the use of the polynomial. The final evaluation is then performed on the same network, preserving the trained weights, but replacing in the PCM-based layer the polynomial model with the spline one, representing our reference model for nominal PCM devices. Since in a physical implementation the state of a PCM cell cannot be programmed to arbitrary accuracy, we also test the robustness of the network towards this kind of perturbation. We model the variation of the PCM state with a white gaussian noise added to the nominal values of the weights (i.e., those suggested by training) during the final evaluation. The variance of the weight noise is normalized to the nominal value, so that their ratio is fixed. Clipping is then applied to ensure that the noisy weights are still within the validity range of the numerical models.

Two different applications are shown, trying to highlight the different features of the setups presented in this work and results are condensed in Fig. 3.4.

3.2.3.1 Fashion-MNIST Classification

The dataset is made of grayscale images of clothing articles, in a 28×28 pixel format. Two examples are shown in Fig. 3.3. The neural network topology being considered has an input-flattening layer followed by a single dense layer with sigmoid activation functions and 10 output nodes. The loss function is the sparse categorical cross-entropy. While the conventional reference network has no constraints on the weights, in the PCM-based one we have introduced a "bathtub" regularization to force them within the [0, 1] range. This implies a physical realization requiring only positively-contributing PCM synapses on each layer output.

To assess the performance we use here the accuracy defined as the correct classification rate. Analyzing the results shown in Fig. 3.4, a monotonic trend is clear, with networks trained on a high-order polynomial model almost matching the performance of the reference network.

The fact that the weights obtained by training a low-order polynomial, as that depicted in Fig. 3.2 is already sufficient to solve the classification task with ~ 0.78 accuracy has been associated to the statistical distribution of pixel intensities. Being their density concentrated around the extremes of the available range, as shown in Fig. 3.3, the inherent nonlinearity of the models is not significantly excited. The model feature that matters is that their output is different for low and high input values. Both the spline and polynomials being employed possess such a feature, resulting in a limited performance drop with respect to the reference case.

The application of noise on the trained weights only becomes significant around 10% relative standard deviation, with a performance loss still within 3.5% of the noiseless setup. State-of-the-art iterative programming techniques of the physical devices may indeed be able to achieve such a level of programming accuracy [62], [63].

3.2.3.2 Spectral Estimation Regression

The second task being evaluated is a regression problem artificially constructed so that the nonlinearity of the PCM I-V characteristic can be excited even more.

The problem is that of estimating the properties of the Fourier-spectrum of random signal instances. Signals are characterized by a given a value $-1 < r < 1$, such as a signal profile is high-pass for $-1 < r < 0$, flat/white for $r = 0$ and low-pass for $0 < r < 1$. Examples of spectra for different values of r are shown in Fig. 3.3, with corresponding representative signal instances depicted in Fig. 3.3. Further details are provided in [20].

Given a value for r, signals can be generated by computing instances of a multivariate gaussian distribution $\mathcal{N}(0, K)$. Inverting the relationship between the power spectrum and r is not possible, and the neural network has to estimate it by looking at each signal instance and providing an answer in the $[-1, 1]$ range.

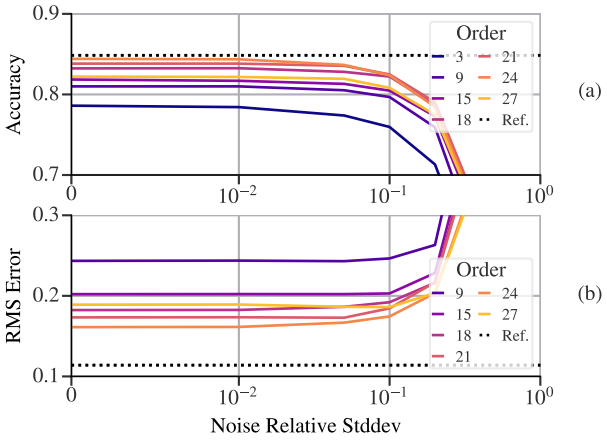

FIGURE 3.4: Results for (a) Fashion-MNIST classification, and (b) spectrum estimation regression. The black, dotted line represents the performance obtained by a neural network employing standard dense layers, without noise. Solid lines refer to the performance of networks using the spline PCM model, with the weights trained on the polynomial description of the device, and additional noise included during the evaluation phase.

The network structure being tested operates on signal instances of 32 samples and it has three dense layers of size 256, 256 and 1. The first two layers have relu activation functions, while the output layer has none. The loss function is the mean squared error, while the performance metric being observed is the root mean squared (RMS) error. A conventional network with such a structure achieves a 0.114 RMS estimation error.

The weighting coefficients of the PCM-based layer in this case have been constrained in the range [-1, 1] . From an implementation point of view, this requires a way for a PCM cell, to have a negative contribution on the sum of synapses currents, which is widely demonstrated in literature [64], [65].

Results in Fig. 3.4 highlight a monotonic trend up to order 24, with a sudden worsening of performance observed at 27.

The detrimental effect of the additive noise on the weights is still under control for 10% relative standard deviation, with variations on the order of 0.014 RMS error with respect to the noiseless setup. It is striking to observe a minimal performance increase when weight noise is applied to the network trained on the order-27 polynomial.

3.3 Structural health monitoring

In this Section, we analyze the employment of PCM cells in the context of Structural health monitoring (SHM), exploiting PCM cells as filter banks coefficients for signal processing. In particular, we focused the characterization on the effects of drift and noise on the filtering accuracy, using a specific monitoring scenario. Moreover, a

dedicated filtering scheme has been developed to reduce the effects of noise in the filters implementation.

Structural health monitoring (SHM) involves the observation and analysis of a system over time using periodically sampled response measurements to monitor changes to the material and geometric properties of engineering structures such as bridges and buildings. SHM systems can be particularly helpful in assessing structural integrity to improve maintenance administration or emergency management [66], [67]. A considerable amount of research has been conducted lately to make vibration-based SHM techniques more and more advanced, dealing with the identification of structures with closely spaced vibration modes [68]. Recently, low-cost sensing components, together with wireless transmission modules, have been studied to cut the costs related to the initial investment for an SHM system [69], [70]. However, frequent battery replacement is not viable when the monitored structures are numerous and distributed over wide areas. For this reason, efficient algorithms and smart data management strategies are gaining interest in both research and field applications [71], [72]. AIMC could be useful in this scenario, as edge computing is gaining interest to implement some tasks in SHM applications.

3.3.1 Identification algorithm

Consider the impulse responses $b_m[\tau]$, with $\tau = 1, \ldots, N$, of one low-pass ($m = 0$) and p bandpass ($m = 1, \ldots, p$) filters such that the central frequencies of the bandpass filters coincide with the first p resonant frequencies of a vibrating structure and their frequency bandwidth is small compared to the distance between consecutive modal frequencies [73], [74]. Let the coefficients of these filters be organized in column vectors $\mathbf{b}_m \in \mathbb{R}^N$. A filter bank matrix can be defined as follows:

$$\mathbf{B} = \left[\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_p\right] \tag{3.3}$$

Here, the term $\mathbf{b}_0$ encloses the coefficients of the low-pass filter that can be employed to extract quasi-static structural features. On the other hand, the terms $\mathbf{b}_m$ indicate the bandpass filters used to extract different modal contributions from the acceleration time response. Specifically, considering a matrix $\mathbf{X}_t$ such that

$$\mathbf{X} = [\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, \ldots, \mathbf{x}_{t,r}] \tag{3.4}$$

where $\mathbf{x}_{t,i}$ are column vectors collecting the samples of the acceleration signal $x_i[t]$ recorded at the instrumented locations $i = 1, \ldots, r$ in the time interval $[t, t + N]$, a

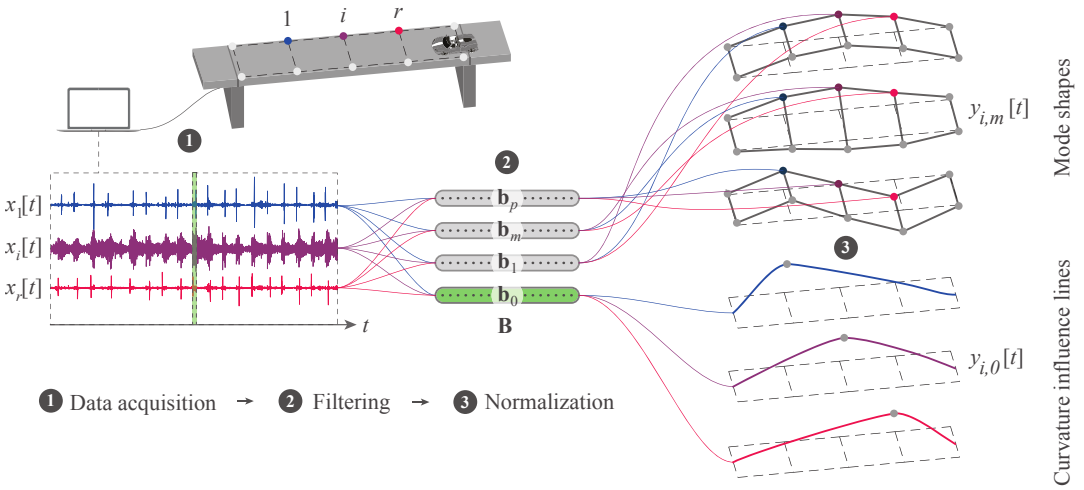

FIGURE 3.5: Scheme of the unified algorithm.

set of decomposed signals can be calculated as

$$\mathbf{Y}_t = \mathbf{X}_t^{\mathrm{T}}\mathbf{B} = \begin{bmatrix} y_{1,0}[t] & y_{1,1}[t] & \cdots & y_{1,p}[t] \\ y_{2,0}[t] & y_{2,1}[t] & \cdots & y_{2,p}[t] \\ \vdots & \vdots & \ddots & \vdots \\ y_{r,0}[t] & y_{r,1}[t] & \cdots & y_{r,p}[t] \end{bmatrix} \tag{3.5}$$

The elements $y_{i,0}[t]$, upon changing the time variable into space (i.e., $z = vt$), represent the samples of the curvature influence line of the beam at the i-th location. Due to the Maxwell-Betti reciprocal work theorem, $y_{i,0}[z]$ is also the structural curvature of the beam generated by a static load applied at the i-th instrumented location. Moreover, the terms $y_{i,m}[t]$ with $m = 1, \ldots, p$ are the t-th samples of the m-th decoupled modal contributions collected at the i-th location. Therefore, the m-th column vector of $\mathbf{Y}_t$, except when $m = 0$, is an instantaneous (the m-th) mode shape of the instrumented structure.

Based on these concepts, the identification algorithm is deeply exposed in [74] and it is based on the computation of 3.5. The procedure is as well schematized in 3.5. It should be noted that the acquisition interval can be triggered to select only the structural response referred to the vehicle passage automatically, e.g., using the signal collected at the bridge expansion joints. The identified parameters can be stored in each sensing node and averaged to the new incomes to improve the robustness to recording noise. Then, the averaged parameters can be transferred to a central unit or directly uploaded in a cloud-based platform at user-defined intervals. Since phase information is neglected (i.e., the sign of the elements of the identified shapes), strict synchronization is not necessary between the sensing nodes.

The filters $b_m[\tau]$ should be highly selective in frequency to avoid the mixing of different contributions that would affect the accuracy of the identified structural parameters. The procedure to generate suitable filters for the monitored structure is described in [74].

The signals decomposition can be implemented using low-pass and high-pass filters applied recursively n times to the input signal, where n is the selected maximum level of the wavelet transform. This implementation is known as Mallat algorithm or FWT [75]. Specifically, the output coefficients of the wavelet packet transform $d_{i,2k}^{(l)}[t]$ and $d_{i,2k+1}^{(l)}[t]$ obtained by decomposing the coefficients $d_k^{(l-1)}$ at the previous level $l-1$ can be calculated as

$$d_{i,2k}^{(l)}[t] = d_k^{(l-1)}[t] * \bar{g}_0[2\tau] \tag{3.6}$$

$$d_{i,2k+1}^{(l)}[t] = d_k^{(l-1)}[t] * \bar{g}_1[2\tau] \tag{3.7}$$

where $*$ denotes the convolution operator, $k = 0, \ldots, 2^{l-1}$ indicates the subband index of the obtained coefficients, and $g_0[\tau] = \bar{g}_0[-\tau]$ and $g_1[\tau] = \bar{g}_1[-\tau]$ are the impulse responses of the low-pass and high-pass filters associated with a selected wavelet function, respectively. The root of the tree $d_0^{(0)}[t]$ can be assumed coincident with the discrete signal $x_i[t]$ collected at location i if the sampling frequency of the collected signal is sufficiently high. Due to the linearity property of the convolution operator, the decomposition of the signal shown in Equations 3.6 and 3.7 can also be implemented as a one-step (or batch) filtering procedure using 2^n equivalent filters that produce the coefficients at the final transformation level n. These filters can be obtained by cascading (i.e., performing recursive convolution upon upsampling the filter at each iteration) $g_0[\tau]$ and $g_1[\tau]$ n times in a particular order [76]. For simplicity, let $G_0(z)$ and $G_1(z)$ be $g_0[\tau]$ and $g_1[\tau]$ in the z-transform domain, respectively. Due to the convolution theorem, the frequency representation of an equivalent bandpass filter $b_m[\tau]$ corresponding to the subband $k = m$ at the transform level n can be obtained as:

$$B_m(z) = \prod_{l=0}^{n-1} G_{l*}\left(z^{2^l}\right) \tag{3.8}$$

where $G_{l*}(z)$ can be either $G_0(z)$ or $G_1(z)$ depending on the level l and on the desired equivalent filter. For instance, $G_{l*}(z) = G_0(z) \forall l$ to generate the low-pass filter $b_0[\tau]$. In Equation 3.8, z^k represents an upsampling in the time domain by a factor k, i.e., the upsampled filter $g_{l*}[\tau]$ at level l can be obtained as:

$$g_{l*}[\tau] = \begin{cases} g_*\left[\frac{\tau}{2^l}\right] & \text{if } \tau = \xi 2^l, \xi \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

where $g_*[\tau]$ is either $g_0[\tau]$ or $g_1[\tau]$ depending on the level l and on the desired equivalent filter, and ξ is an integer value. Consequently, the number of null coefficients of $g_{l*}[\tau]$ increases with l, while the number of non-zero coefficients is constant.

Each filter obtained through this procedure at level n has a bandpass range width of $F_s/2^{n+1}$, where F_s is the sampling frequency of the collected signal.

The equivalent decomposition filters were obtained by cascading Fejér-Korovkin 22 wavelet filters, and have a relatively high number of taps (i.e., 22), which generate

equivalent filters that may be particularly challenging for implementations in smart sensing nodes. For instance, considering the wavelet transform level 6, each equivalent filter has 1326 taps. The low-pass and high-pass analysis filters have 4 taps, are symmetrical (anti-symmetrical for the high-pass filter), and are formed of only two coefficients, the higher of which is three times the lower, as shown in 3.6. Although most equivalent filters obtained through this wavelet function are scarcely selective, the low-pass filter, as well as some bandpass filters, are acceptable for identification purposes, as it will be shown later. In particular, ordering the equivalent filters obtained by cascading the wavelet filters in all the possible orders with an increasing central frequency, the $(2^{n-l}+1)$-th filters are sufficiently selective, especially for low l values (with $l = 1, \ldots, n$). These filters have a center frequency equal to:

$$F_l = \frac{F_s}{2^{l-1}} \tag{3.10}$$

Sampling the structural response (i.e., selecting F_s) such that the structural resonant modes have a natural frequency close to the F_l values allows the extraction of the corresponding modal contributions.

In this study, 48 memory cells of the aforementioned testchip were programmed in a laboratory environment to store 24 low and 24 high rbio3.1 decomposition filter coefficients. The following parameters were used in the described programming algorithm: A_{MIN} = 150 µA, T_{WAIT} = 1 ms, and ΔA = 10 µA. The coefficients of each filter were converted in conductance values $b_\zeta[\eta]$, which were then stored into specific memory cells. In particular, low filter coefficients were converted into 18 µS, while high filter coefficients were converted into 54 µS, considering that a scale factor of 2 relates the coefficients of the high-pass and low-pass filter (see 3.6). The initial conductance value of every filter coefficient was memorized with a maximum tolerable error of ±5%, and the mean number of intermediate steps required to program memory cells was 9.

An effective method for evaluating the above-mentioned long-term effects on PCM cells is to bake the memory array in a thermal chamber for some dozens of hours in order to accelerate the amorphization phenomena of the crystal lattice [36]. Recent studies have represented the behavior of PCM cells in time as a power model

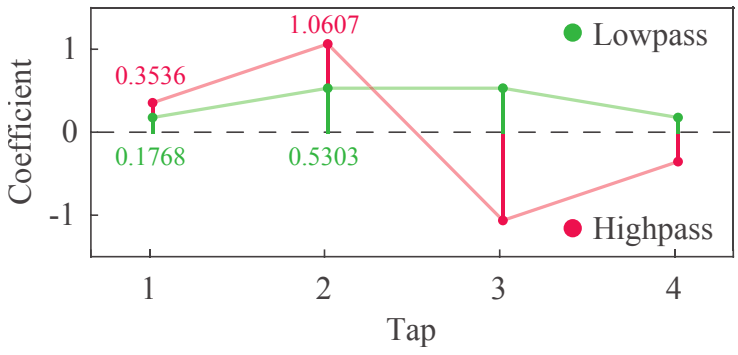

FIGURE 3.6: Reverse biortogonal 3.1 wavelet decomposition filters.

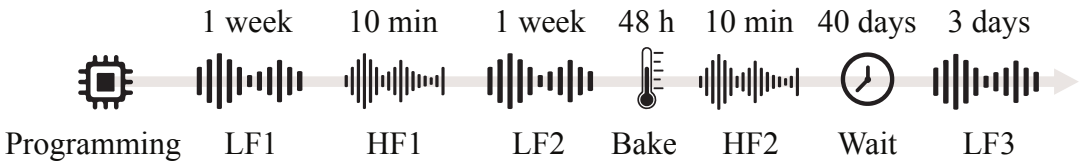

FIGURE 3.7: Observation schedule of programmed filters.

with the form [35].

The conductance of the PCM cells was observed using a current Source Meter Unit (SMU) in the laboratory following the time schedule reported in 3.7. The filter coefficients are collected with a sampling period of 6000 s in low sampling frequency (LF) observation intervals, while every 0.02 s in high sampling frequency (HF) intervals. Between LF2 and HF2, the memory array was baked for 48 hours at 150 °C to evaluate the effects of time-related non-idealities at an ideal infinite time after programming.

3.8 shows the conductance in time of all the monitored cells. Thin lines represent the behavior of individual cells, while the reference power law [35], fitted to the first two drift intervals, is represented as a thick line for high and low coefficients. According to the power law, the coefficients recorded during the interval LF3 (i.e., after bake and additional 40 days at room temperature) correspond to an equivalent observation time in the order of tens of years since programming. It is therefore assumed that short-term drift effects have completely vanished.

The coefficients observed in the two HF intervals are used to build the 6 low-pass (one for each transformation level) and 4 high-pass (only used in the first four transformation levels) wavelet filters employed in this study to filter the structural vibration response. Each filter is time-dependent due to a noise-related variability, as the stored coefficients are affected by the aforementioned non-idealities.

As explained, the signal can be decomposed into different wavelet components either using a set of equivalent filters corresponding to a given transformation level (i.e., batch approach) or performing a recursive procedure. The batch approach is

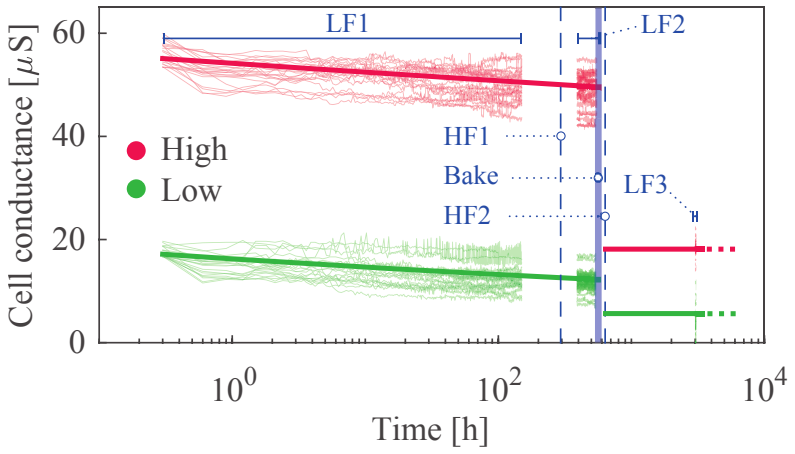

FIGURE 3.8: Drift of the programmed PCM cells.

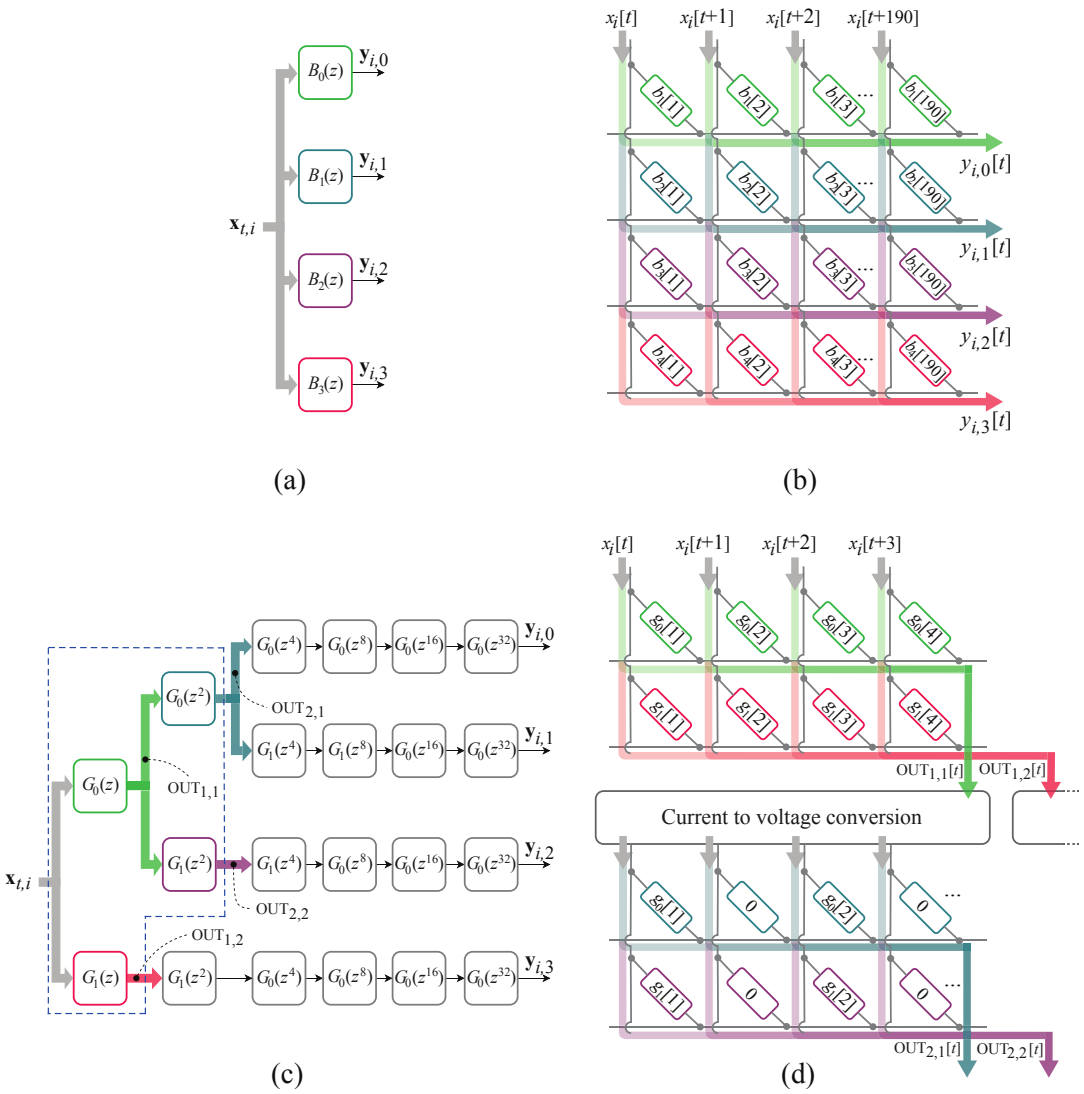

FIGURE 3.9: Filtering through the batch (a) and recursive (c) modes, and their respective implementation in a PCM-based architecture (b,d); in (d) the architecture of the dashed portion of (c) is represented.

represented schematically in 3.9a-b, and compared to the recursive procedure in 3.9c-d (the last figure shows only the first two levels of the transform). In this work, the recursive implementation of the signal decomposition task on the PCM-based architecture is proposed and compared with a batch implementation in terms of power consumption and accuracy of the results. Both algorithms are implemented using real observation of the filter coefficients in PCMs, collected as described in the PCM programming section, in the laboratory. The structures of the filtering algorithms were simulated in this study using the MATLAB environment. The input signal, consisting of pre-collected structural vibration data, is sampled and filtered using low-pass and high-pass wavelet filters in a fast wavelet transform implementation (see Equations (3.6) and (3.7)) to retrieve the signal components associated to a wavelet decomposition level equal to n (in this case, $n = 6$). If a batch procedure is adopted, the input samples are decomposed by m (in this case, $m = 4$) equivalent

filters whose impulse response is the inverse z-transform of $B_m(z)$ in Equation (3.8). In this case, the filter bank consists of $N_F = 4$ filters, each with $N_T = 190$ taps. The implementation of this strategy is shown in 3.9b, where 4 WLs and 190 BLs are required. On the other hand, the recursive implementation is represented in 3.9c. The filter bank consists of 6 layers, each of them having a different number of filters N_F, ranging from 2 to 4, with an increasing number of taps N_T, ranging from 4 to 97, with an increasing number of null values (3.9d). As illustrated in 3.9d, the coefficients of each filter are implemented in a single WL and different BLs, as every tap must be multiplied with a different value of the input signal. If two or more filters share the same input values (i.e., filters 1 and 2 in this case), they are programmed in different WLs, sharing however the same BLs. Thereby, their outputs are available at the same time and can be cast to the next filters. Between the two filter layers, a current-to-voltage conversion is processed.

In 3.1, the features of batch and recursive approaches are summarized, together with the number of non-zero coefficients per filter N_{ON}.

The recursive procedure has two principal advantages with respect to the memorization of equivalent filters: (1) it drastically reduces the power consumption of the sensing device, and (2) it reduces the noise effects of non-ideal PCM elements.

The performances in terms of power consumption of batch and recursive implementations were compared considering the energy required to entirely process a single input sample in both cases, neglecting the cost of current-to-voltage conversion steps. Assuming that the energy is given by $E = \int_0^T x_S I dt$, where x_S is the supplied voltage, I is a current and T is the operating time interval, the energy per input sample E' is

$$E' = \int_0^T x_S I dt = x_S K \bar{\imath} \tau \tag{3.11}$$

where K is the total number of taps to fully process the sample, $\bar{\imath}$ is the mean cell current, and τ is the time required by the PCM array to compute a single product. As x_S and τ are equal in both implementations, the product $K\bar{\imath}$ is the actual energy benchmark. In the batch implementation, $K = \sum N_F N_T = 760$ and $\bar{\imath} = 10.6\,\mu A$, whereas in the recursive procedure, $K = 1245$ and $\bar{\imath} = 0.61\,mA$, thus, the power required by the iterative strategy is only 9.43% of the power required by batch filtering, neglecting,

TABLE 3.1: Parameters of the batch and iterative filter banks.

Filter bank	Layer	N_T	N_F	N_{ON}
Batch	I	190	4	190
Iterative	I	4	2	4
	II	7	3	4
	III	13	4	4
	IV	25	4	4
	V	49	4	4
	VI	97	4	4

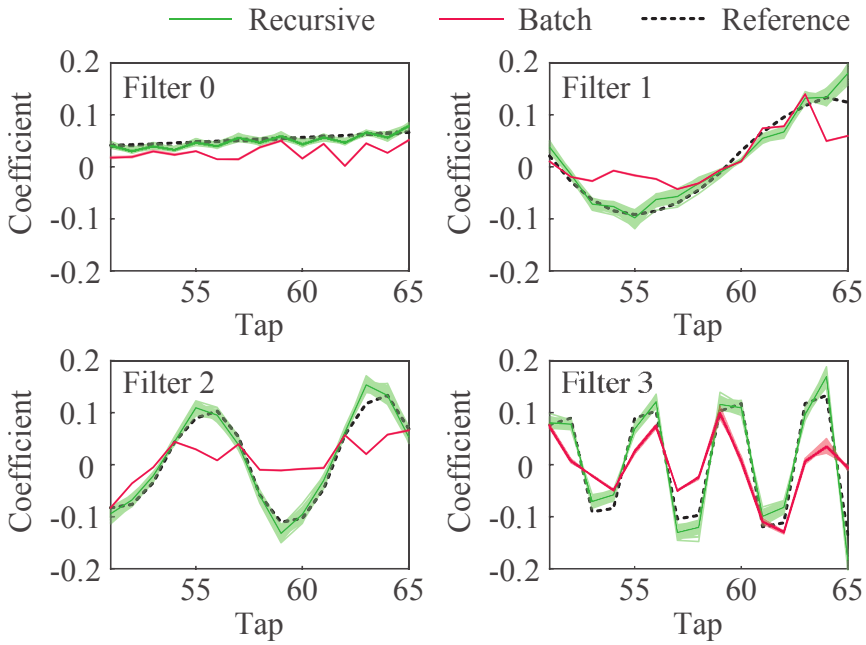

FIGURE 3.10: Noise effects on the equivalent filter for a level 6 transform.

in a first approximation, the contribution of current to voltage conversion circuits. In fact, even if the iterative implementation involves more taps than the batch procedure, the total required current is much lower as, according to Equations 3.8 and 3.9, and 3.1, a large number of coefficients are null, thus involving no current consumption.

In order to compare the performance of batch and recursive implementation, 15 samples of the 4 equivalent filters used in this study were stored in PCM elements and observed after a 48 h baking. 3.10 compares the observed interval (between tap 50 and 65) of the equivalent filter directly memorized in PCM elements (i.e., using a batch approach, see 3.9a) with the equivalent filter obtained by convolving the low-pass and high-pass coefficients observed in the interval HF2 according to 3.9c. Specifically, both for the recursive and batch implementation, the filter observed at 100 different time samples collected every 0.02 s is reported (light green and magenta spreads), together with their average (solid green and magenta lines). It is possible to observe that the coefficients of the filter obtained through recursive implementation are closer to the reference values (i.e., the ideal filter that does not account for the PCM non-idealities), although the spread (which represent the short-term noise) is generally higher. The selective performance of the four filters is observable in the frequency domain: 3.11 shows the equivalent filters obtained through a recursive implementation before and after baking. As in the previous representation, the filter coefficients observed at 100 different time samples are reported as spread and average lines. Although the spread increases after baking, the selective performance of the filters is comparable.

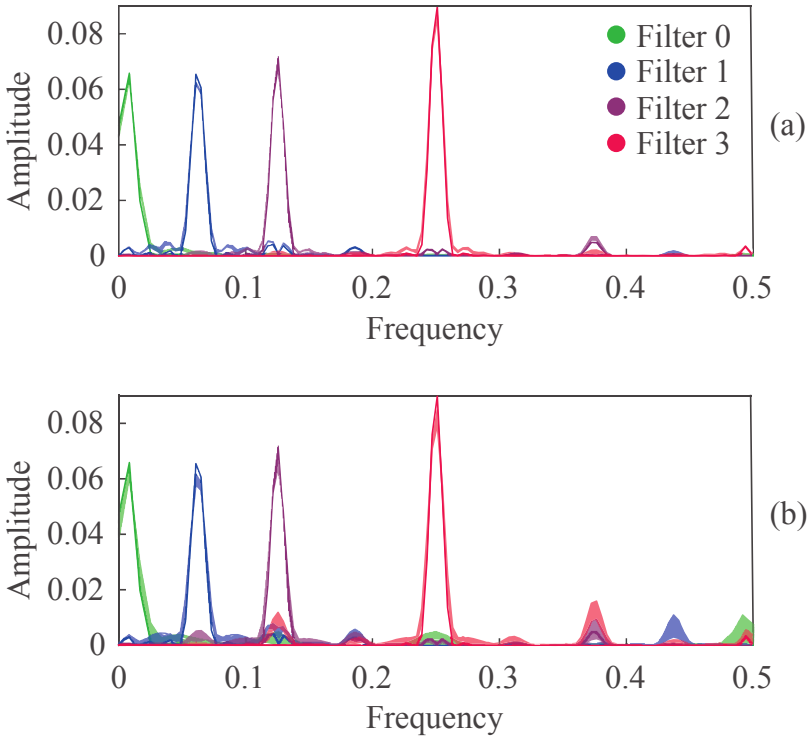

FIGURE 3.11: Selected filters in the frequency domain: pre-bake (a) and post-bake (b) environment.

3.3.2 Identification of structural parameters using PCM cells

This section presents the identification results obtained using the proposed algorithm on the experimental data collected on a viaduct of the Italian A24 motorway. Specifically, dynamic and quasi-static identification results are obtained using filters programmed and observed in the test PCM unit. These results are obtained using the memory cells in freshly programmed and long-term conditions, represented by pre-and post-bake environments (i.e., the observation intervals HF1 and HF2, respectively).

The viaduct, called Temperino [77]–[80], consists of a series of single-span post-tensioned prestressed beams in a simply-supported isostatic configuration.

Since this study is aimed at investigating the usability of PCMs in structural identification applications, the modal parameters identified using the proposed algorithm and implementation technology will be compared to reference parameters identified using a widely used algorithm for structural identification, namely, the FDD [81]. Precisely, a traditional centralized application of the FDD is employed using 10 acceleration time histories of 1500 s collected at all the locations, subsampled at 50 Hz. This method allows the identification of four vibration modes with natural frequencies $\bar{F}_m$ equal to 2.48 Hz, 5.06 Hz, 7.56 Hz, and 9.01 Hz.

In order to identify the mode shapes of the first, second, and fourth modes using the proposed method, the signal is resampled at a frequency of 41.5 Hz. This way, since $\bar{F}_1 \approx F_3$ and $4F_1 \approx 2\bar{F}_2 \approx \bar{F}_4$, the filters corresponding to a decomposition

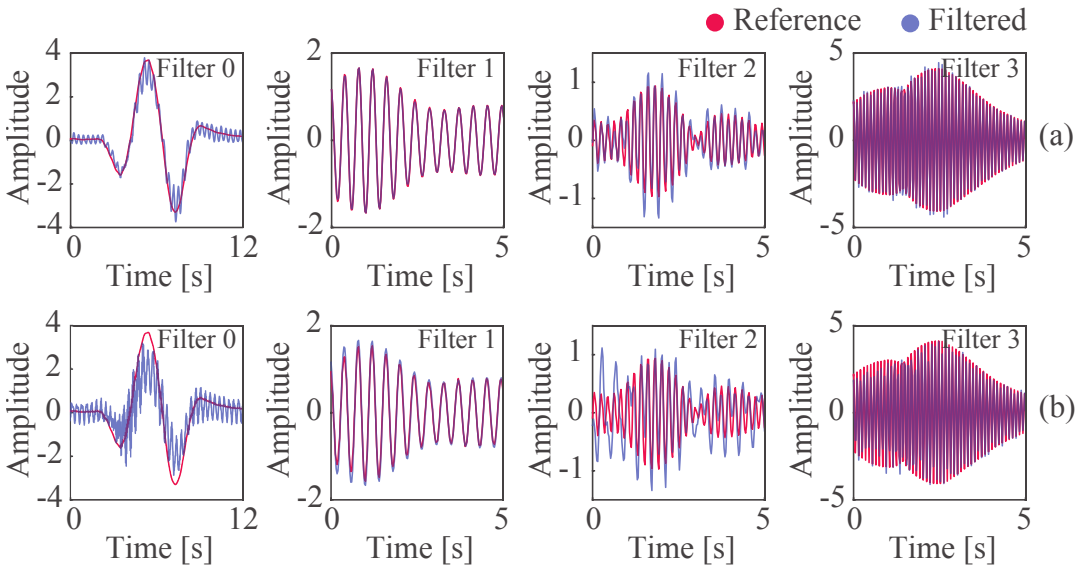

FIGURE 3.12: Filtered signals in pre-bake (a) and post-bake (b) environment; filter 0 indicates the low-pass filter, while filters 1, 2, and 3, are band-pass filters with central frequencies F1, F2, and F3, respectively.

level 6, with central frequencies $F_3 = 2.59$ Hz, $F_2 = 5.19$ Hz, and $F_1 = 10.38$ Hz, can be effectively employed to extract the modal contributions associated with the modes 1, 2, and 4, respectively. It should be noted that, in this study, it is assumed that the resonant frequencies of the structure (of a rough estimate of them) are already known, e.g., from previous monitoring campaigns, in order to design the filters for identification. This is a reasonable assumption since preliminary tests are usually performed before designing a monitoring system. A low-pass filter obtained for a decomposition level 5 is also employed to extract the quasi-static response component with a frequency lower than $F_s/2^6 = 0.64$ Hz.

Figure 3.12 shows time windows of the filtered signals obtained using the filters observed in the intervals HF1 and HF2 (i.e., in the pre- and post-baking environment), compared to the reference filtered signals obtained using ideal filters that do not include the noise generated by PCM cells. Moreover, Figure 3.13 shows the error of the filtered signal for each filter. Specifically, nRMSE represents the normalized RMS error. The normalization is obtained by dividing both the reference and the filtered signals by their standard deviation. It is possible to observe that the low-pass filter is generally affected by a higher noise level and, as expected, the noise increases in the post-bake environment. Moreover, the nRMSE of filter 1 is generally the lowest, denoting a good quality of the extracted first modal contribution.

Although the error in the filtered signal is non-negligible, the mode shapes reconstructed using the extracted modal contributions (3.14) are very close to the reference ones – obtained using the traditional FDD – both for the pre- and post-bake environments. In 3.14, the sign of mode shapes is determined using the sign identified through the preliminary FDD-based identification. The high accuracy is confirmed

using the MAC [82], [83]. 3.15 shows that values close to 1 are obtained comparing the reference and identified shapes, especially for the first two modes. Since the identification method provides absolute values of the modal amplitudes, their sign is determined based on the reference identified values.

It should also be noted that, although the central frequencies of the filters do not correspond exactly to the resonant frequencies of the structure, the identification results are in good agreement with the reference parameters. The method is therefore also robust to slight variations of the resonant frequencies, *e.g.*, due to varying temperature conditions.

Figure 3.16 shows the influence lines identified in pre- and post- bake environments. In particular, the average results are obtained considering 24 individual estimates computed during as many vehicle crossings. Although the estimates are visibly affected by noise compared to the reference estimates, the maxima of the influence lines are in the right location. Also, the results obtained in the pre- and post-bake environments are very similar to each other, denoting a good performance of the algorithm for long-term applications. The literature has already shown that, although the noise level can be high in quasi-static parameters, they are generally very sensitive to structural damage. Moreover, considering a larger set of individual estimates, the noise level would decrease.

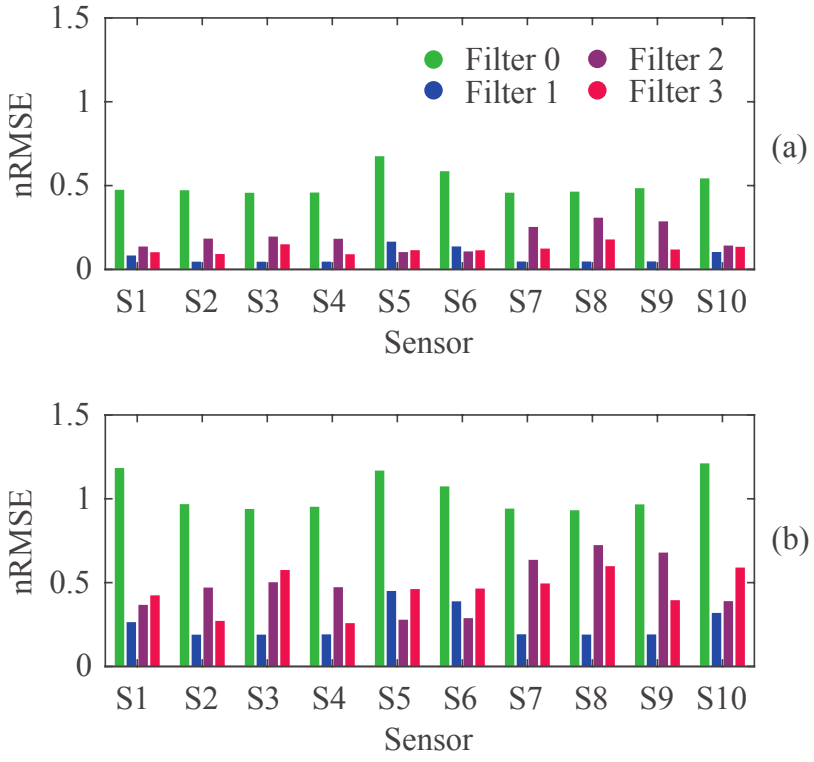

FIGURE 3.13: Normalized root mean square error of the filtered signals in pre-bake (a) and post-bake (b) environment.

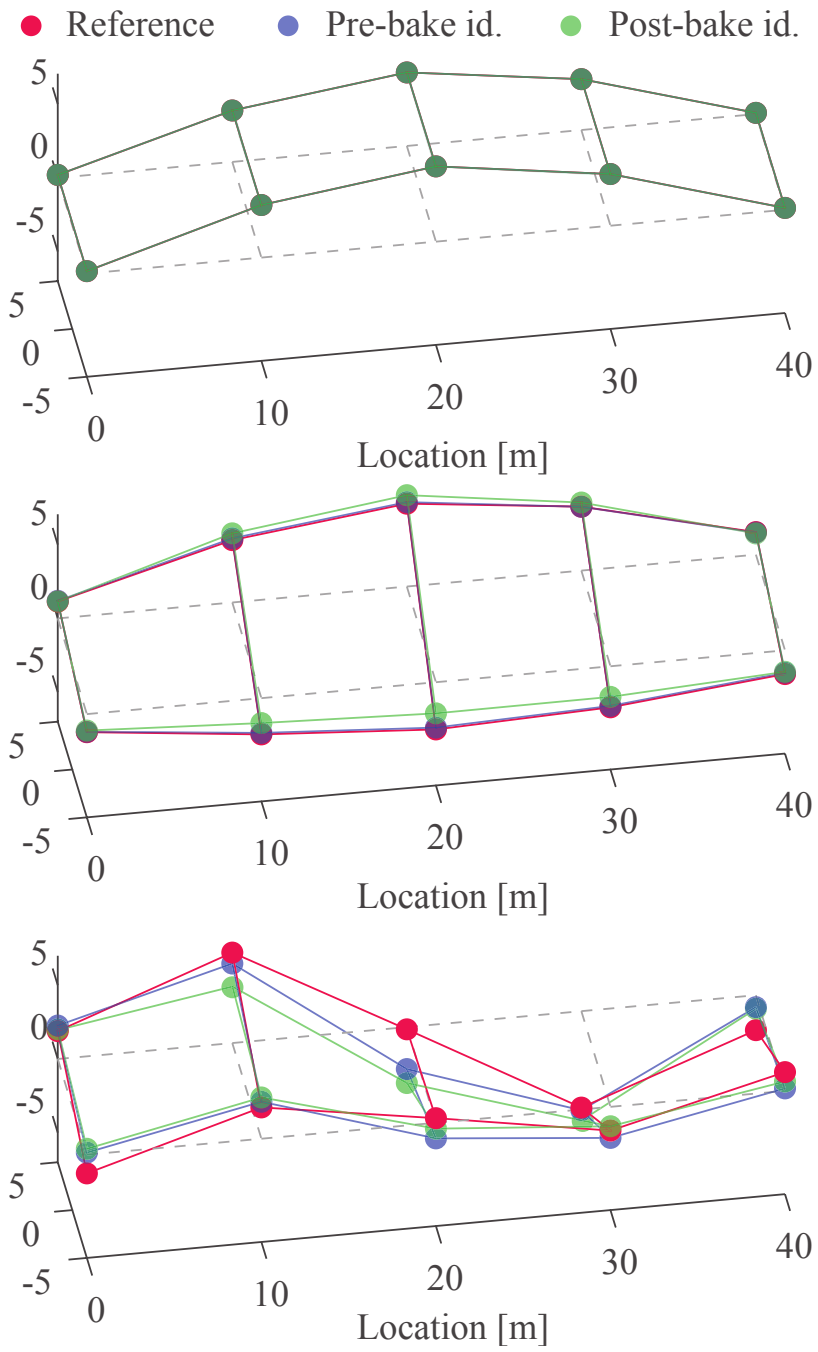

FIGURE 3.14: Reference and identified mode shapes; from top to bottom, output of filters 1, 2, and 3.

3.4 Conclusion

In this Chapter, an analysis of PCM devices being employed in two testcase applications has been carried out. In the first one, we have proposed a way of including arbitrary synapse models within a neural layer, targeting specifically phase-change memory devices. Two test setups have validated the procedure, a classification task on the Fashion-MNIST dataset and an artificially constructed regression task. The

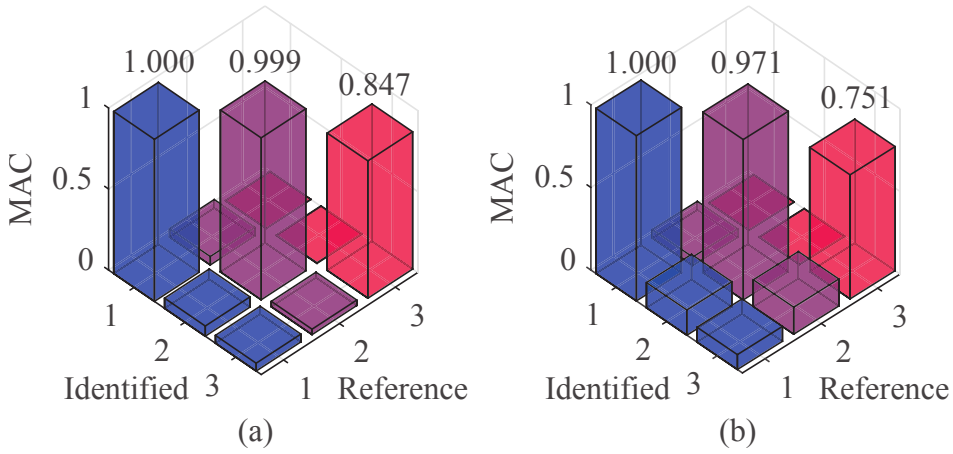

FIGURE 3.15: Modal assurance criterion matrices calculated between identified and reference mode shapes in pre-bake (a) and post-bake (b) environment.

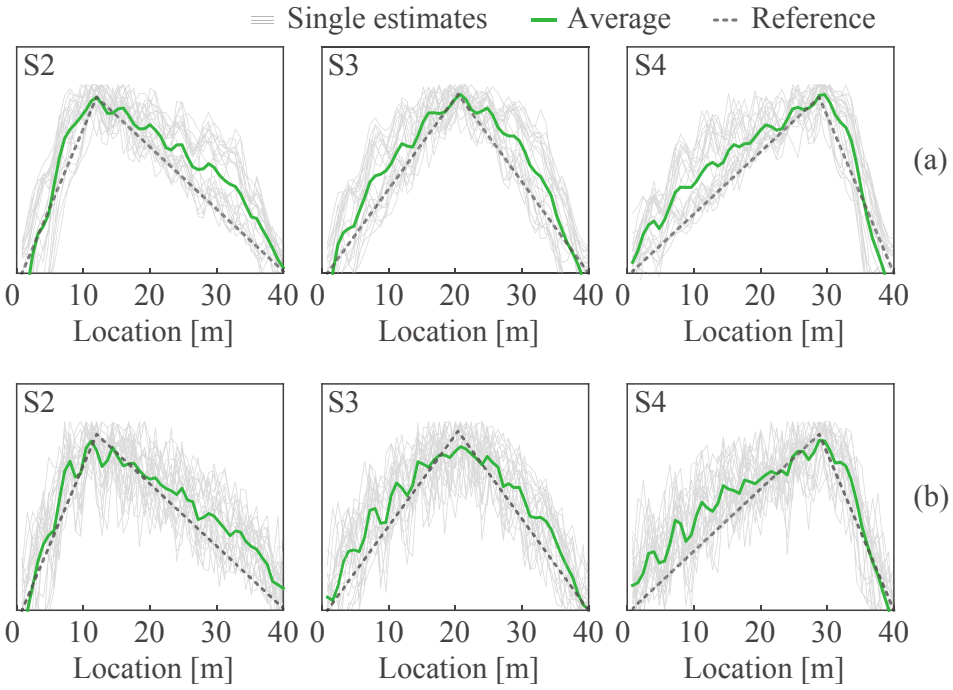

FIGURE 3.16: Influence lines identified in pre- bake (a) and post-bake (b) environment for different sensor locations.

injection of noise on the trained weights has highlighted the robustness of the networks to a point that makes the devices promising candidates in actual circuital implementations. Then, an identification procedure of modal and quasi-static structural parameters employing recursive filtering has been proposed, implemented through PCM cells, that have been used for the first time in this research field. Specifically, this study shows that a recursive implementation improves filter accuracy, also reducing energy consumption. The challenges related to time-dependent non-idealities of PCM devices are also investigated. Structural parameters identified in two environments, showing that the PCM does not necessarily need to be freshly

programmed for SHM applications. Therefore, energy-consuming periodic reprogramming can be avoided, even under the effects of cells drift.

Chapter 4

Design and testing of an embedded AIMC unit based on PCM cells

This Chapter presents an integrated peripheral unit interfaced to an embedded Phase-change Memory (ePCM) macrocell, with the aim of adding Analog In-memory Computing (AIMC) feature without any modifications to the internal structure of the memory array. The testchip has been designed and manufactured in a 90-nm STMicroelectronics CMOS technology. The unit allows the execution of signed Multiply and Accumulate (MAC) operations at the edge of the memory array exploiting the physical characteristics of memory devices. I-V characteristic non-linearity and transconductance time drift of PCM cells are overcome through a regulated bitline readout circuitry with time-coded inputs, along with a drift compensation technique based on a conductance ratio. The testing setup of the prototype is as well described, along with a brief discussion on issues and future developments. To validate the employment of the proposed hardware solution in the field of AIMC, measurement-based models are exploited to emulate the prototype use in the field of Deep Neural Networks (DNNs).

Some of the material reported in this Chapter is reused from [65], in agreement with IEEE copyright on theses and dissertations.

4.1 AIMC unit implementation

The aim of the proposed AIMC unit is to overcome the limitations of the basic architecture illustrated in Chapter 3. The peripheral unit is interfaced with a 128-kB embedded PCM (ePCM) array in a 90-nm STMicroelectronics CMOS technology, with the purpose of executing one-step MAC operations with both signed inputs and coefficients. The developed testchip is mainly intended to demonstrate a readout technique for non-linearity and time drift compensation, which differs from solutions based on empirical models and post-processing compensation [84]–[86]. Moreover, the unit is conceived to avoid any changes of the internal structure of the memory.

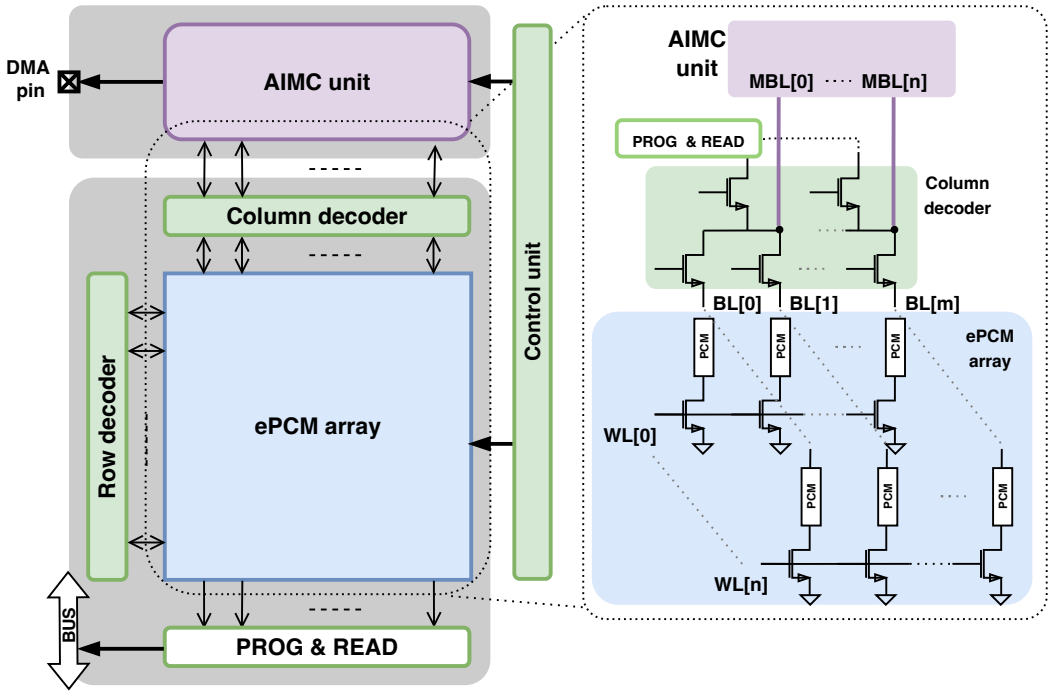

FIGURE 4.1: Left: Block diagram. Right: Simplified schematic of the array architecture and column decoder. DMA pin is used to access various internal analog signals.

4.1.1 Testchip structure and interface to the ePCM array

Figure 4.1 shows a simplified schematic of the 128-kB ePCM array architecture [54] and AIMC unit interface. The AIMC unit is directly connected to the main bitlines (MBL) and during AIMC computation standard ePCM read and program operations are disabled. To perform MAC tasks, the AIMC unit sets the voltage of each MBL and reads the current of the cells belonging to the addressed word line (WL). Unlike other works [85]–[87], where MVM is performed in a single step, the proposed solution implements a MVM with multiple consecutive MACs; this requires a sequential activation of different WLs, but prevents the row decoder from being modified, so that the ePCM can be employed as a binary memory as well.

4.1.2 MAC computation architecture

The proposed architecture, shown in 4.2, is designed to perform a single signed MAC operation:

$$Z = \mathbf{w}_j \cdot \mathbf{x} = \sum_{i=1}^{n} w_{j,i} x_i \tag{4.1}$$

The input array $\mathbf{x} = [x_1, \ldots, x_n]$, where x_i are 5-bit signed data, is stored in the control unit. A set of DACs converts the 4-bit absolute value of each x_i to analog value V_i, while the sign bits $x_{i,sign}$ are directly connected to the readout circuit. 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

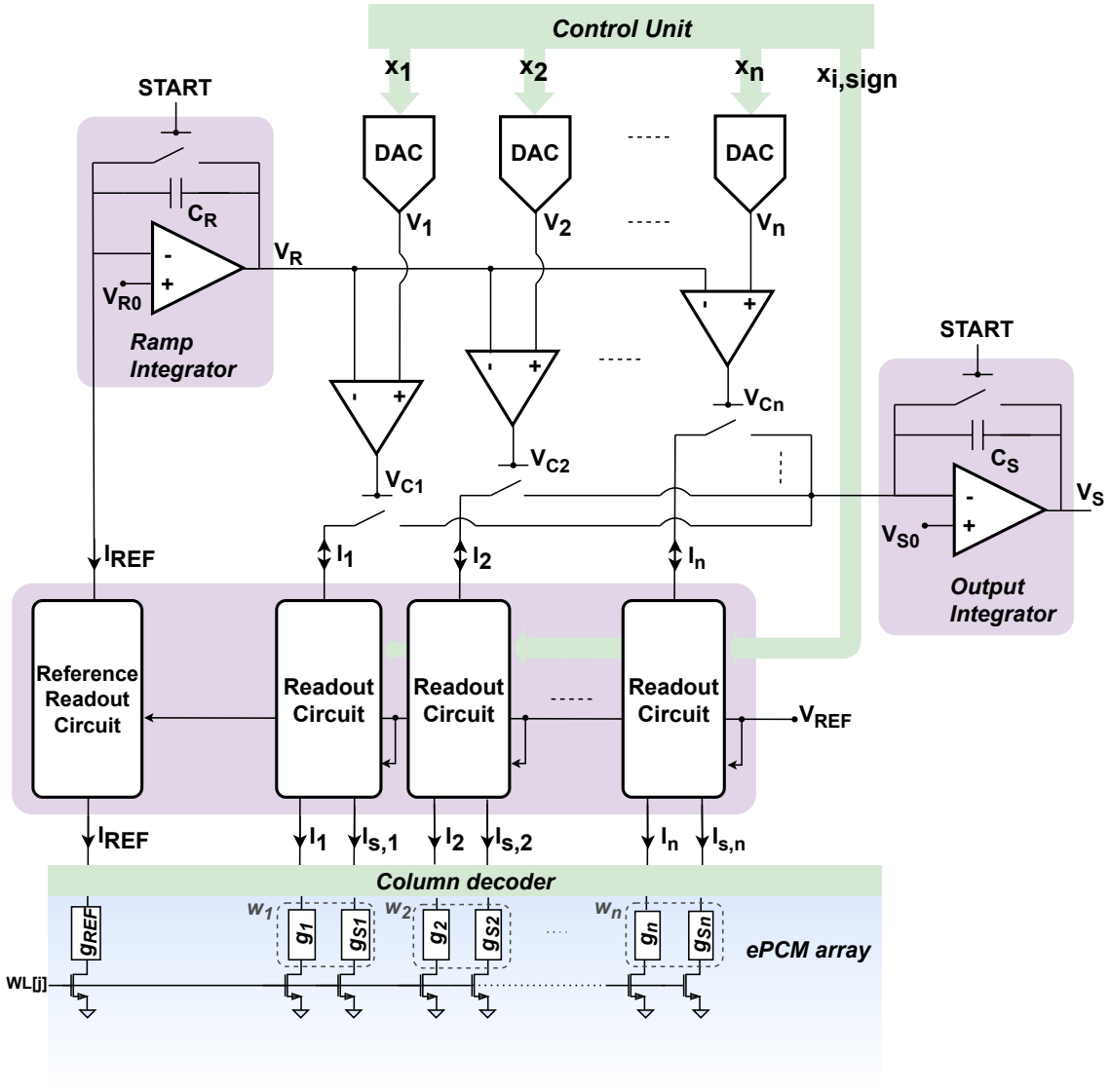

FIGURE 4.2: Block diagram of the AIMC unit architecture.

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:

$$w_{j,i} = \begin{cases} g_{j,i} & \text{if } g_{S,j,i} < g_{\text{th}} \\ -g_{j,i} & \text{if } g_{S,j,i} \geq g_{\text{th}} \end{cases} \tag{4.2}$$

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 the next Paragraphs.

The Reference Readout Circuit sets the read voltage V_{REF} across the reference conductance g_{REF}. According to Figures 4.2 and 4.3, when the START signal switches to logic low, a current $I_{REF} = g_{REF} V_{REF}$ is integrated on capacitance C_R, generating a ramp signal V_R starting from voltage V_{R0}:

$$V_R(t) = \frac{I_{REF}}{C_R}t + V_{R0} \tag{4.3}$$

The same reference read voltage V_{REF} is applied across each weight cell g_i through n Readout Circuits. Each current $I_i = g_i V_{REF}$ is then sourced to or sunk from the output integrator circuit according to the sign of the product $w_i x_i$, which is obtained combining the corresponding sign cell g_{Si} value and $x_{i,sign}$ sign bit, as described . Current I_i is then integrated on capacitance C_S for a time window T_{ON_i}, which begins at the START falling edge and ends when the output $V_{C,i}$ of the i-th comparator switches to logic low, i.e., when $V_R(t) = V_i$. According to (4.3):

$$T_{ON_i} = \frac{(V_i - V_{R0})C_R}{I_{REF}} = \frac{(V_i - V_{R0})C_R}{g_{REF}V_{REF}} \tag{4.4}$$

is the time-coded version of V_i. Summing all the n currents $\pm I_i$, the output variation $\Delta V_S = V_S - V_{S0}$ is:

$$\Delta V_S = \sum_{i=1}^{n}\left[\pm\frac{I_i T_{ON_i}}{C_S}\right] = \frac{C_R}{C_S}\sum_{i=1}^{n}\left[\pm\frac{g_i}{g_{REF}}(V_i - V_{R0})\right] \tag{4.5}$$

Considering $V_i - V_{R0}$ and g_i/g_{REF} as the absolute values of x_i and w_i, respectively, one can obtain that:

$$\Delta V_S = \frac{C_R}{C_S}\sum_{i=1}^{n} w_i x_i = \frac{C_R}{C_S}Z \tag{4.6}$$

is therefore proportional to the signed MAC operation Z.

4.1.3 Drift compensation

As previously discussed in Chapter 2, the drift of a generic cell conductance $g(t)$ has been shown to follow the power law $g(t) = g_0(t/t_0)^{-\alpha}$ [33], where g_0 is the conductance at arbitrary initial time t_0, and α is the drift coefficient, which is positive and cell-to-cell variable. The MAC result is proportional to the conductance ratio g_i/g_{REF}, and combining the drift model of $g(t)$ with 4.5, the MAC operation evaluated at time t_1 after t_0 becomes:

$$\Delta V_S(t_1) = \sum_{i=1}^{n}\left[\pm\frac{I_i T_{ON_i}}{C_S}\right] = \frac{C_R}{C_S}\sum_{i=1}^{n}\left[\pm\frac{g_{0,i}}{g_{0,REF}}\left(\frac{t}{t_0}\right)^{-(\alpha_i - \alpha_{REF})}(V_i - V_{R0})\right] \tag{4.7}$$

where $g_{0,i}$ and $g_{0,REF}$ are the weight and reference cells conductance at t_0, respectively. Therefore, each resulting drift coefficient is reduced to $(\alpha_i - \alpha_{REF})$, and drift is partially compensated. In other words, the slope of ramp $V_R(t)$ decreases accordingly to the reference cell conductance drift; this leads to an increase in integration time T_{ON_i}, which compensates for the drift-induced drop of weight cells currents.

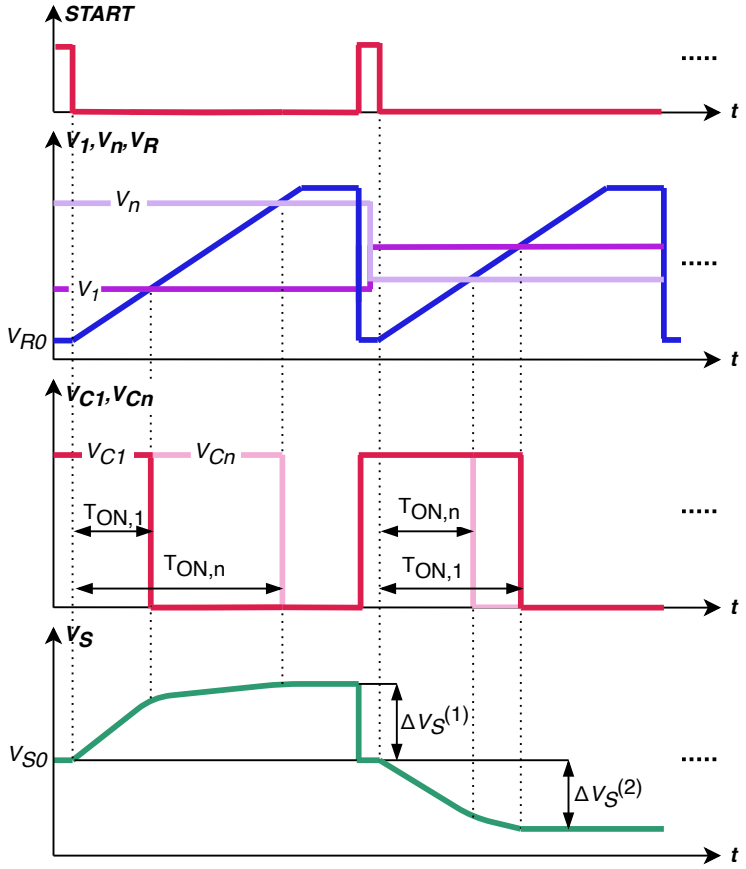

FIGURE 4.3: Sketch of waveforms showing two consecutive MAC operations. The first one represents a positive MAC, while the second a negative one.

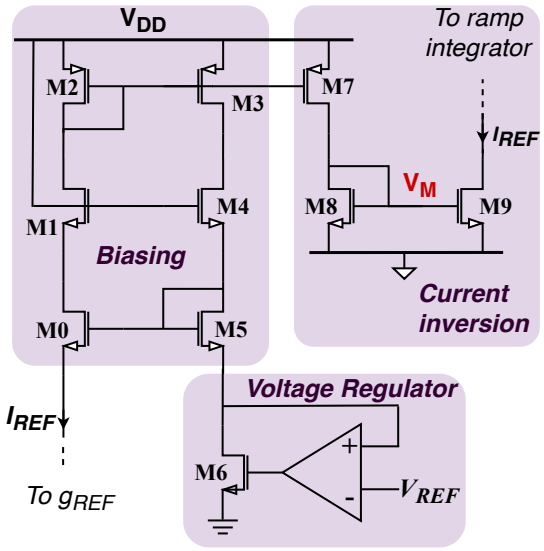

FIGURE 4.4: Schematic of reference readout circuit.

Moreover, the adoption of time-coded inputs T_{ON_i}, along with cells being read at fixed voltage, addresses cells I-V characteristic non-linearity issue.

4.1.4 Reference and Readout circuit with sign management

The detailed schematic of the Reference Readout Circuit is shown in Figure 4.4. The biasing circuit along with the voltage regulator allows to read the reference cell at a fixed reference voltage level V_{REF}. The reference voltage V_{REF} applied to the source terminal of transistor M5 is generated using a voltage regulator circuit, composed of an operational amplifier and transistor M6. Feedback from the source of transistor M5 is provided to the non-inverting input of the amplifier, while the inverting input is connected to V_{REF}, which is generated from a bandgap circuit. Current mirrors M5-M0 and M2-M3 provide voltage feedback that forces the gate-to-source voltages of transistors M0 and M5 to be equal. Thus, neglecting voltage drop through column decoder shown in Figure 4.1 right, reference voltage V_{REF} is applied to the reference cell too. The current mirroring ratio of both current mirrors is 10:1, which is the same used between transistors M2 and M7 to provide I_{REF} to the ramp integrator.

Figure 4.5 shows one of the n Readout Circuits. The biasing and voltage regulator circuits are equal to those previously described and apply V_{REF} to the selected cells. Moreover, the n biasing circuits share a single voltage regulator. The current switching and sign generator circuits allow to manage both the input and the weight signs. Sign cell current I_{Si}, which is the bit line current from the weight bit cell g_{Si}, is mirrored by current mirror M14-M15 and compared to reference current I_{REF}, mirrored from reference readout circuit to M13 by means of voltage V_M. The result of the current comparison is a logic signal $w_{i,sign}$ that represents the sign of w_i. When $I_{Si} < I_{REF}$, this being indicative of a positive sign, $w_{i,sign}$ voltage value is logic low. Whereas, when $I_{Si} > I_{REF}$ (i.e., a negative sign), $w_{i,sign}$ is logic high. $w_{i,sign}$ is then

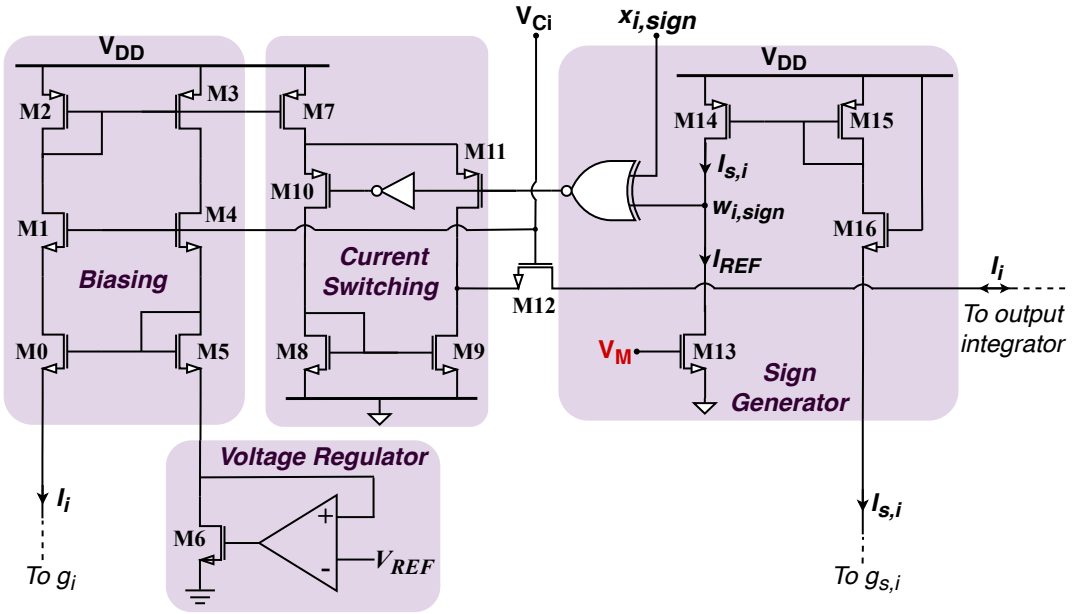

FIGURE 4.5: Schematic of the cells readout circuit, with sign generation system.

combined with the sign bit $x_{i,sign}$ to produce a control signal applied to transistors M10 and M11 of the current switching circuit. Thus, the sign of the multiplication, as summarized in the Table 4.1, determines the direction of current I_i applied to the integrator.

4.2 Testchip implementation and control

The testchip includes a single 128-kB ePCM array interfaced with the AIMC unit, and it is mainly intended to validate the proposed drift compensation technique. In this first prototype, the dimension of the input and coefficient arrays is $n = 12$. Circuits have been designed with $V_{DD} = 1.2$ V and $V_{REF} = 0.3$ V, leading to PCM cells currents ranging from hundreds of nA to 10 μA. The minimum time required to perform a single MAC operation is 150 ns, and depends on the reference conductance value, as shown in (4.4), while the maximum output voltage ΔV_S^{MAX} is $\pm$400 mV.

An additional 5-V power supply is used for the DMA output buffer (later explained), whereas a 1.203-V bandgap voltage is employed to generate the analog voltages of the input signals DACs, and it is conveyed to a voltage buffer, so that no

TABLE 4.1: Inputs and coefficients signs management.

$w_{i,sign}$	$x_{i,sign}$	$x_i w_i$
0	0	> 0
0	1	< 0
1	0	< 0
1	1	> 0

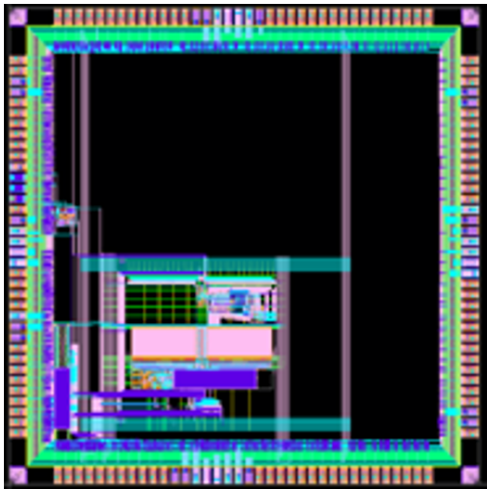

FIGURE 4.6: Layout of the whole testchip, including both AIMC unit and PCM IP.

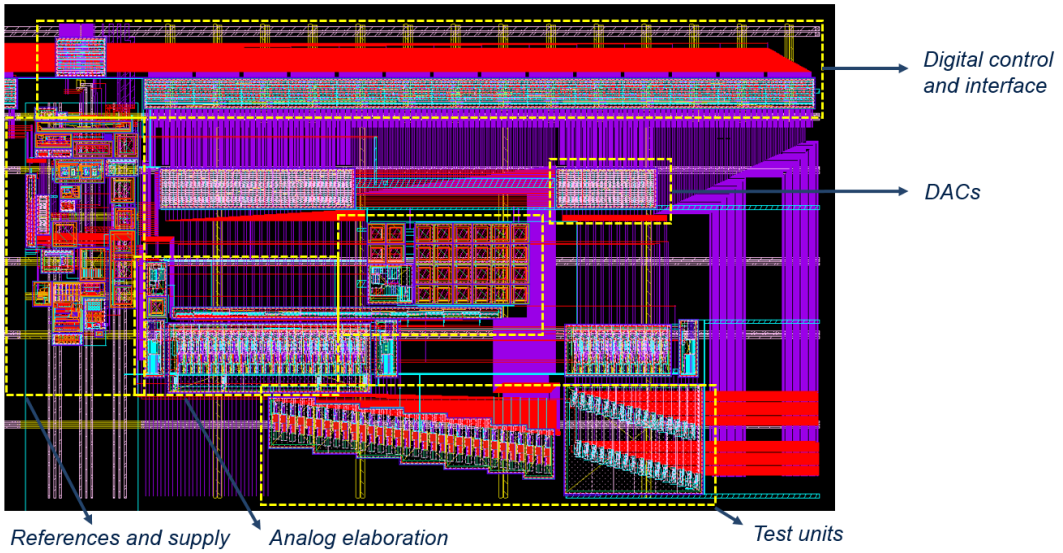

FIGURE 4.7: Detailed layout of the AIMC unit; the PCM IP is not shown.

current is absorbed by the external generator towards the circuit. The layout of the testchip is reported in Figures 4.6 and 4.7, where the ePCM is included as well.

4.2.1 Digital interface

A digital and interface unit was designed to perform the following functions:

- Storage of the binary data which, once converted to analog, constitute the input voltage values to the calculation modules;
- Storage of the configuration bits of the test units;
- Interface the AIMC unit with the external environment, thus propagating the external digital signals that coordinate the operation of the analog circuits within the unit.

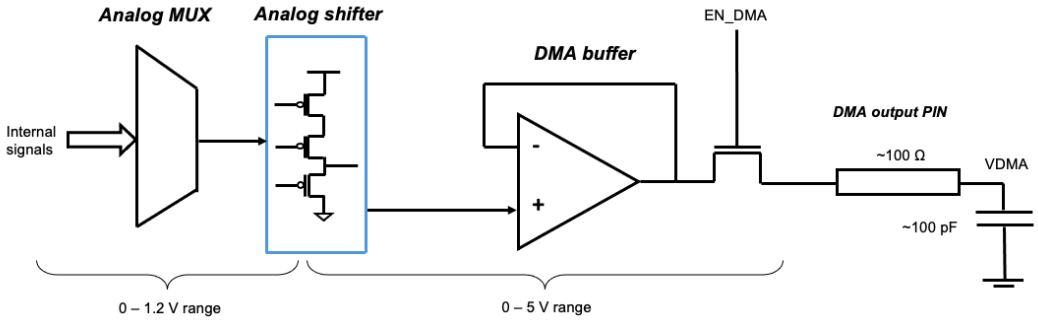

FIGURE 4.8: Connection between the 1.2-V internal signals and the DMA output pin. The estimated DMA pad capacitance and resistance too are shown.

The read and write procedures of the AIMC registers are compliant with the AMBA AHB protocol, as it is used in the STMicroelectronics testchip data-bus and address-bus. The read operation is used to access the contents of both the user and configuration registers. To access the AIMC registers, on the edge of the system clock a write enable signal must be logic low, a selection signal must be logic high, together with a valid address. Read access to the registers is performed at each clock cycle, with the address sampled on the rising edge of the clock. Read access to the registry can be done at 1-clock cycle speed. To write the AIMC registers, a write enable and a selection signal must be high on the edge of the system clock, and an address must be valid. Read access to the memory is performed at each clock cycle with the address sampled on the rising edge of the clock and the data are available on the data bus on the next clock cycle. The clock frequency is equal to 25 MHz.

4.2.2 Digital-to-analog converters

The input D/A converters have been designed using a series of complementary pass-transistors, which deliver to the AIMC unit input voltages V_i, in a range of 200-575 mV with a resolution of 25 mV. The size of the input strings is 4-bit, therefore the possible output voltage levels are equal to 16. The D/A converters also provide the input sign bits to the single analog cell without the need to convert it, accordingly with the aforementioned circuital implementation. The resistor divider that generates the analog input voltages is powered with the bandgap voltage; the resistors have been sized so that the current absorbed by this branch is equal to 10 μA.

4.2.3 Design for testability

4.2.3.1 Internal signals accessibility

The design of the AIMC unit has been conceived to observe several internal nodes on the analog DMA output pin, which is the only available output, due to the pinout structure of the testchip. The electrical connection between the internal resources

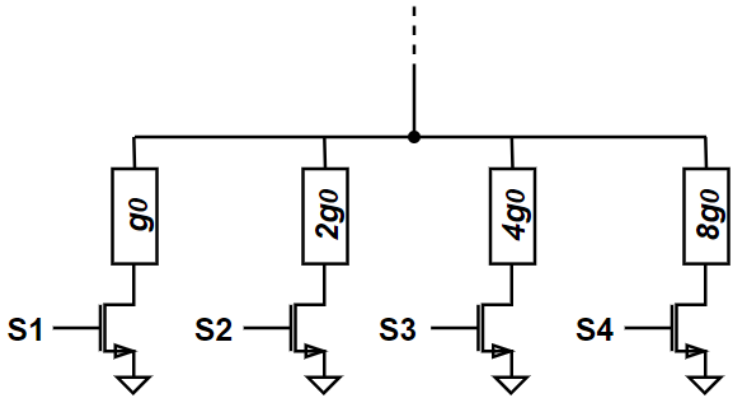

FIGURE 4.9: Single test conductance of the AIMC test unit. Signals $S1, ..., S4$ represent the conductance programming binary string.

and the DMA pin is outlined in Figure 4.8, and it allows the selection, through an analog multiplexer, of the internal signal to be observed. The DMA pin is accessed through an output voltage buffer, which requires a 5-V power supply, and a 5-V pass transistor, enabled by the control signal EN-DMA. The DMA buffer is optimized for an input voltage range of [1.2 - 5] V approximately. The whole AIMC unit is supplied with a $V_{DD} = 1.2$ V voltage, consequently, the internal signals range between 0 and V_{DD}. In particular, the ramp V_R and the analog inputs vary between 200 and 600 mV, whereas the output voltage V_S between 0 and 900 mV, being the output integrator bias voltage $V_{S0} = 450$ mV (see Figure 4.2). An analog level shifter connects the 1-2 V multiplexer to the output buffer. The level shifter is composed of a 5-V pMOS common source stage, which shifts the input signal up to the input MOS threshold voltage.

4.2.3.2 Test unit

In order to analyze the performances of the computation technique, a test unit has been added to the AIMC unit, with the aim of neglecting the effects of the PCM devices employed as MAC coefficients and reference conductance. To do this, an array of programmable conductances has been added to the design. The single conductance element, depicted in Figure 4.9, can be programmed among 16 conductance levels through a 4-bit input string, which comes from the digital interface. The conductance levels are obtained with the parallel-combination of four binary-weighted conductances of values g_0, $2g_0$, $4g_0$ and $8g_0$. The test conductances can replace either the MAC coefficients g_i and the reference conductance g_{REF}. Consequently, the AIMC unit can operate exploiting both test conductances and PCM elements, through the REF-MODE and WEIGHT-MODE control bits, according to Table 4.2. Some of the listed computation modes will be exploited in the characterization results Section.

4.2.4 Testchip control

4.2.4.1 Power-up sequence

The whole testchip activation sequence consists of the 5-V power supply activation, followed by the 1.2-V one; after that, the bandgap voltage is supplied. To turn the AIMC unit on, the internal power down signal is first deactivated; the reset is kept at logic low in order to reset the initial conditions of the digital modules. When the reset is released, it is possible to operate with the AIMC unit control signals, which will follow the sequence illustrated in the next Paragraph.

4.2.4.2 AIMC operations control sequence

The AIMC unit operating phases have been designed so that the AIMC module does not affect the normal execution of operations by the ePCM module. The operations of the AIMC unit can be organized as follows:

- Memory array programming phase: the ePCM memory must be programmed with a specific programming algorithm.
- Write (and read) of the internal digital registers: the digital inputs and the configuration bits must be written on the digital interface registers.
- Array-circuit interfacing and execution of the calculation: the analog modules receive the converted data from the DACs (enabled by two control signals named ELAB and RE-INREG), as well as the power supply by the PCM cells is enabled. Once the integrators are enabled with START signal, the calculation is performed.

The phases and conditions imposed on the control signals are summarized in Table 4.3, and are established by the listed control signals and their management is entrusted to the external control of the testchip through a special evaluation board. The switching frequency of the signals and therefore of the states has a frequency determined by the execution of specific firmware. This frequency is estimated in the order of 10 KHz.

TABLE 4.2: AIMC unit computation modes.

REF-MODE	WEIGHT-MODE	AIMC mode
0	0	Reference: test conductance Weights: test conductances
0	1	Reference: test conductance Weights: PCM cells
1	0	Reference: PCM cells Weights: test conductances
1	1	Reference: PCM cells Weights: PCM cells

4.3 Testchip validation

4.3.1 Testing procedure

To perform automated measures on the testchip, the same GUI employed in Chapter 2 has been customized with additional features. In particular, the GUI allows the observation on DMA pin of some internal signals (power supply, V_{REF} voltage, analog inputs, ramp signal, and AIMC output). The software interface can be also used to configure the digital interface, i.e., to set the AIMC inputs and control signals, and to program the PCM cells involved in the MAC operations. An automated calibration of the output to neglect the effects of the level shifter is also implemented. Furthermore, it is possible to execute MAC operations using either the test unit resistances or the PCM cells. A snapshot of the GUI is reported in 4.10, while the testchip micrograph is shown in Figure 4.11, where the evaluation board is depicted as well.

TABLE 4.3: AIMC unit control sequence.

State	Operation	Control signal conditions
Idle	AIMC unit not used	ELAB = 0 START-AIMC = 0 RE-INREG = 0 EN-DMA = 0
PCM programming	ePCM cells standard read and write operations	ELAB = 0 START-AIMC = 0 RE-INREG = 0 EN-DMA = 0
AIMC registers configuration	Write and read operations of AIMC testchips	ELAB = 0 START-AIMC = 0 RE-INREG = 0 EN-DMA = 0
AIMC unit to array connection	Connection between analog modules and ePCM array enabled	ELAB = 1 START-AIMC = 0 RE-INREG = 1 EN-DMA = 0
Internal signal observation	Observation of ramp, analog inputs, power supply)	ELAB = 0 START-AIMC = 0 RE-INREG = 1 EN-DMA = 1
AIMC MAC computation	MAC operation enabled (ramp and ouput integrators enabled)	ELAB = 1 START-AIMC = 1 RE-INREG = 1 EN-DMA = 1

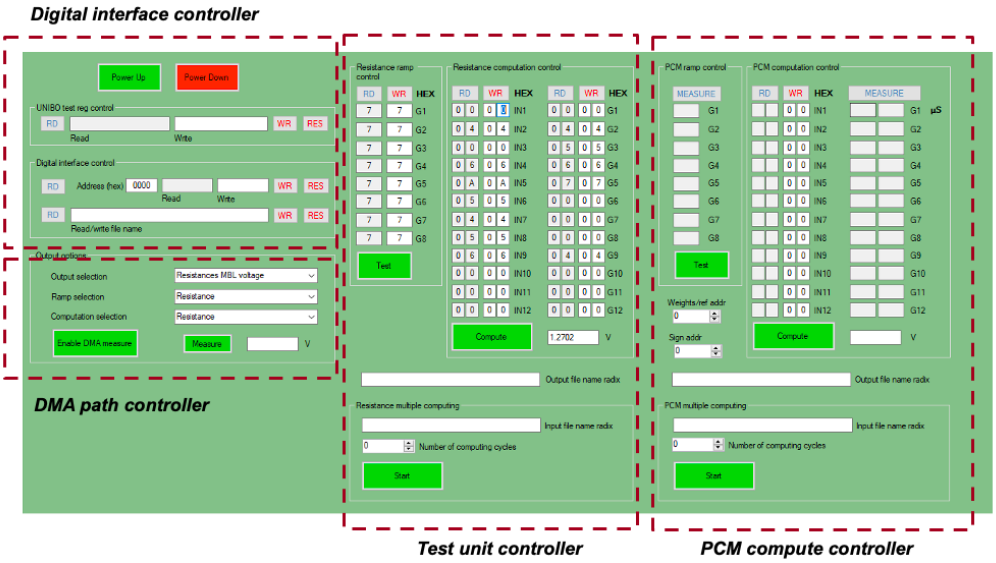

FIGURE 4.10: Snapshot of the developed GUI for AIMC unit testing.

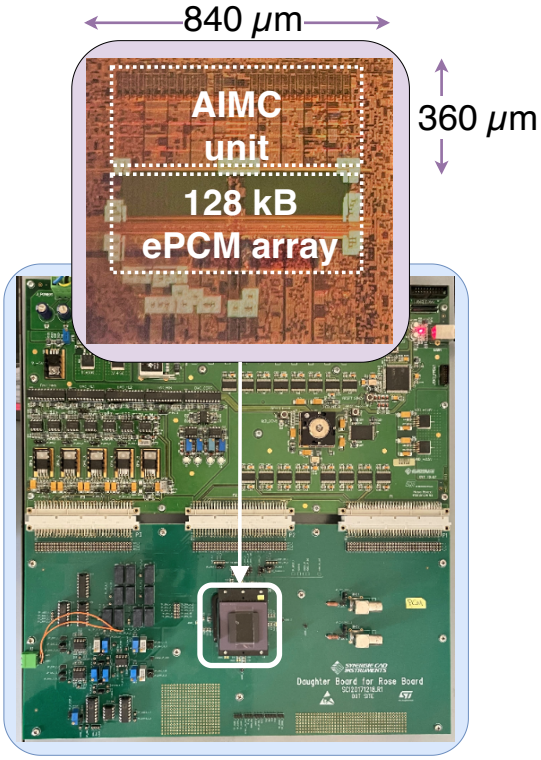

FIGURE 4.11: Die micrograph and evaluation board.

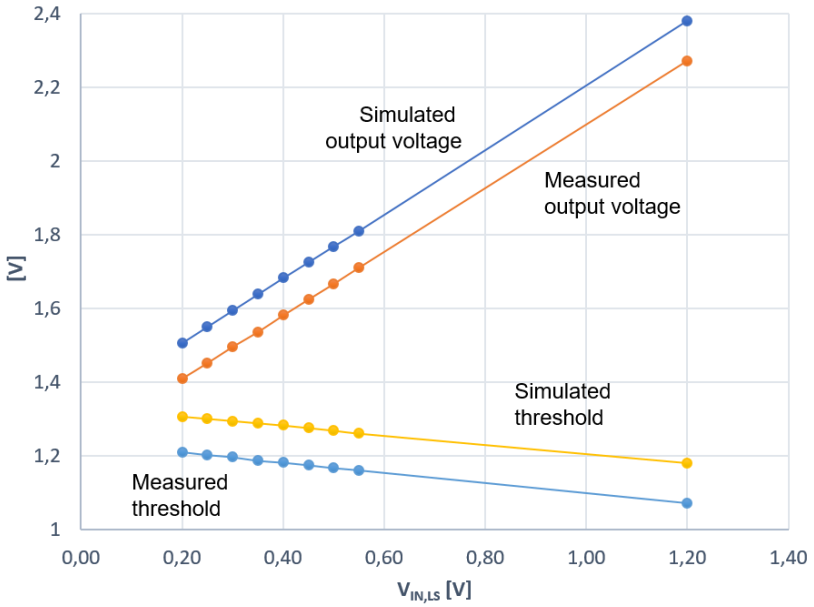

FIGURE 4.12: Simulation and experimental characterization of the analog level shifter.

4.3.2 Testing results

As previously mentioned, all the observable internal nodes are visible on DMA output pin through the DMA buffer, whose bandwidth is limited by the internal dominant pole and by the DMA pad to around 2 MHz. Consequently, signals that are faster than approximately 0.5 μs will be limited to the buffer bandwidth. This is not a limiting factor for what concerns the MAC accuracy evaluation, as only the starting and the final values of V_S are relevant.

Another important aspect to be considered in the testchip evaluation is the aforementioned analog level shifter. This block is designed to shift its input signal up to a quantity corresponding to the input pMOS threshold voltage V_{th}. This quantity is expected to depend on the input signal itself, as shown in Figure 4.12, where it is evident that the threshold voltage V_{th} shows a linear dependence on the input voltage. The experimental characterization of the level shifter has been performed measuring on the DMA pin the analog input voltages of the AIMC unit, along with the 1.2-V power supply; these values are reported on the x-axis of Figure 4.12, whereas on the y-axis both simulated and measured values of both output and threshold voltages are shown.

Measurements related to the analog inputs are reported in Figure 4.13, where eight consecutive levels of V_i are shown. Figure 4.14 shows instead the MBL voltage during a MAC operations, which is correctly kept at a constant V_{REF} voltage, validating thus the readout circuit previously presented. Finally, different acquisitions of ramp signals V_R are shown in Figure 4.15, where the slope of the ramp varies in accordance with the chosen g_{REF} level.

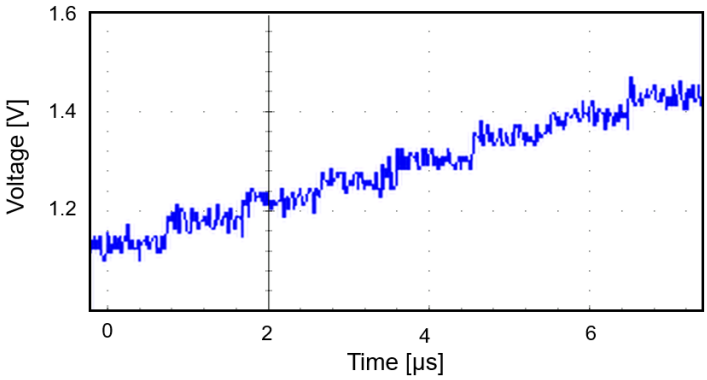

FIGURE 4.13: Measurement of the analog inputs.

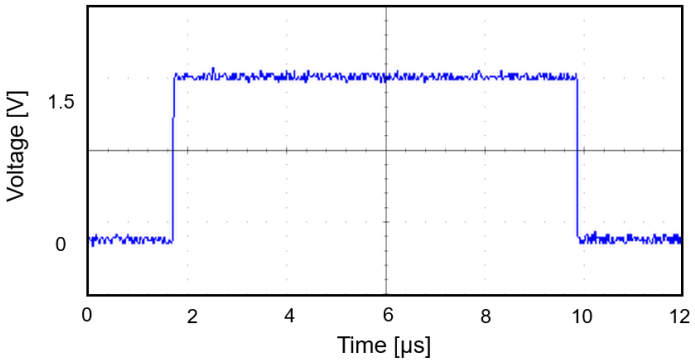

FIGURE 4.14: Measurement of the reference voltage for several MAC computations.

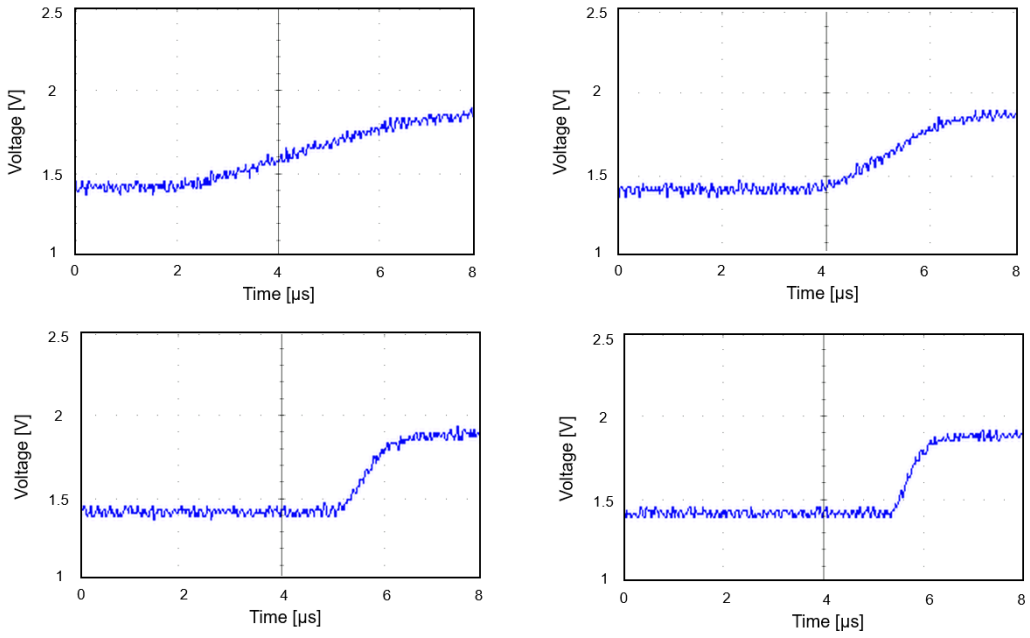

FIGURE 4.15: Measurement of the ramp signal with different levels of reference conductance g_{REF}.

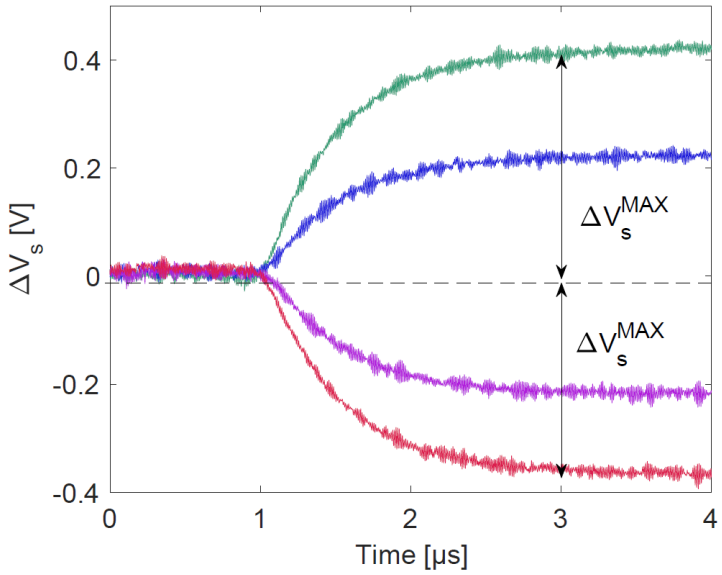

FIGURE 4.16: Waveforms showing four different MAC operations. The results here shown are interpolated to rule out the effects of the level shifter.

4.4 Characterization results

The AIMC performances have been evaluated in terms of MAC accuracy. The result of the generic operation was obtained measuring on DMA pin the output voltage ΔV_S, as defined in 4.5. Data were digitally converted using a 16-bit ADC available on a dedicated evaluation board.

Measured data are then postprocessed to compensate the effects of the level shifter, exploiting an automated interpolation based on the input analog voltages and the 1.2-V supply, accordingly with Figure 4.12.

In the following Paragraphs, results are related to the normalized MAC output z, defined as:

$$z \doteq \frac{Z}{Z^{MAX}} = \frac{\mathbf{w} \cdot \mathbf{x}}{\max(\mathbf{w} \cdot \mathbf{x})} \tag{4.8}$$

which is obtained measuring $\frac{\Delta V_S}{\Delta V_S^{MAX}}$. Figure 4.16 represents instead waveforms associated to two positive and two negative MAC operations. It is possible to execute an arbitrary sequence of MAC operations exploiting the implemented GUI environment, which writes all the MAC outputs in a log file.

4.4.1 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 cells has been replaced with the conductances of the test unit,

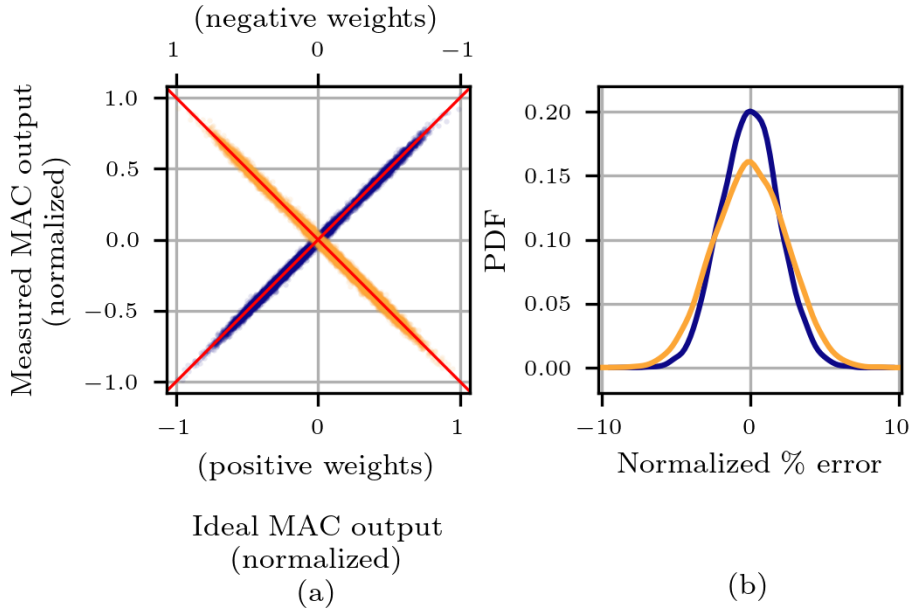

FIGURE 4.17: Accuracy of the AIMC unit. Left: Measured MAC operations as a function of the ideal MACs, where MAC coefficients are implemented with programmable integrated resistances. MAC weights w_i employed for the yellow (purple) data are negative (positive). Right: MAC error distributions of the two data sets.

presented in the Design for testability Section. Measurements performed on four different testchips are shown in Figure 4.17 (left), where the yellow curve refers to MAC 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 4.5 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 Figure 4.17 (right). The accuracy of the circuit, defined as $(1 - \sigma_\varepsilon)$ [15], where σ_ε 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.

4.4.2 Single conductance time drift compensation

The drift compensation technique on individual cells has been tested evaluating a set of normalized MAC outputs depending on a single weight w_i. To this purpose, 960 PCM cells, belonging to 80 different WLs, have been programmed, with the iterative algorithm developed in Chapter 2, with four different conductance levels. Then, in accordance with 4.5, all but the i-th input x_i have been set to 0, whereas x_i was chosen equal to the maximum x^{MAX}; then, the normalized MAC output $z = w_i/w^{MAX}$ was measured. This operation has been repeated for all the 960 cells after T1 = 1 day, T2 = 4 days, and T3 = 7 days from initial time T0. Then, the testchip has been baked at 85°C in a controlled climate chamber to accelerate cells drift phenomena

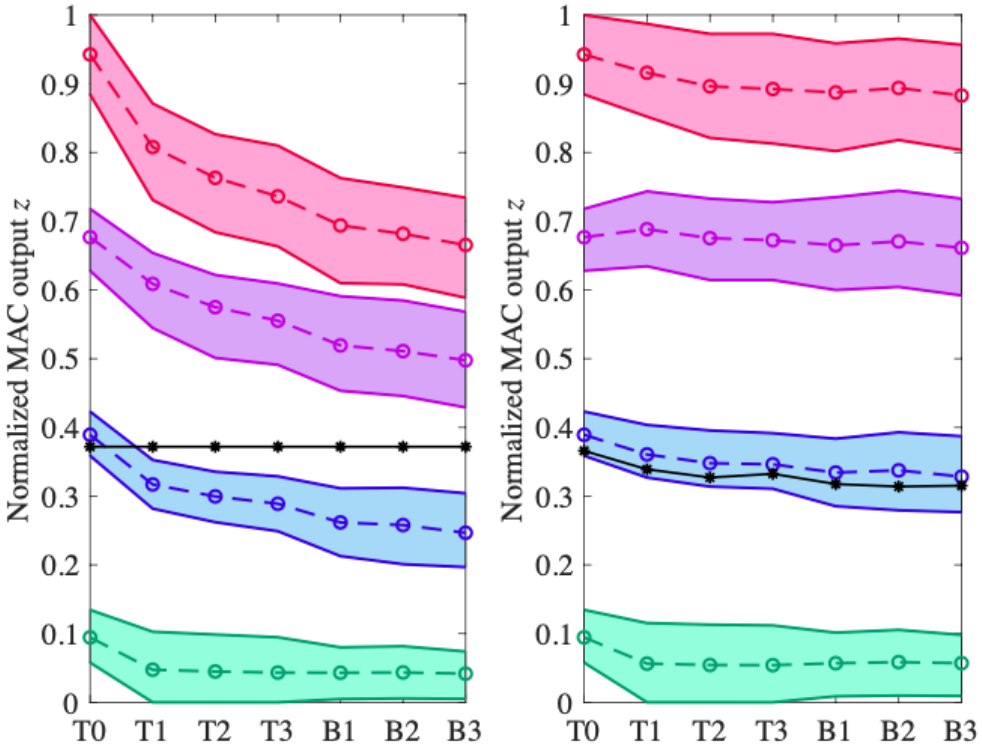

FIGURE 4.18: Measured normalized outputs z after programming (T0), T1 = 1 day, T2 = 4 days and T3 = 7 days at room temperature, and then after B1 = 1 hour, B2 = 5 hours and B3 = 24 hours bake at 85°C. Left: constant reference current (black line); right: PCM reference current (black line). Dashed lines identify the mean measured values, while areas borders identify the minimum and the maximum.

[88]. Measures have been repeated after B0 = 1 hour, B1 = 5 hours and B3 = 24 hours bake.

All measurements have been performed first with a ramp signal V_R generated with a test conductance g_{REF} constant in time, thus expecting no drift compensation (left plot of 4.18); then, V_R was generated by a PCM reference cell g_{REF} from the array (right plot of 4.18). As expected, in the first case, normalized outputs z tend to decrease in time under the effect of cells conductance drift, which becomes even stronger after the bake. On the contrary, in the second case, the compensation mechanism successfully reduces the drop of results in time, in agreement with 4.7, keeping the four considered output levels widely separated. Nonetheless, the spread of MAC operations belonging to the same level is unaffected by drift compensation, as it is related to programming precision and to PCM cell-to-cell drift variability. In this example, g_{REF} was programmed to the second conductance level. To quantify the effect of compensation, the normalized drift errors after 7 days at room temperature $\Delta z(\mathrm{T3}) \doteq \mathrm{z(T0)} - \mathrm{z(T3)}$, and after 24-hours bake $\Delta z(\mathrm{B3}) \doteq \mathrm{z(T0)} - \mathrm{z(B3)}$ have been calculated for each multiplication in both uncompensated and compensated case. Results of 4.19 show that the proposed technique keeps mean drift error under 6%, even after bake.

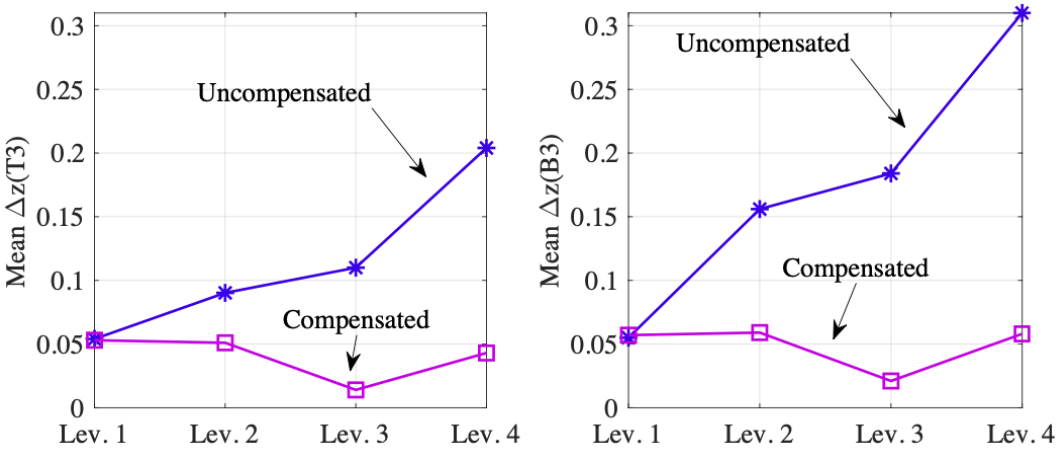

FIGURE 4.19: Normalized mean drift error Δz as a function of the PCM cells levels, in T3 (left) and B3 (right).

4.4.3 Reference cell choice and full MAC drift compensation

As previously shown, 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 ΔV_{OUT} can vary between $\pm\Delta V_{\text{OUT}}^{\text{MAX}}$, a limit determined by the design of the output integrator. The maximum output swing, $\Delta V_{\text{OUT}}^{\text{MAX}}$, enforces an upper bound on the maximum MAC operation, i.e., from (4.5):

$$\Delta V_{\text{OUT}}^{\text{MAX}} = \frac{k V_{\text{IN}}^{\text{MAX}}}{g_{\text{REF}}} \left[\max_j \left(\sum_{i=1}^{n} g_{j,i} \right) \right] = \frac{k V_{\text{IN}}^{\text{MAX}}}{g_{\text{REF}}} \left[n g^{\text{MAX}} \right] \tag{4.9}$$

Thus, one can obtain a minimum value for g_{REF}:

$$g_{\text{REF}} \geq k \frac{V_{\text{IN}}^{\text{MAX}}}{\Delta V_{\text{OUT}}^{\text{MAX}}} \left[n g^{\text{MAX}} \right] \tag{4.10}$$

where $V_{\text{IN}}^{\text{MAX}}$ is the analog value corresponding to the maximum input x^{MAX}. Condition (4.10) represents the worst-case constraint on g_{REF}, as it assumes the maximum programmable conductance g^{MAX} for each stored weight $g_{j,i}$ [31]. 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:

$$g_{\text{REF}} \geq k \frac{V_{\text{IN}}^{\text{MAX}}}{\Delta V_{\text{OUT}}^{\text{MAX}}} \left[\max_j \left(\sum_{i=1}^{n} g_{j,i} \right) \right] = g_{\text{REF}}^{\text{min}} \tag{4.11}$$

If the inequality in (4.11) is satisfied, all possible MAC operations are mapped within the available output swing (as shown in the black line in Figure 4.20); otherwise, the output voltage may saturate (as represented by the purple line). Thanks

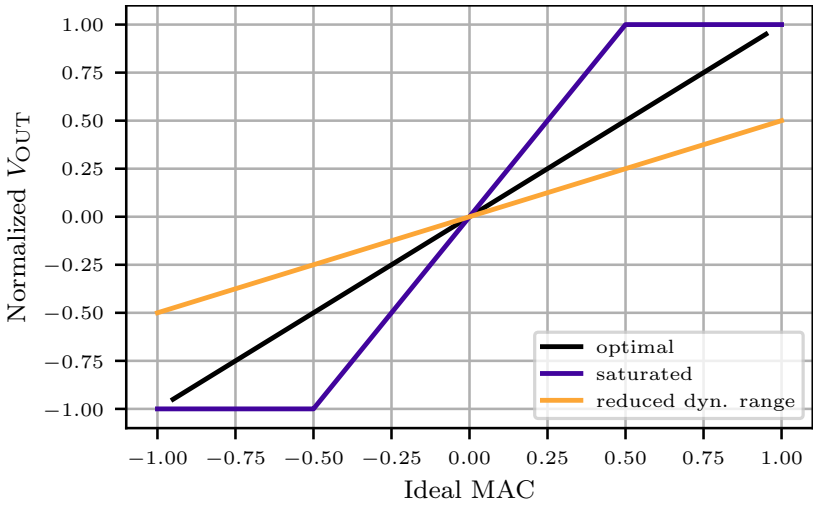

FIGURE 4.20: Effects of different values for the reference conductance g_{REF} on the output swing of the MACs. *i*) optimal swing (Equation (6)); *ii*) saturation due to low g_{REF} level; *iii*) swing reduction induced by PCM cells drift with constant g_{REF}.

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 Figure 4.20, 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 [31], [53], [89], both effects leading to an imperfect compensation of the drift exponents in (4.7). 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 4.4.1). ΔV_{OUT} has been computed according to the model (4.5) at time t_0 with the target values of cells programming (i.e., without drift), obtaining the target MAC values $\hat{\mathbf{z}}$; then, the effects of drift have been simulated in $\mathbf{z}(t)$ with (4.7), where the drift coefficient values have been taken from a previous work [53]. Figure 4.21 depicts the MAC accuracy, already defined as $(1-\sigma_\varepsilon)$, as a function of the $\frac{g_{REF}}{g_{MAX}}$ 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 [65], letting thus MAC operations depend on drift. It is evident that the more effective interval of values for g_{REF} is between $\sim [0.4-0.6]\, g^{MAX}$. The accuracy gain with respect to the uncompensated case in the three considered scenarios, when drift compensation is implemented with $\frac{g_{REF}}{g^{MAX}} = 0.5$, is 4.79, 5.38 and 7.81%.

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

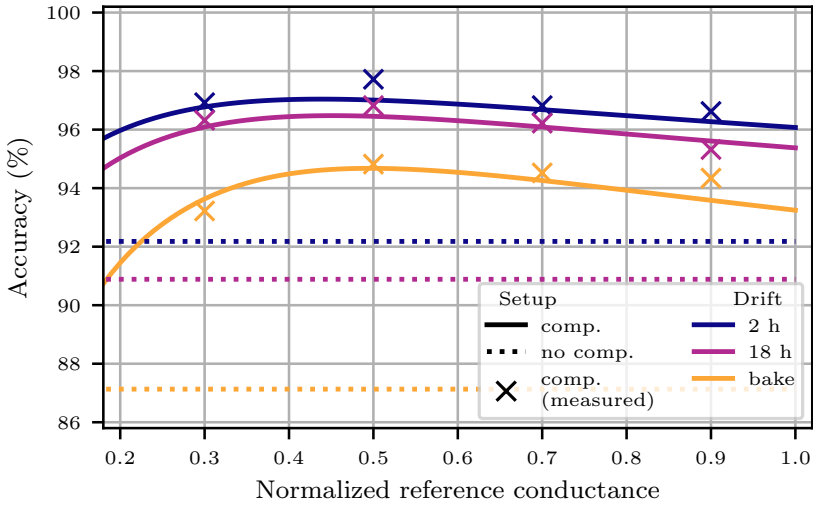

FIGURE 4.21: MAC accuracy as a function of $\frac{g_{REF}}{g^{MAX}}$. Continuous lines refer to simulated results after 2- and 18-hours room temperature and 90°C bake drift. Dashed lines represent the MAC accuracy in the same conditions when no compensation is adopted. Crosses report the results of experimental evaluation of MAC accuracy for different g_{REF} levels.

to 0.5. The full set of MAC operations with $\frac{g_{REF}}{g^{MAX}} = 0.5$ is reported in Figure 4.22, where the 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 24-hours 90°C bake (94.8%); otherwise, its standard deviation $\sigma(\boldsymbol{\varepsilon})$ 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.

4.5 Power analysis

A theoretical power analysis of the analog core has been performed considering the simulated current consumptions of each block, which are reported in Table 4.4.

The following expression summarizes the total mean current consumption I_{TOT} of the analog core (i.e., not considering the digital unit, the D/A converters and the DMA buffer):

$$I_{TOT} = k_1 I_{CELL} + I_{VR} + I_{RAMP} + \frac{k_2 n}{2} I_{CELL} + n I_{CMP} + I_{OUT} + k_3 n I_{CELL} \qquad (4.12)$$

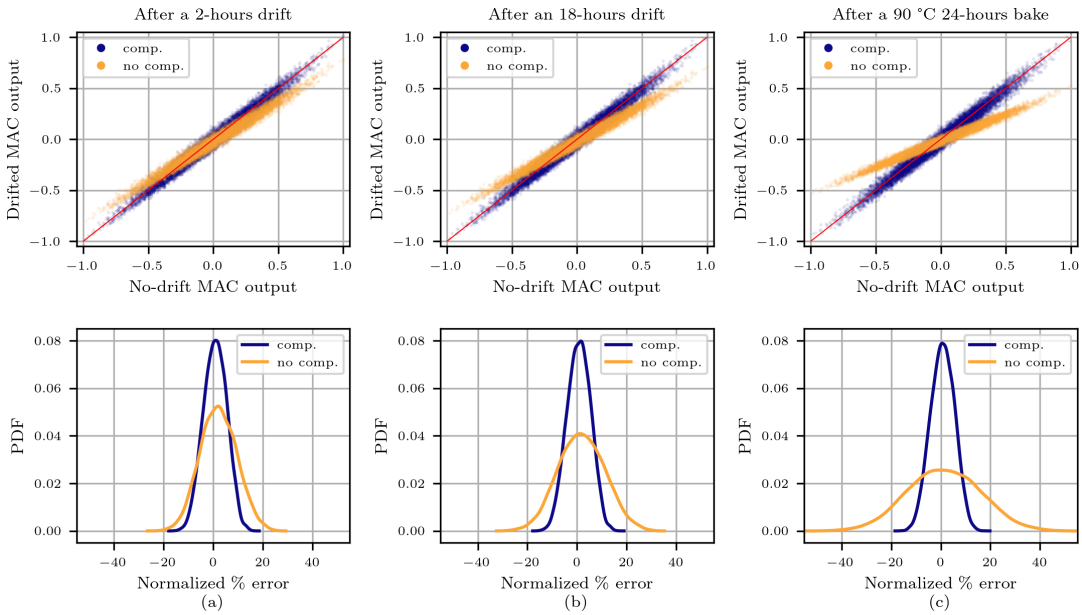

FIGURE 4.22: (Top) Comparison of the MAC output for compensated and uncompensated cells (a) after 2 hours from programming, (b) after 18 hours and (c) after a 24-hours bake at 90 °C. (Bottom) Distribution of the MAC error ε.

where $k_1 = 1.05, k_2 = 1.15$ and $k_3 = 1.2$ are dimensionless coefficients determined by the current mirrors ratio employed in the readout circuits, and $n = 12$ is the size of the MAC operation. Moreover, it has been assumed the inputs being at mean level ($V_i = V^{MAX}/2 \; \forall i$), so that the cell currents are halved. The total current I_{TOT} is then equal to 288 μA.

In accordance with the reference works, the energy efficiency η of the analog core is quantified in terms of Number of Operations per Watt (OPS/W):

$$\eta = \frac{N_{OP}^{MAC}}{T_{MAC} V_{DD} I_{TOT}} \tag{4.13}$$

where $N_{OP}^{MAC} = 23$ is the total number of elementary operations performed for a MAC (i.e., 12 products and 11 sums), whose mean execution time is $T_{MAC} = 200$ ns. Therefore, the energy efficiency turns to be $\eta \simeq 191$ GOPS/W. This result lays several orders of magnitude under the state of the art [46], [90]–[92], as the current testchip has not been optimized in terms of power consumption. Nonetheless,

TABLE 4.4: Current consumption of the considered analog blocks.

Analog block	Current name	Mean current consumption
Single PCM cell	I_{CELL}	5 μA
Ramp integrator	I_{RAMP}	15 μA
Comparator	I_{CMP}	7 μA
Voltage regulator	I_{VR}	1 μA
Output integrator	I_{OUT}	75 μA

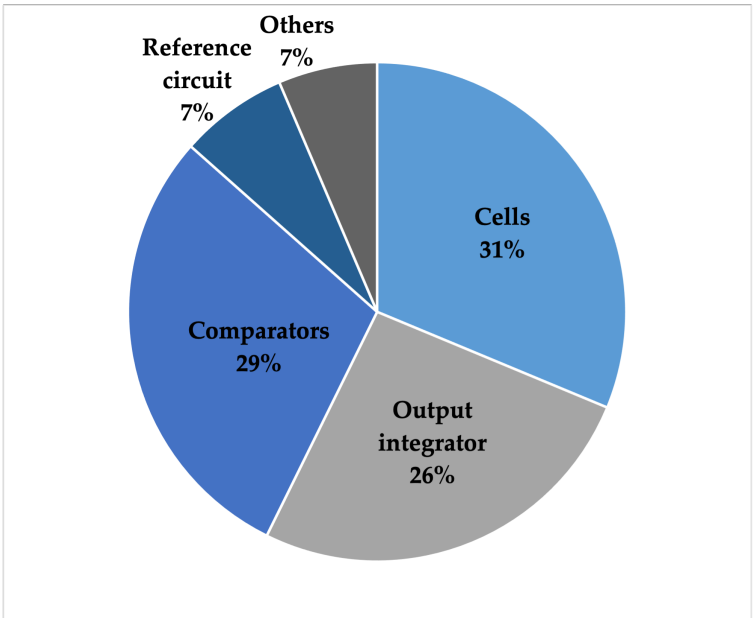

FIGURE 4.23: Power consumption diagram of the analog core.

some considerations can be done is sight of future developements and optimizations. In Figure 4.23 the power consumption diagram is reported. It is clear that the most power-hungry contribution comes from PCM cells. This can be addressed with either a lower readout voltage V_{REF} level, and employing lower conductances for MAC weights, though with a consequent increase of noise and drift figures, according to Chapter 2. 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 [91], [93]). Finally, the high output integrator consumption is due to the large amount of cells currents to be integrated on the output capacitance C_S. Alternative solutions to the output integrator, as for example a current-to-digital conversion [46] must be taken into account, also considering some related observations of the next Paragraph.

4.6 Challenges and perspectives

The current testchip represents a first prototype to demonstrate the possibility to circuitally compensate the drift of PCM cells. The choice of integrating the cells current to generate the analog MAC result, requires a power-hungry output integrator, as shown in the previous Section. Furthermore, the testchip validation showed that the integration mechanism is affected by the leakage currents of the analog core, whose contribution affects the computation even once the MAC computation has ended. Figures 4.24 and 4.25 show two MAC operations, where the time scale is intentionally set to show the evolution of the output voltage after the end of the operations themselves. It is evident that the output voltage keeps growing, even after

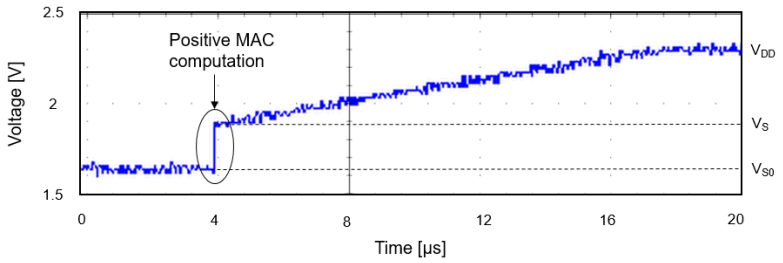

FIGURE 4.24: Saturation of the output voltage after a positive MAC operation.

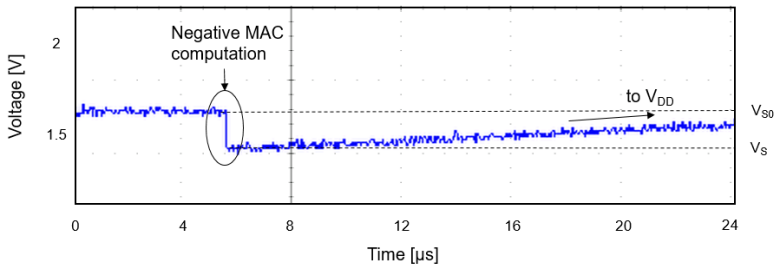

FIGURE 4.25: Saturation of the output voltage after a negative MAC operation.

the computation has ended, till it reaches the 1.2-V power supply. The phenomenon manifests itself both with a positive and a negative MAC output. This is another important issue of employing the output current integrator. By analyzing the speed of the output saturation, it is possible to estimate the leakage current being integrated and it is equal to approximately 340 pA, which is confirmed by circuital simulations. Consequently, this condition is particularly critical as the output must be precisely sampled, otherwise it will be affected by an addition term. This consideration underlines again the necessity to differently implement the output generation, as different schemes are available [46]. Furthermore, this output conversion chain has been designed to handle MAC operations with a limited size ($n = 12$). However, the size of the operations is expected to be larger for actual applications, as implemented in state-of-the-art works [90], [91], [93], where this scheme is not suitable.

Moreover, the MAC computation time T_{MAC}, as previously shown, depends inversely on the value of reference cell conductance g_{REF}, which is time-varying to implement drift compensation. As g_{REF} decreases in time under the effect of drift, T_{MAC} tends to increase, and therefore, the number of OPS/W tends to shrink accordingly. Different approaches to implement a similar circuital drift compensation must be then considered.

Additionally, in this first implementation, only a single PCM reference cell has been involved in the ramp generation. As the drifting behavior of cells is purely random, with a partial correlation to their programmed conductance level (as stated in Chapter 2), the reference current may be generated averaging on a set of different cells. This feature has been not yet implemented in this prototype due to layout

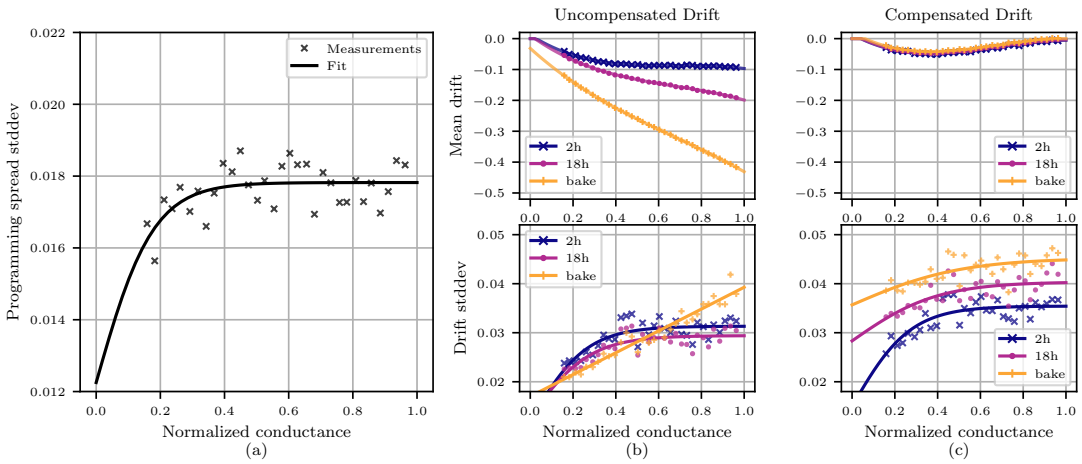

FIGURE 4.26: (a) Standard deviation of the spread resulting from the iterative programming procedure, as a function of the average conductance of each programmed level. (b) Mean and standard deviation of the drift-induced conductance variation for cells without compensation and (c) with compensation.

constraints, but should be considered in possible next developments to better characterize the proposed drift compensation technique.

4.7 Application in a Deep Neural Network scenario

4.7.1 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 4.5, then reads:

$$V_{\mathrm{OUT}_j} = \frac{g_{j,k}}{g_{\mathrm{REF}}} V_k \tag{4.14}$$

and depends on the single cell $g_{j,k}$ behavior only. The only nonzero input, V_k, was forced to its maximum value $V_{\mathrm{IN}}^{\mathrm{MAX}}$ 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 is depicted in Figure 4.26 against the mean normalized conductance. A continuous model has been fit to the data, using the equation

$$\sigma_p(g) = \sigma_0 + \sigma_1 \tanh\left(\frac{g}{\gamma_0}\right) \tag{4.15}$$

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 Figure 4.17.

Conductance drift has been observed, both for compensated and uncompensated cells, in the same settings described in the previous Section, 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 Figures 4.26 (b) and 4.26 (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 (4.15). 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 Figure 4.26.

The curve describing the standard deviation of an uncompensated drift in Figure 4.26 after the 24-hours bake results in a 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 Figure 4.27. In that region the 2-hours and 18-hours setups experience the a growth trend in standard deviation versus conductance. Hence, when the drift is mapped back to the original conductance value, right after the cells have been programmed, the trend is expanded and fills the entire horizontal axis. Additionally, as larger values of initial conductance lead to a more pronounced average drift, the larger mean variation determines an increase in the spread as well, with the yellow curve overcoming the behavior of the other setups on the upper part of the domain. 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.

4.7.2 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 [94]. Two popular neural networks have been used, the Lenet-5 [95] and the VGG-8 [47], having significantly different complexities, with $\sim 8 \times 10^5$

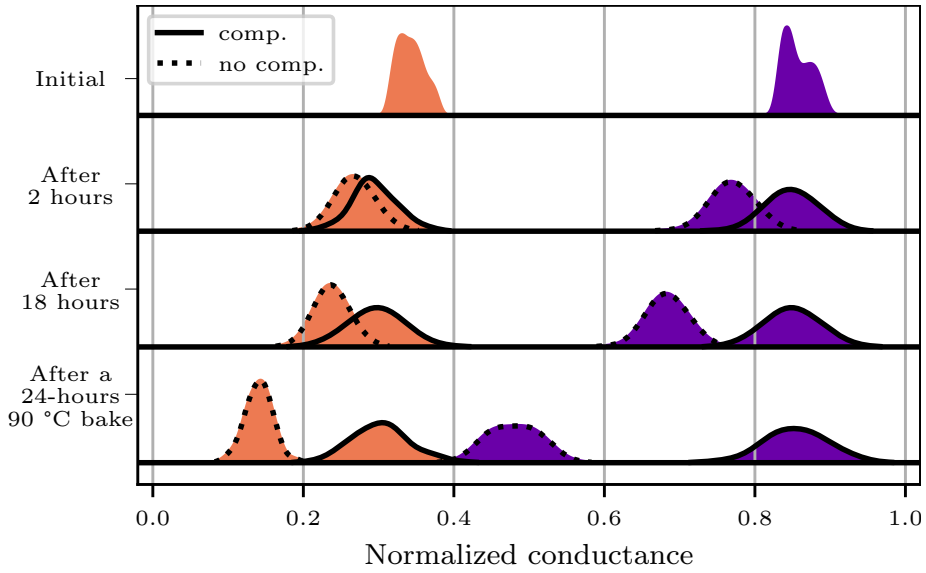

FIGURE 4.27: Evolution over time of two batches of PCM devices, programmed to a target normalized conductance of 0.35 and 0.85. Both compensated and uncompensated cells are shown.

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:

$$h_j = f\left(b_j + \sum_i k \frac{g_{j,i}}{g_{\text{REF}}} V_i\right), \tag{4.16}$$

where equation (4.5) 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

$$g_{j,i} = g = g_0 + \Delta g_p(g_0) + \Delta g_d(g_0, \Delta t), \tag{4.17}$$

where $\Delta g_p(g_0)$ is the programming 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 $\mathcal{N}\left(\mu_d(g_0, \Delta t), \sigma_d(g_0, \Delta t)\right)$ distribution, using the models depicted in Figure 4.26

Both neural networks have been trained with the Adam optimizer [96], 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.

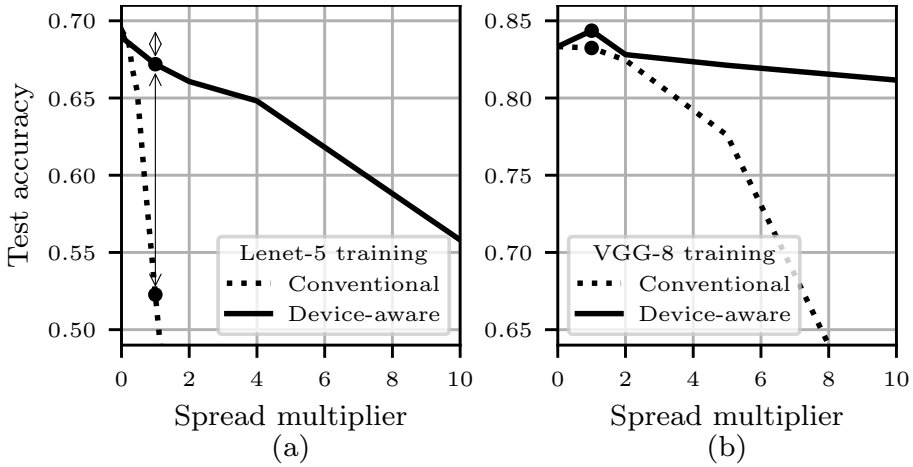

FIGURE 4.28: Accuracy of the trained networks versus the programming spread scaling coefficient, both for the conventional and device-aware trainings. (a) Lenet-5 and (b) VGG-8 DNNs.

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 δ_g of the programming algorithm described in Chapter 2, 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 Figure 4.28 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. The datapoint corresponding to a spread multiplier 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 [97] has been employed. 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 [60]. 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 (4.17). Results in Figure 4.28 refers 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

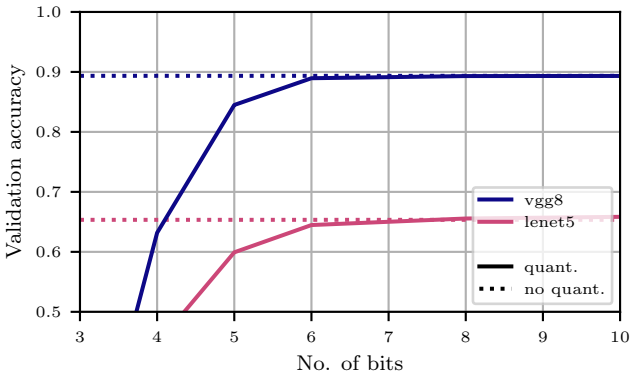

FIGURE 4.29: Classification accuracy when quantizing the signals applied to and read from every layer, for NNs trained to exclusively address PCM programming spread using a multiplier of 1.

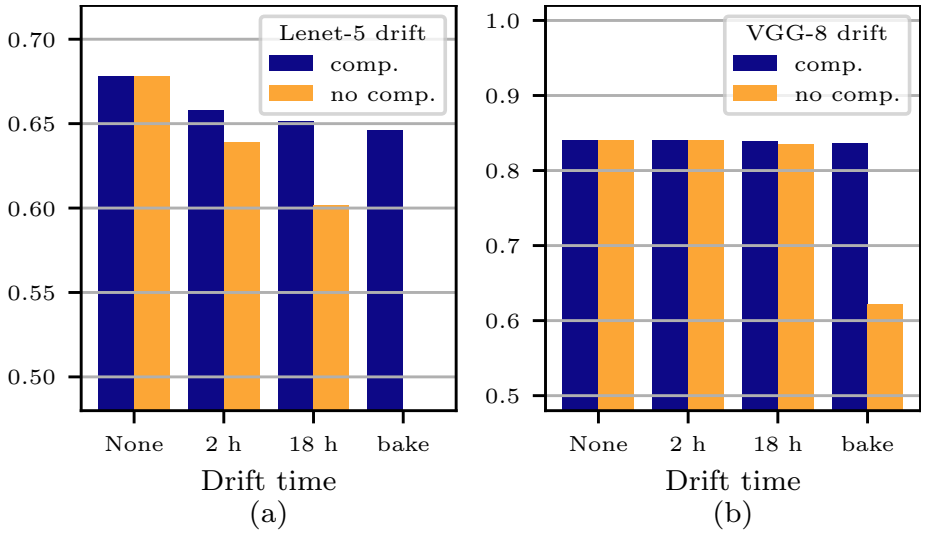

FIGURE 4.30: Accuracy achieved when drift is applied to the DNN weights at inference time, both with and without compensation. (a) Lenet-5 and (b) VGG-8 DNNs.

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 [98], highlights the value of the device-aware training technique to construct small and robust DNNs.

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 [51], [99]. As an example, Fig. 4.29 shows the classification accuracy of the two device-aware trained networks 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 [100]. 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 [99].

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, have been re-evaluated by introducing the drift component of the conductance Δg_d at inference time. From Figure 4.30 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 hardware compensation technique.

4.8 Conclusion

In this Chapter a peripheral unit adding analog in-memory multiply and accumulate (MAC) computing function to an embedded phase-change memory (ePCM) macrocell has been presented. The unit exploits an innovative readout scheme to address non-linearity of I-V characteristic and time drift of cells conductances. The unit is conceived to operate with signed inputs and coefficients and does not require any modification to the internal structure of the ePCM. MAC operations are performed with a 1-σ accuracy of 95.56which is not significantly affected in time by drift effects, even after 24-hours bake at 85°C. The challenges and the possible future developments of this solution are discussed as well.

To evaluate the employment of the proposed hardware in a Deep Neural Network (DNN) scenario, the spread and retention of the programmed conductances have been characterized and modeled, including the effects of the proposed 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.

Chapter 5

Conclusions

In this chapter the results exposed so far will be summarized and discussed. As already stated, a thorough characterization of PCM cells has been first carried out to investigate their possible employment as enabling device for AIMC. Drift, dispersion and noise have been then analyzed in relation to memory elements programmed with a dedicated programming algorithm, showing their dependencies on conductance targets and temperature. These results have been then employed to evaluate PCM devices in specific applications. First, a possible use in a deep neural network (DNN) scenario has shown a way of including arbitrary synapse models within a neural layer. The injection of noise on the trained weights has highlighted the robustness of the networks to a point that makes the devices promising candidates in circuital implementations. Moreover, challenges related to time-dependent non-idealities of PCM devices have been investigated in a structural health monitoring (SHM) context. Structural parameters have been identified in two environments, showing that PCM devices do not necessarily need to be freshly programmed for this application. All these experimental activities have contributed to the design and the development of a peripheral unit adding analog in-memory multiply and accumulate (MAC) computing function to an embedded phase-change memory (ePCM) macrocell. The unit exploits an innovative readout scheme to address non-linearity of cells I-V characteristic and time drift of cells conductances. MAC operations are performed with a $1-\sigma$ accuracy of 95.56%, which is not significantly affected in time by drift effects. Drift compensation has also been tested in DNN classification tasks, recovering up to 36% in accuracy compared to the uncompensated case.

This research field embraces a wide spectrum of topics, thus requiring deep investigations on different subjects, from devices to circuits and to applications. Many challenges are still open and more investigations in several fields are needed. Device physics is currently being studied, as well as IMC architectural design and heterogeneous systems.

This thesis represents a first step to contribute to this topic, and establishes an inspiring basis for next developments, providing evidence that the intersection between different disciplines and the collaboration among the community are fundamental to the advancement of research and technology.

Bibliography

[1] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020, ISSN: 17483395. DOI: 10.1038/s41565-020-0655-z.

[2] M. Di Ventra and Y. V. Pershin, "The parallel approach," *Nature Physics*, vol. 9, no. 4, pp. 200–202, 2013, ISSN: 17452481. DOI: 10.1038/nphys2566.

[3] S. Quqa, A. Antolini, E. F. Scarselli, *et al.*, "Phase change memories in smart sensing solutions for structural health monitoring," *Journal of Computing in Civil Engineering*, vol. 36, no. 4, p. 04 022 013, 2022. DOI: 10.1061/(ASCE)CP.1943-5487.0001027.

[4] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: Analog computing," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 108–122, 2018, ISSN: 15582256. DOI: 10.1109/JPROC.2018.2871057.

[5] W. Haensch, T. Gokmen, and R. Puri, "The Next Generation of Deep Learning Hardware: Analog Computing," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 108–122, 2019, ISSN: 15582256. DOI: 10.1109/JPROC.2018.2871057.

[6] J. Pronold, J. Jordan, B. Wylie, I. Kitayama, M. Diesmann, and S. Kunkel, "Routing brain traffic through the von neumann bottleneck: Efficient cache usage in spiking neural network simulation code on general purpose computers," *Parallel Computing*, vol. 113, p. 102 952, 2022, ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2022.102952. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167819122000461.

[7] A. Kneip and D. Bol, "Impact of analog non-idealities on the design space of 6t-sram current-domain dot-product operators for in-memory computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 5, pp. 1931–1944, 2021. DOI: 10.1109/TCSI.2021.3058510.

[8] V. Seshadri, D. Lee, T. Mullins, *et al.*, "Ambit: In-memory accelerator for bulk bitwise operations using commodity dram technology," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 273–287.

[9] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 288–301.

[10] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-mb in-memory-computing cnn accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019. DOI: 10.1109/JSSC.2019.2899730.

[11] A. Biswas and A. P. Chandrakasan, "Conv-sram: An energy-efficient sram with in-memory dot-product computation for low-power convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2019. DOI: 10.1109/JSSC.2018.2880918.

[12] N. Verma, H. Jia, H. Valavi, *et al.*, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43–55, 2019. DOI: 10.1109/MSSC.2019.2922889.

[13] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 449–464, 2008, ISSN: 00188646. DOI: 10.1147/rd.524.0449.

[14] T. Nirschl, J. B. Philipp, T. D. Happ, *et al.*, "Write strategies for 2 and 4-bit multi-level phase-change memory," *Technical Digest - International Electron Devices Meeting, IEDM*, pp. 461–464, 2007, ISSN: 01631918. DOI: 10.1109/IEDM.2007.4418973.

[15] M. Le Gallo, A. Sebastian, R. Mathis, *et al.*, "Mixed-precision in-memory computing," *Nature Electronics*, vol. 1, no. 4, pp. 246–253, 2018, ISSN: 25201131. DOI: 10.1038/s41928-018-0054-8. arXiv: 1701.04279.

[16] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed sensing with approximate message passing using in-memory computing," *IEEE Transactions on Electron Devices*, vol. 65, no. 10, pp. 4304–4312, 2018. DOI: 10.1109/TED.2018.2865352.

[17] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[18] D. L. Donoho, "Compressed sensing," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1289–1306, 2006.

[19] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 72–82, 2008.

[20] C. Paolino, A. Antolini, F. Pareschi, *et al.*, "Compressed sensing by phase change memories: Coping with encoder non-linearities," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, vol. 2021-May, 2021, pp. 1–5, ISBN: 9781728192017. DOI: 10.1109/ISCAS51556.2021.9401176.

[21] V. Joshi, M. Le Gallo, S. Haefeli, *et al.*, "Accurate deep neural network inference using computational phase-change memory," en, *Nature Communications*, vol. 11, no. 1, p. 2473, May 2020, ISSN: 2041-1723. DOI: 10.1038/s41467-020-16108-9. [Online]. Available: https://www.nature.com/articles/s41467-020-16108-9 (visited on 10/22/2021).

[22] S. Bianchi, I. Muñoz-Martín, S. Hashemkhani, G. Pedretti, and D. Ielmini, "A bio-inspired recurrent neural network with self-adaptive neurons and pcm synapses for solving reinforcement learning tasks," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9181103.

[23] M. Baldo, E. Petroni, L. Laurin, *et al.*, "Interaction between forming pulse and integration process flow in epcm," in *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, 2022, pp. 145–148. DOI: 10.1109/PRIME55000.2022.9816795.

[24] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM Journal of Research and Development*, vol. 52, no. 4-5, pp. 449–464, 2008, ISSN: 00188646. DOI: 10.1147/rd.524.0449.

[25] G. W. Burr, R. M. Shelby, C. Di Nolfo, *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element," in *Technical Digest - International Electron Devices Meeting, IEDM*, vol. 2015-February, IEEE, 2015, pp. 29.5.1–29.5.4, ISBN: 9781479980017. DOI: 10.1109/IEDM.2014.7047135. [Online]. Available: http://ieeexplore.ieee.org/document/7047135/.

[26] A. Sebastian, M. Le Gallo, G. W. Burr, S. Kim, M. Brightsky, and E. Eleftheriou, "Tutorial: Brain-inspired computing using phase-change memory devices," *Journal of Applied Physics*, vol. 124, no. 11, 2018, ISSN: 10897550. DOI: 10.1063/1.5042413.

[27] F. Bedeschi, R. Fackenthal, C. Resta, *et al.*, "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, 2008, ISSN: 00189200. DOI: 10.1109/JSSC.2008.2006439.

[28] F. Bedeschi, R. Fackenthal, C. Resta, *et al.*, "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 217–227, 2009, ISSN: 00189200. DOI: 10.1109/JSSC.2008.2006439.

[29] M. Pasotti, R. Zurla, M. Carissimi, *et al.*, "A 32-kb epcm for real-time data processing in automotive and smart power applications," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2114–2125, 2018. DOI: 10.1109/JSSC.2018.2828805.

[30] N. Papandreou, H. Pozidis, A. Pantazi, *et al.*, "Programming algorithms for multilevel phase-change memory," *Proceedings - IEEE International Symposium on Circuits and Systems*, pp. 329–332, 2011, ISSN: 02714310. DOI: 10.1109/ISCAS.2011.5937569.

[31] A. Antolini, E. Franchi Scarselli, A. Gnudi, *et al.*, "Characterization and programming algorithm of phase change memory cells for analog in-memory computing," *MATERIALS*, 2021. DOI: 10.3390/ma14071624.

[32] R. Jeyasingh, J. A. Chroboczek, G. Ghibaudo, M. Mouis, and H. S. Wong, "Low frequency noise in phase change materials," *Proceedings of the IEEE 21st International Conference on Noise and Fluctuations, ICNF 2011*, pp. 476–479, 2011. DOI: 10.1109/ICNF.2011.5994373.

[33] D. Ielmini, A. L. Lacaita, and D. Mantegazza, "Recovery and drift dynamics of resistance and threshold voltages in phase-change memories," *IEEE Transactions on Electron Devices*, vol. 54, no. 2, pp. 308–315, 2007, ISSN: 00189383. DOI: 10.1109/TED.2006.888752.

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

[35] D. Ielmini and G. Pedretti, "Device and circuit architectures for in-memory computing," *Advanced Intelligent Systems*, vol. 2, no. 7, p. 2 000 040, 2020, ISSN: 2640-4567. DOI: 10.1002/aisy.202000040.

[36] 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 *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, IEEE, 2019, pp. 418–421, ISBN: 9781728109961. DOI: 10.1109/ICECS46596.2019.8964986.

[37] D. Ielmini, D. Sharma, S. Lavizzari, and A. L. Lacaita, "Physical mechanism and temperature acceleration of relaxation effects in phase-change memory cells," in *2008 IEEE International Reliability Physics Symposium*, 2008, pp. 597–603. DOI: 10.1109/RELPHY.2008.4558952.

[38] D. Ielmini and Y. Zhang, "Analytical model for subthreshold conduction and threshold switching in chalcogenide-based memory devices," *Journal of Applied Physics*, vol. 102, no. 5, p. 054 517, 2007. DOI: 10.1063/1.2773688. eprint: https://doi.org/10.1063/1.2773688. [Online]. Available: https://doi.org/10.1063/1.2773688.

[39] M. Boniardi and D. Ielmini, "Physical origin of the resistance drift exponent in amorphous phase change materials," *Applied Physics Letters*, vol. 98, no. 24, p. 243 506, 2011. DOI: 10.1063/1.3599559. eprint: https://doi.org/10.1063/1.3599559. [Online]. Available: https://doi.org/10.1063/1.3599559.

[40] N. Ciocchini, E. Palumbo, M. Borghi, P. Zuliani, R. Annunziata, and D. Ielmini, "Modeling resistance instabilities of set and reset states in phase change memory with ge-rich gesbte," *IEEE Transactions on Electron Devices*, vol. 61, no. 6, pp. 2136–2144, 2014. DOI: 10.1109/TED.2014.2313889.

[41] M. Le Gallo and A. Sebastian, "An overview of phase-change memory device physics," *Journal of Physics D: Applied Physics*, 2020. DOI: 10.1088/1361-6463/ab7794.

[42] S. Ghazi Sarwat, T. M. Philip, C.-T. Chen, *et al.*, "Projected mushroom type phase-change memory," *Advanced Functional Materials*, vol. 31, no. 49, p. 2 106 547, 2021. DOI: https://doi.org/10.1002/adfm.202106547. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/adfm.202106547. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/adfm.202106547.

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

[44] 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 Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4782–4790, 2018. DOI: 10.1109/TNNLS.2017.2778940.

[45] P. Yao, H. Wu, B. Gao, *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020, ISSN: 1476-4687. DOI: 10.1038/s41586-020-1942-4. [Online]. Available: https://doi.org/10.1038/s41586-020-1942-4.

[46] R. Khaddam-Aljameh, M. Stanisavljevic, J. Fornt Mas, *et al.*, "Hermes-core—a 1.59-tops/mm2 pcm on 14-nm cmos in-memory compute core using 300-ps/lsb linearized cco-based adcs," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 4, pp. 1027–1038, 2022. DOI: 10.1109/JSSC.2022.3140414.

[47] X. Sun, W. S. Khwa, Y. S. Chen, *et al.*, "Pcm-based analog compute-in-memory: Impact of device non-idealities on inference accuracy," *IEEE Transactions on Electron Devices*, vol. 68, no. 11, pp. 5585–5591, 2021. DOI: 10.1109/TED.2021.3113300.

[48] I. Muñoz-Martín, 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 Transactions on Electron Devices*, vol. 68, no. 12, pp. 6076–6081, Dec. 2021, ISSN: 1557-9646. DOI: 10.1109/TED.2021.3118996.

[49] S. Kariyappa, H. Tsai, K. Spoon, *et al.*, "Noise-Resilient DNN: Tolerating Noise in PCM-Based AI Accelerators via Noise-Aware Training," *IEEE Transactions on Electron Devices*, vol. 68, no. 9, pp. 4356–4362, Sep. 2021, ISSN: 1557-9646. DOI: 10.1109/TED.2021.3089987.

[50] A. Chen, S. Ambrogio, P. Narayanan, *et al.*, "Enabling High-Performance DNN Inference Accelerators Using Non-Volatile Analog Memory (Invited)," in *2020 4th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*, Penang, Malaysia: IEEE, Apr. 2020, pp. 1–4, ISBN: 9781728125398. DOI: 10.1109/EDTM47692.2020.9117896. [Online]. Available: https://ieeexplore.ieee.org/document/9117896/ (visited on 12/12/2022).

[51] V. Joshi, M. Le Gallo, and S. Haefeli, "Accurate deep neural network inference using computational phase-change memory," *Nat Commun 11, 2473*, 2020. DOI: 10.1038/s41467-020-16108-9.

[52] R. Bruce, S. G. Sarwat, I. Boybat, *et al.*, "Mushroom-Type phase change memory with projection liner: An array-level demonstration of conductance drift and noise mitigation," in *2021 IEEE International Reliability Physics Symposium (IRPS)*, Monterey, CA, USA: IEEE, Mar. 2021, pp. 1–6, ISBN: 9781728168937. DOI: 10.1109/IRPS46558.2021.9405191. [Online]. Available: https://ieeexplore.ieee.org/document/9405191/ (visited on 12/12/2022).

[53] 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 *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, 2022, pp. 233–236. DOI: 10.1109/PRIME55000.2022.9816788.

[54] M. Carissimi, R. Mukherjee, V. Tyagi, *et al.*, "2-Mb Embedded Phase Change Memory with 16-ns Read Access Time and 5-Mb/s Write Throughput in 90-nm BCD Technology for Automotive Applications," *ESSCIRC 2019 - IEEE 45th European Solid State Circuits Conference*, pp. 135–138, 2019. DOI: 10.1109/ESSCIRC.2019.8902656.

[55] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," *Nature Nanotechnology*, vol. 11, no. 8, pp. 693–699, 2016, ISSN: 1748-3387. DOI: 10.1038/nnano.2016.70. [Online]. Available: http://www.nature.com/articles/nnano.2016.70.

[56] A. Cabrini, S. Braga, A. Manetto, and G. Torelli, "Voltage-driven multilevel programming in phase change memories," *Proceedings of the 2009 IEEE International Workshop on Memory Technology, Design, and Testing, MTDT 2009*, pp. 3–6, 2009. DOI: 10.1109/MTDT.2009.11.

[57] S. Braga, A. Sanasi, A. Cabrini, and G. Torelli, "Voltage-driven partial-RESET multilevel programming in phase-change memories," *IEEE Transactions on*

Electron Devices, vol. 57, no. 10, pp. 2556–2563, 2010, ISSN: 00189383. DOI: 10.1109/TED.2010.2062185.

[58] Y. Zhang, J. Feng, Y. Zhang, *et al.*, "Multi-bit storage in reset process of phase-change Random Access Memory (PRAM)," *Physica Status Solidi - Rapid Research Letters*, vol. 1, no. 1, 2007, ISSN: 18626270. DOI: 10.1002/pssr.200600020.

[59] 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," *2019 26th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2019*, pp. 418–421, 2019. DOI: 10.1109/ICECS46596.2019.8964986.

[60] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," en, *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, ISSN: 1476-4687. DOI: 10.1038/323533a0. [Online]. Available: https://www.nature.com/articles/323533a0 (visited on 10/22/2021).

[61] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv:1708.07747 [cs, stat]*, Sep. 2017. [Online]. Available: http://arxiv.org/abs/1708.07747 (visited on 10/22/2021).

[62] G. F. Close, U. Frey, J. Morrish, *et al.*, "A 256-Mcell Phase-Change Memory Chip Operating at 2+ Bit/Cell," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 6, pp. 1521–1533, Jun. 2013, ISSN: 1558-0806. DOI: 10.1109/TCSI.2012.2220459.

[63] A. Antolini, E. Franchi Scarselli, A. Gnudi, *et al.*, "Characterization and Programming Algorithm of Phase Change Memory Cells for Analog In-Memory Computing," en, *Materials*, vol. 14, no. 7, p. 1624, Mar. 2021, ISSN: 1996-1944. DOI: 10.3390/ma14071624. [Online]. Available: https://www.mdpi.com/1996-1944/14/7/1624 (visited on 10/22/2021).

[64] M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed Sensing With Approximate Message Passing Using In-Memory Computing," *IEEE Transactions on Electron Devices*, vol. 65, no. 10, pp. 4304–4312, Oct. 2018, ISSN: 0018-9383, 1557-9646. DOI: 10.1109/TED.2018.2865352. [Online]. Available: https://ieeexplore.ieee.org/document/8450603/ (visited on 10/22/2021).

[65] A. Antolini, A. Lico, E. Franchi Scarselli, *et al.*, "An embedded pcm peripheral unit adding analog mac in memory computing feature addressing non linearity and time drift compensation," in *2022 IEEE 48th European Solid State Circuit Research (ESSCIRC)*, 2022, pp. 1–4.

[66] V. Y. Tan, M. Johnson, and A. S. Willsky, "Necessary and sufficient conditions for high-dimensional salient feature subset recovery," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, IEEE, 2010, pp. 1388–1392.

[67] D. Zonta, B. Glisic, and S. Adriaenssens, "Value of information: impact of monitoring on decision-making," *Structural Control and Health Monitoring*, vol. 21, no. 7, pp. 1043–1056, 2014, ISSN: 15452255. DOI: 10.1002/stc.1631.

[68] L. Qu, P. S. Routh, and P. D. Anno, "Wavelet Reconstruction of Nonuniformly Sampled Signals," *Signal Processing Letters, IEEE*, vol. 16, no. 2, pp. 73–76, 2009.

[69] H. Jo, S.-H. Sim, T. Nagayama, and B. F. Spencer, "Development and Application of High-Sensitivity Wireless Smart Sensors for Decentralized Stochastic Modal Identification," *Journal of Engineering Mechanics*, vol. 138, no. 6, pp. 683–694, 2012, ISSN: 0733-9399. DOI: 10.1061/(ASCE)EM.1943-7889.0000352.

[70] A. Sabato, C. Niezrecki, and G. Fortino, "Wireless MEMS-Based Accelerometer Sensor Boards for Structural Vibration Monitoring: A Review," *IEEE Sensors Journal*, vol. 17, no. 2, pp. 226–235, 2017, ISSN: 1530-437X. DOI: 10.1109/JSEN.2016.2630008.

[71] J. Long and O. Büyüköztürk, "A power optimised and reprogrammable system for smart wireless vibration monitoring," *Structural Control and Health Monitoring*, vol. 27, no. 2, 2020, ISSN: 15452263. DOI: 10.1002/stc.2468.

[72] A. B. Noel, A. Abdaoui, T. Elfouly, M. H. Ahmed, A. Badawy, and M. S. Shehata, "Structural Health Monitoring Using Wireless Sensor Networks: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1403–1423, 2017, ISSN: 1553877X. DOI: 10.1109/COMST.2017.2691551.

[73] S. Quqa, L. Landi, and P. P. Diotallevi, "Instantaneous modal identification under varying structural characteristics: A decentralized algorithm," *Mechanical Systems and Signal Processing*, vol. 142, p. 106 750, 2020, ISSN: 10961216. DOI: 10.1016/j.ymssp.2020.106750.

[74] S. Quqa, L. Landi, and P. Paolo Diotallevi, "Modal assurance distribution of multivariate signals for modal identification of time-varying dynamic systems," *Mechanical Systems and Signal Processing*, vol. 148, p. 107 136, 2021, ISSN: 10961216. DOI: 10.1016/j.ymssp.2020.107136.

[75] S. G. Mallat, *A wavelet tour of signal processing*, Third edit. Academic Press, 2009, ISBN: 9780123743701. DOI: 10.1016/B978-0-12-374370-1.X0001-8.

[76] M. Vetterli and J. Kovačević, *Wavelets and Subband Coding*. Prentice-hall, 1995, ISBN: 0130970808.

[77] A. Aloisio, R. Alaggio, and M. Fragiacomo, "Dynamic identification and model updating of full-scale concrete box girders based on the experimental torsional response," *Construction and Building Materials*, vol. 264, p. 120 146, 2020, ISSN: 09500618. DOI: 10.1016/j.conbuildmat.2020.120146.

[78] A. Aloisio, R. Alaggio, and M. Fragiacomo, "Time-domain identification of the elastic modulus of simply supported box girders under moving loads: Method and full-scale validation," *Engineering Structures*, vol. 215, p. 110 619, 2020, ISSN: 18737323. DOI: 10.1016/j.engstruct.2020.110619.

[79] A. Aloisio, D. P. Pasca, R. Alaggio, and M. Fragiacomo, "Bayesian estimate of the elastic modulus of concrete box girders from dynamic identification: A statistical framework for the a24 motorway in italy," *Structure and infrastructure engineering*, vol. 17, no. 12, pp. 1626–1638, 2021, ISSN: 17448980. DOI: 10.1080/15732479.2020.1819343.

[80] A. Aloisio, R. Alaggio, and M. Fragiacomo, "Bending stiffness identification of simply supported girders using an instrumented vehicle: Full scale tests, sensitivity analysis, and discussion," *Journal of Bridge Engineering*, vol. 26, no. 1, p. 04 020 115, 2021, ISSN: 1084-0702. DOI: 10.1061/(asce)be.1943-5592.0001654.

[81] R. Brincker, L. Zhang, and P. Andersen, "Modal identification of output-only systems using frequency domain decomposition," *Smart materials and structures*, vol. 10, no. 3, p. 441, 2001, ISSN: 09641726. DOI: 10.1088/0964-1726/10/3/303.

[82] R. J. Allemang, "The modal assurance criterion - Twenty years of use and abuse," *Sound and Vibration*, vol. 37, no. 8, pp. 14–21, 2003, ISSN: 15410161.

[83] R. Brincker and C. Ventura, *Introduction to operational modal analysis*. John Wiley & Sons, 2015, pp. 1–360, ISBN: 9781118535141. DOI: 10.1002/9781118535141.

[84] C. Paolino, A. Antolini, F. Pareschi, *et al.*, "Compressed Sensing by Phase Change Memories: coping with encoder non-linearities," pp. 1–5, 2021. DOI: 10.1109/iscas51556.2021.9401176.

[85] X. Sun, W. S. Khwa, Y. S. Chen, *et al.*, "PCM-Based Analog Compute-In-Memory: Impact of Device Non-Idealities on Inference Accuracy," *IEEE Transactions on Electron Devices*, vol. 68, no. 11, pp. 5585–5591, 2021, ISSN: 15579646. DOI: 10.1109/TED.2021.3113300.

[86] R. Khaddam-Aljameh, M. Stanisavljevic, J. Fornt Mas, *et al.*, "HERMES Core-A 14nm CMOS and PCM-based In-Memory Compute Core using an array of 300ps/LSB Linearized CCO-based ADCs and local digital processing," *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, vol. 2021-June, 2021. DOI: 10.23919/VLSICircuits52068.2021.9492362.

[87] V. Joshi, M. Le Gallo, S. Haefeli, *et al.*, "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol. 11, no. 1, 2020, ISSN: 20411723. DOI: 10.1038/s41467-020-16108-9. arXiv: 1906.03138.

[88] M. Pasotti, R. Zurla, M. Carissimi, *et al.*, "A 32-KB ePCM for Real-Time Data Processing in Automotive and Smart Power Applications," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 7, pp. 2114–2125, 2018, ISSN: 00189200. DOI: 10.1109/JSSC.2018.2828805.

[89] M. Boniardi and D. Ielmini, "Physical origin of the resistance drift exponent in amorphous phase change materials," in *AIP Applied Physics Letters*, 2011. DOI: 10.1063/1.3599559.

[90] C.-X. Xue, J.-M. Hung, H.-Y. Kao, *et al.*, "16.1 a 22nm 4mb 8b-precision reram computing-in-memory macro with 11.91 to 195.7tops/w for tiny ai edge devices," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 245–247. DOI: 10.1109/ISSCC42613.2021.9365769.

[91] H. Jia, M. Ozatay, Y. Tang, *et al.*, "15.1 a programmable neural-network inference accelerator based on scalable in-memory computing," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 236–238. DOI: 10.1109/ISSCC42613.2021.9365788.

[92] M. E. Sinangil, B. Erbagci, R. Naous, *et al.*, "A 7-nm compute-in-memory sram macro supporting multi-bit input, weight and output and achieving 351 tops/w and 372.4 gops," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 188–198, 2021. DOI: 10.1109/JSSC.2020.3031290.

[93] R. Khaddam-Aljameh, M. Stanisavljevic, J. F. Mas, *et al.*, "Hermes core – a 14nm cmos and pcm-based in-memory compute core using an array of 300ps/lsb linearized cco-based adcs and local digital processing," in *2021 Symposium on VLSI Technology*, 2021, pp. 1–2.

[94] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[95] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, ISSN: 00189219. DOI: 10.1109/5.726791. [Online]. Available: http://ieeexplore.ieee.org/document/726791/ (visited on 10/14/2022).

[96] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980 [cs]*, 2017. [Online]. Available: http://arxiv.org/abs/1412.6980 (visited on 02/11/2022).

[97] V. Peluso and A. Calimera, "Energy-driven precision scaling for fixed-point convnets," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2018, pp. 113–118. DOI: 10.1109/VLSI-SoC.2018.8644902.

[98] D. Ielmini, N. Lepri, P. Mannocci, and A. Glukhov, "Status and challenges of in-memory computing for neural accelerators," in *2022 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, Hsinchu, Taiwan: IEEE, 2022, pp. 1–2, ISBN: 9781665409230. DOI: 10.1109/VLSI-TSA54299.2022.9770972. [Online]. Available: https://ieeexplore.ieee.org/document/9770972/ (visited on 10/14/2022).

[99] 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 *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19, New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 1–6, ISBN: 9781450367257. DOI: 10.1145/3316781.3317870. [Online]. Available: https://doi.org/10.1145/3316781.3317870 (visited on 12/12/2022).

[100] 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 *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, ISSN: 1558-1101, Mar. 2022, pp. 1119–1122. DOI: 10.23919/DATE54114.2022.9774736.