Computational phase-change memory: beyond von Neumann computing

To cite this article: Abu Sebastian et al 2019 J. Phys. D: Appl. Phys. 52 443002

View the article online for updates and enhancements.
1. Introduction

Historically, there has always been a technology that defines and shapes the economy and society over a significant period of time. It is widely believed that in the coming decades, this technology will be artificial intelligence (AI). However, AI faces a severe computing efficiency problem. Even though recent demonstrations of AI such as IBM’s Watson beating two former world champions in Jeopardy! [2] or AlphaGo beating 18-time world champion Lee Sedol in Go [3] are truly remarkable, they were achieved at the expense of an energy consumption that was orders of magnitude higher than that of the human brain. A key reason for this inefficiency is that the vast majority of AI algorithms run on conventional computing systems such as central processing units (CPUs), graphical processing units (GPUs) and field programmable gate arrays (FPGAs). Recently, application-specific integrated circuits (ASICs) that rely on reduced precision arithmetic and highly optimized dataflow are being actively developed [4, 5]. However, one of the central issues, namely, the need to shuttle large amounts of data back and forth during computation, remains unaddressed. In highly data-centric computing, most of the energy is consumed not in computation but in transferring data to and from the memory [6]. It is estimated that an internal cache access uses an order of magnitude more energy than typical arithmetic operations whereas access from the dynamic random access memory (DRAM) typically consumes another two orders of magnitude more energy [7, 8]. Thus, it is becoming increasingly clear that to build highly efficient AI computing hardware, we need to transition to novel architectures where memory and processing are better collocated. Computational memory is one such approach where certain computational tasks are performed in place in the memory itself by exploiting the physical attributes and state dynamics of the memory devices [1, 9–15]. In this paradigm, the memory elements are not just used to store information but they also execute
computational tasks with collocated memory and processing often at considerably reduced time complexity (see figure 1).

2. Phase-change memory as computational memory

Resistive memory devices or memristive devices are particularly well suited for in-memory computing. In memristive devices, information is stored based on the resistance or conductance of nanoscale devices [16, 17]. Their operation relies on a range of physical mechanisms such as ionic drift [18], magnetoresistance [19], and phase transition [20]. One such resistive memory device is phase-change memory (PCM) which indeed has a long history starting with the pioneering work of Stanford R. Ovshinsky [21]. Already in 1970, a 256-bit array of memory cells was developed by Neale et al [22]. Further attempts to develop reliable PCM devices from the early 2000s up to early 2000s encountered significant difficulties due to device degradation and instability of operation, and thus the interest in making electrical memory devices with phase-change materials gradually decreased. However, phase-change materials became widely used since the 1990s in optical memory devices and even today, they serve as the information storage medium in CDs, DVDs and Blu-Ray disks [23]. The research results and success of optical storage with phase-change materials led to a renewed interest in PCM from the early 2000s onwards [24–28]. PCM is arguably the most advanced resistive memory technology and is currently establishing itself in the application area of storage class memory [29, 30]. However, the focus of this review article will be on its application as computational memory.

PCM exploits the behavior of certain phase-change materials that can be switched reversibly between amorphous and crystalline phases of different electrical resistivity. These materials are typically compounds of Ge, Sb and Te. The amorphous phase tends to have high electrical resistivity, while the crystalline phase exhibits a low resistivity, sometimes three or four orders of magnitude lower. A PCM device consists of a certain volume of this phase change material sandwiched between two electrodes. Figure 2 shows a schematic illustration of a PCM device with a ‘mushroom-type’ device geometry [31]. An access device such as a field effect transistor (FET) is typically placed in series with a PCM device. Applying current pulses to a PCM device results in significant heating due to Joule heating. A RESET pulse refers to a current pulse which can melt a significant portion of the phase change material. When the pulse is stopped abruptly, the molten material quenches into the amorphous phase due to glass transition. In the resulting RESET state, the device will be in a high resistance state provided, the amorphous region blocks the bottom electrode. When a current pulse (typically referred to as the SET pulse) is applied to a PCM device in the RESET state, a part of the amorphous region crystallizes. The temperature that corresponds to the highest rate of crystallization is typically ≈400 °C, which is lower than the melting temperature (≈600 °C). The resistance state achieved after the application of RESET or SET pulses can be deciphered by biasing the device with a small amplitude read voltage that does not disturb the phase-configuration.

2.1. Key enabling properties for in-memory computing

One of the key properties of PCM that enable in-memory computing is simply the ability to store two levels of resistance/conductance values in a non-volatile manner and to reversibly switch from one level to the other (binary storage capability). As described later in section 3, these SET and RESET states could serve as additional logic state variable. Figure 3(a) shows the resistance values achieved upon repeated switching of a PCM device between the SET and RESET states. Such curves are typically referred to as ‘cycling endurance’ curves. It can be seen that it is possible to achieve over 10⁹ switching cycles in a PCM device. This is quite remarkable given that there is significant intentional atomic rearrangement in the active volume of these devices. However, note that the absolute values of SET and RESET states could change with repeated cycling and this could have some ramifications for in-memory computing. Moreover, across a large array of devices, there will be significant intra-device variability associated with the SET and RESET states (see figure 3(b)).

Another key property of PCM that enables in-memory computing is its ability to achieve not just two levels but a
continuum of resistance or conductance values (analog storage capability) [32]. This is typically achieved by creating intermediate phase configurations through the application of suitable partial RESET pulses [33, 34]. For example, figure 4(a) shows a continuum of resistance levels achieved by the application of RESET pulses with varying amplitude. The device is first programmed to the fully crystalline state, after which RESET pulses are applied with progressively increasing amplitude. The device resistance is measured after the application of each RESET pulse. The device resistance, related to the size of the amorphous region, increases with increasing RESET current. The curve shown in figure 4(a) is typically referred to as the programming curve. The programming curve is usually bidirectional, i.e. it is possible to increase as well as decrease the resistance by modulating the programming current. A PCM device can be programmed to a certain desired resistance value through iterative programming by applying several pulses in a closed-loop manner [34]. In iterative programming, after each programming pulse, a read-verify step is performed. The programming current applied to the PCM device in the subsequent iteration is adapted according to the value of the error between the target level and the read value of the device conductance. The algorithm runs until the programmed conductance reaches a value within a predefined margin from the target value. Figure 4(b) shows experimental results of the iterative programming of 32 representative conductance levels of approx. 10,000 devices from a prototype multi-level PCM chip fabricated in the 90 nm technology node. Even though it is possible to achieve a desired resistance value through iterative programming, there are significant temporal fluctuations associated with the resistance values. There is a temporal evolution of the conductance values (drift) arising from a spontaneous structural relaxation of the amorphous phase [35, 36]. PCM devices also exhibit significant 1/f noise behavior [37]. Moreover, the ambient temperature variations lead to significant resistance changes due to the thermally activated nature of electrical transport in amorphous phase-change materials [38].

The third key property that enables in-memory computing is the accumulative behavior arising from the crystallization dynamics [39]. It is possible to progressively reduce the device resistance by the successive application of SET pulses with the same amplitude (see figure 5(a)). Figure 5(b) shows experimental measurements of this accumulative behavior across an array of PCM devices. It can be seen that the mean conductance increases monotonically with an increasing SET current (ranging from 50 µA to 100 µA) and with increasing number of SET pulses. Note that it is very challenging to achieve a progressive increase in the size of the...
amorphous region. Hence, the curves shown in figures 5(a) and (b), typically referred to as accumulation curves, are unidirectional. It can also be seen from figures 5(a) and (c) that there is significant intra and inter-device randomness associated with the accumulation process attributed to the crystallization dynamics [40–42]. The crystallization mechanism is assumed to be mainly dominated by crystal growth due to the large amorphous-crystalline interface area and small volume of the amorphous region. Although crystal growth is a deterministic process, small variations in the atomic configurations of the amorphous volume created upon RESET can lead to variations in the effective amorphous thickness initially created [40].

3. Logical operations

In this section, we investigate in-memory computing that exploits the binary storage capability of PCM devices. In conventional CMOS, voltage serves as the single logic state variable. The input signals are processed as voltage signals and are output as voltage signals. By combining CMOS circuitry with resistive memory devices, it is possible to exploit an additional logical state variable, namely the resistance/conductance. For example, the high resistance could indicate logic ‘0’ and the low resistance could denote logic ‘1’. This enables logical operations that rely on the interaction between the voltage and resistance state variables and this could enable the seamless integration of processing and storage. This is the essential idea behind memristive logic, which is an active area of research [43, 44].

One particularly interesting characteristic of certain memristive logic families is statefulness. In this case, the Boolean variable is represented only in terms of the resistance states. The same devices are used simultaneously to store the inputs, to perform the logic operation, and to store the output result. By using a ‘material implication’ gate \( q \leftarrow p \IMp q \) (equivalent to \( \text{NOT}p \lor q \)) combined with a FALSE operation, Borghetti et al showed that the NAND logic operation can be performed using three memristive devices connected to a load resistor, implying extension to all Boolean logic (since NAND is logically complete) [43]. It was shown in subsequent work that the IMP logic can be implemented in a memristive crossbar [45]. Such a crossbar implementation can be used to design more complex logic units, for example an 8-bits full adder, by connecting 27 memristive devices on the same bitline [45]. Additional memristive logic crossbar architectures have been demonstrated based on different basic logic operations such as NOR, which is also logically complete [46–48]. Many of the stateful logic concepts can be extended to PCM devices that operate in a binary manner, even though there is a scarcity of published literature on this topic.

However, there are reports of non-stateful logic using the physics of crystallization [49] and melting [50] of PCM. In those schemes, the inputs are applied as voltage pulses to a single PCM device and the output is given by the resulting resistance state of the device after the application of the pulses. One potential merit of the phase-change logic is that the same PCM device can be used for performing different logic operations. For example, NOR and NAND can both be realized using a single PCM device with different ways of generating the input signals [50]. However, in contrast to stateful logic, only the output of the computation is stored in the device whereas the inputs are external voltage pulses. In
this sense, this variant of memristive logic is not state-full, because not only the resistance state of the memristive device is used to represent the data throughout the computation (hence, a conversion from resistance to voltage is needed to perform consecutive logic operations) [51].

There are also reports of performing bulk bitwise operations very efficiently inside a memory chip by enabling the PCM sense amplifier to detect fine-grained differences in cell resistance [52]. With the enhanced sense amplifier, bitwise AND/OR operations are performed by simultaneously sensing multiple PCM cells connected to the same sense amplifier. In this implementation, the PCM devices are programmed rather infrequently and hence the relatively low cycling endurance of PCM devices is not detrimental. Applications such as in-memory database queries and learning frameworks such as hyperdimensional computing [53] could benefit immensely from in-memory logical operations [6].

4. Matrix-vector multiplication

A very useful in-memory computing primitive enabled by the analog storage capability of PCM devices, is matrix-vector multiplication [12, 55]. The physical laws that are exploited to perform this operation are Ohm’s law and Kirchhoff’s current summation laws. For example, to perform the operation $Ax = b$, the elements of $A$ are mapped linearly to the conductance values of PCM devices organized in a crossbar configuration (see figure 6(a)). The $x$ values are mapped linearly to the amplitudes or durations of read voltages and are applied to the crossbar along the rows. The result of the computation, $b$, will be proportional to the resulting current measured along the columns of the array. Note that, if the inputs are mapped onto durations, the result $b$ will be proportional to the total charge (e.g. current integrated over a certain fixed period of time). It is also possible to perform a matrix-vector multiplication with the transpose of $A$ using the same cross-bar configuration. This is achieved by applying the input voltage to the column lines and measuring the resulting current along the rows. Mapping of the matrix elements to the conductance values can be achieved through iterative programming as described earlier [34]. The negative elements of $x$ are typically applied as negative voltages whereas the negative elements of $A$ are coded on separate devices together with a subtraction circuit. An experimental demonstration of matrix-vector multiplication using PCM devices fabricated in the 90 nm technology node is shown in figure 6(b). In this experiment, $A$ is a $256 \times 256$ Gaussian matrix coded in a prototype PCM chip and $x$ is a 256-long Gaussian vector applied as voltages to the devices. Each scalar multiply operation using Ohm’s law has an accuracy comparable to 4-bit fixed-point arithmetic. Much of this
inaccuracy arises from the conductance variations arising from drift and 1/f noise. It is assumed that the accumulation of current along the columns is of arbitrarily high precision.

One of the most promising applications of in-memory matrix-vector multiplication is deep learning inference [56–60]. Recently, artificial deep neural networks (DNNs) have shown remarkable human-like performance in cognitive tasks such as processing image, audio and natural language. DNNs are loosely inspired by biological neural networks. Parallel processing units called neurons are interconnected by plastic synapses. By tuning the weights of these interconnections, these networks are able to solve certain problems remarkably well. Deep learning inference refers to just the forward propagation in a DNN once the weights have been learned. A highly efficient inference engine could be designed where the computational memory unit is used to store the synaptic weights as well as to perform the matrix-vector multiply operations in place without the need to shuttle around the synaptic weight values from memory to processing units. PCM is also an excellent candidate for storing synaptic weights and this concept was experimentally demonstrated for the task of handwritten digit classification using a two layer neural network [61]. A cloud-based API enables users to run this inference experiment on a computational memory hardware located at IBM Research—Zurich. In this experiment, each synaptic weight is represented by two PCM devices organized in a differential configuration. More recent demonstrations show the efficacy of this concept even for state-of-the-art convolutional neural networks such as residual networks [59].

In-memory matrix-vector multiplications could also play a key role in signal processing related applications such as compressed sensing and recovery [62, 63]. In this application, a high dimensional signal is acquired at a sub-Nyquist sampling rate and is subsequently reconstructed. Unlike many other types of compression techniques, the signal is getting compressed as it is sampled. The compressed measurements can be viewed as a mapping of a signal $s$ (length $N$) to a measured signal $y$ (length $M < N$). The compression operator, if linear, can be modeled by a $M \times N$ measurement matrix $M$. The essential idea is to store this fixed measurement matrix in a computational memory unit comprising a crossbar array of PCM devices. Naturally, this facilitates the compression operation to be performed in $O(1)$ time complexity. However, for the reconstruction, one has to resort to more intricate algorithms such as the approximate message passing algorithm (AMP). AMP is an iterative algorithm that involves several matrix-vector multiplications on the very same measurement matrix and its transpose. Hence, we can employ the same crossbar array for the reconstruction. Moreover, it can be shown that the reconstruction complexity reduces from $O(MN)$ to $O(N)$. Le Gallo et al presented an experimental demonstration of compressed sensing recovery in the context of image compression [63]. A 128 × 128 pixel image was compressed by 50% and later recovered using the measurement matrix elements encoded in a PCM array.

5. Computing with accumulative behavior

In this section, we present in-memory computing that relies on the accumulative behavior of a group of PCM devices. The essential idea is as follows: depending on the operation to be performed, a suitable electrical signal is applied to the memory devices. The conductance of the devices evolves in accordance with the electrical input, and the result of the operation is retrieved by reading the conductance values at an appropriate time instance (see figure 7(a)).

The accumulation property of PCM has been exploited to perform the basic arithmetic processes of addition, multiplication, division and subtraction with simultaneous storage of the result [64, 65]. For example, performing a base-10 addition can be done by inputting a number of voltage pulses equal to the first addend, followed by voltage pulses equal in number to the second addend. In order to access the stored result, an undetermined number of pulses is additionally sent until a conductance threshold is reached. If this threshold is set such that it is always reached after the application of ten pulses in total, the number of additional pulses needed to reach the threshold after sending the two addends reveals the result. Similar schemes based on this accumulation property can be adapted for multiplication, division and subtraction.
A fascinating application of this concept is that of finding factors of numbers [65, 66]. Let us assume that a PCM device is initialized in such a way that after the application of a number of $X$ pulses, the resistance value drops below a certain threshold. To check whether $X$ is a factor of $Y$, $Y$ number of pulses are applied to the device, performing a RESET operation on the device each time the resistance value drops below the specified threshold. It can be seen that if after the application of $Y$ pulses, the resistance of the device is below the threshold, then $X$ is a factor of $Y$. Extending this concept, we could organize a number of $N$ PCM devices to switch to a low resistance value after the application of $N$ different $X$ values (see figure 7(b)). Thereafter, a number of $Y$ pulses are applied to all the $N$ devices in parallel. Now, by just reading back the resistance value of the devices after the application of the $Y$ pulses, one can decipher which $X$ values are factors of $Y$. An experimental illustration of this concept is shown in figure 7(c), where the $X$ values are chosen to be 4, 6, 9, 11 and 13.

Another demonstration of in-memory computing that exploits the accumulative behavior is that of unsupervised learning of temporal correlations between binary random processes [1]. Each process is assigned to a single PCM device and all the devices are initialized to be in the RESET state. A partial SET pulse is applied to the device whenever the process takes a value 1 and the amplitude of the pulse is such that it is proportional to the sum of all the processes at that time instance. This ensures that over time, the devices that are interfaced to the temporally correlated processes evolve towards high conductance values whereas the devices that are interfaced to the temporally uncorrelated processes remain at relatively low conductance values. This way, just by measuring the conductance values of the PCM devices and performing a binary classification, one could decipher the binary processes that are temporally correlated. This approach was experimentally demonstrated for a million weakly correlated processes assigned to a million PCM devices, thus proving the efficacy of this concept even in the presence of device variability and other non-ideal behavior [1].

The computational time complexity is reduced from $O(N)$ to $O(k \log(N))$, where $k$ is a small constant and $N$ is the number of data streams. A detailed system-level comparative study with respect to state-of-the-art computing hardware showed that, using such a computational memory module, it is possible to accelerate the task of correlation detection by a factor of 200 relative to an implementation that uses 4 P100 GPU devices.

6. Mixed-precision in-memory computing

For most of the computational tasks that were presented so far, approximate solutions were sufficient. However, there are some applications that require exact solutions. Mixed-precision in-memory computing is a promising approach to address this challenge. In this approach, a computational memory unit is used in conjunction with a high-precision computing unit (typically a von Neumann machine) [67] (see figure 8(a)). The essential idea is to use the computational memory unit to compute those segments of an algorithm where exactness is not essential. Through a judicious combination of the high and low precision units, it is possible to achieve arbitrarily high accuracy. The bulk of the computation is still realized in the computational memory and hence it is possible to achieve significant areal/power/speed improvements while addressing the challenge of imprecision associated with computational memory.

One prime application of mixed-precision in-memory computing is that of solving systems of linear equations. The problem is to find $x$ if $Ax = b$. As shown in figure 8(b), an initial solution is chosen as a starting point and is then iteratively updated based on a low-precision error-correction term, $z$. The correction term, $z$, is computed by solving yet another linear equation, $Az = r$, with an inexact inner solver, using the residual $r = b - Ax$. The matrix-vector multiply operations associated with the inner solver are performed using in-memory computing, albeit with reduced precision. However, the residual is calculated with high precision and the iterative algorithm runs until the norm of the residual falls below a desired
pre-defined tolerance, $tol$. Le Gallo et al presented an experimental demonstration of this concept using model covariance matrices [67]. It could be seen that the solution accuracy was not limited by the precision of in-memory computing. By performing enough iterations, the linear system could be solved down to a solution error of $\sim 1.3 \cdot 10^{-15}$. The error was ultimately limited by the machine precision of the high-precision processing unit. It was shown that since the majority of the computation is still performed in reduced precision, there is a significant overall gain in performance.

Another problem that is particularly well-suited to the mixed-precision in-memory computing concept is that of training DNNs. The training of DNNs is computational intense and is based on a global supervised learning algorithm that requires access to the first-order derivatives of the loss function. Typically this gradient is computed using the backpropagation algorithm. The input data is forward-propagated through the neuron layers with the synaptic networks performing multiply-accumulate operations. The final layer responses are compared with input data labels and the errors are back-propagated. Both steps involve sequences of matrix-vector multiplications. Subsequently, the synaptic weights are updated to reduce the error. Because of the need to repeatedly show very large datasets to very wide and deep neural networks, this optimization approach can take multiple days or weeks when training state-of-the-art networks on conventional machines. However, recent DL research shows that when training DNNs, it is possible to perform the forward and backward propagations rather imprecisely while the gradients need to be accumulated in high precision [69]. This observation makes the DL training problem amenable to the mixed-precision in-memory computing approach. The computational memory unit is used to store the synaptic weights and to perform the forward and backward passes, while the weight changes are accumulated in high precision (see figure 9(a)) [68, 70]. When the accumulated weight exceeds a certain threshold, pulses are applied to the corresponding memory devices to alter the synaptic weights. Note that these pulses are applied in a blind manner without verifying the conductance states of the corresponding PCM devices. It can be seen that the analog storage capability is exploited to perform the forward and backward passes (matrix-vector multiply operations) while the synaptic weight update relies on the accumulative behavior. This idea of mixed-precision deep learning using computational phase-change memory was tested using the handwritten digit classification problem based on the MNIST data set. A two-layered neural network was employed and models [71] of two PCM devices in differential configuration were used to represent the synaptic weights. Training was performed using 60000 images and the test accuracy was evaluated using 10000 images. The resulting test accuracy after 10 epochs of training was 97.78% which is remarkably close to that achieved by 62-bit floating point arithmetic (see figure 9(b)).

By exploiting the crossbar topology, it is also possible to estimate the gradient to perform the resulting synaptic weight update in place in O(1) complexity [72–75]. However, the test accuracy corresponding to a mixed hardware/software demonstration of this concept on the MNIST data set was 82.9% which is relatively low. In a recent experiment, compact ‘3T1C’ circuit structures that combine three transistors with one capacitor were shown to greatly increase the linearity and granularity of the weight update, allowing PCM devices to
be used for non-volatile storage of weight data transferred from the 3T1C structures [76]. In spite of using the same PCM devices used in the 2014 experiment, the classification accuracy was shown to increase the accuracy of the mixed-hardware-software experiment to software-equivalent levels. But the successful extension of this concept to deeper and larger networks need to be shown. By obviating the need to perform gradient accumulation externally, this approach could yield a better performance than the mixed-precision approach. However, significant improvements to the PCM technology, in particular the accumulative behavior, is needed to apply this to a wide range of DNNs.

7. Exploiting randomness

The randomness associated with the accumulative behavior is thought to be undesirable for applications such as training deep neural networks, learning temporal correlations etc. But there are a few applications where one could exploit this inherent randomness. Tuma et al exploited this randomness to design stochastically firing phase-change neurons and used a population of such neurons to represent high frequency signals [77]. Another fascinating application is random number generation that is important for a variety of areas such as stochastic computing, data encryption, machine learning and neuromorphic computing [78, 79]. In many instances, such circuits are the key limiting factors in the wide spread applicability of these concepts. Therefore, there is a significant interest in exploiting the inherent stochasticity of memristive devices as an entropy source for a true random number generator (TRNG) [80–82]. As opposed to a pseudo-random number generator (PRNG), a TRNG does not require a seed and uses the entropy coming from physical phenomena such as Johnson–Nyquist noise, time-dependent dielectric breakdown or ring oscillator jitter [83].

The inherent stochasticity associated with the accumulative behavior in PCM devices, which was discussed in section 2, can be exploited to generate a TRNG [40, 84]. As shown in figure 10(a), there is significant randomness associated with the total number of pulses needed to fully crystallize. In figure 10(b), the distributions of the number of pulses to crystallize for different pulse widths are shown. Based on these distributions, the number of pulses can be therefore adapted such that the device will switch with a given probability $p$ for a certain pulse width. The procedure for generating the random bits is shown in figure 10(c). To generate one bit, a sequence of one RESET, one crystallizing, and one READ pulse is applied. If the current during the READ pulse is higher than a threshold value (chosen to be 5 $\mu$A), a 1 is output, otherwise a 0. It is possible to achieve a bit throughput of approximately 1Mb/s. The characterization of random bitstreams generated
from 100M switching events was presented to evaluate the quality of the randomness of the bitstreams with standard tests such as the NIST test [85].

A key challenge is aging and temperature effects and these require some kind of feedback mechanism to ensure that the ratio of 1s (bias) remains close of 50%. Moreover, further post-processing of the data is necessary in order to have a suitable bias for the NIST tests. The bias of the raw 100M bitstream generated was 49.69%, which is still not sufficient for a sequence of this length to be considered truly random.

The bitstreams of hardware random generators are often prone to be slightly biased and algorithms such as the von Neumann algorithm [86] are employed to correct them. Figure 10(d) shows the results of the 15 NIST tests for 23M unbiased von Neumann-corrected bits. The bitstreams pass most tests except for the runs test. Other post-processing methods, such as XORing the bitstream with a cryptographically secure pseudorandom sequence, could help in this regard. Recently, a new approach using coupled resistive-RAM devices was proposed that provides unbiased RNG without the need of careful probability tracking and adjustment of switching parameters [87]. Another approach is to use the LSB of the stochastic switching delay time to generate the random bits. This was shown to eliminate bias-related issues and could pass all NIST tests without post-processing using a diffusive memristor device [88]. While a low-power, inherently stochastic nanoscale device may sound attractive for a TRNG, there are still important challenges that need to be overcome through careful engineering.

8. Discussion

PCM devices are also employed for the realization of synaptic and neuronal elements of artificial spiking neural networks (SNNs) exploiting both the analog storage capability as well as the accumulative behavior [77, 89–94]. SNNs are believed to be computationally more powerful, given the added temporal dimension. They are also much more biologically plausible than DNNs. However, there are key challenges such as the lack of killer applications that transcend conventional deep learning as well as robust scalable global training algorithms that can harness the local learning rules. A detailed exposition of this
promising yet nascent topical area is beyond the scope of this review.

Besides conventional electrical PCM devices, photonic memory devices [95] based on phase-change materials, which can be written, erased, and accessed optically, are rapidly bridging a gap towards all-photonic chip-scale information processing. By integrating phase-change materials onto an integrated photonics chip, the analog multiplication of an incoming optical signal by a scalar value encoded in the state of the phase change material was achieved [96]. In this device, the weight could be adjusted with optical write pulses carried by the same waveguide. This scheme of embedding a phase change element as an optically programmable attenuator has also been used for another example of optical in-memory computing, an optical abacus that can perform numerical operations with optical pulses as inputs [97].

System-level studies show that even with today’s PCM technology, we can achieve a significantly higher performance compared to conventional approaches [67]. This performance improvement is expected to be substantially higher for future generations of PCM devices. Phase change materials are known to undergo a reversible phase transition down to nanoscale dimensions [98]. It is also possible to operate these devices at timescales on the order of nanoseconds [99]. Moreover, the retention time, which is a key requirement for traditional memory applications, is less important for many in-memory computing applications and this could enable the exploration of new material classes such as elemental Antimony [100]. It was recently shown that Antimony could be melt-quenched to form a stable amorphous state at room temperature.

However, there are also several challenges associated with computational phase-change memory. One key challenge is the conductance variations arising from 1/f noise as well as the structural relaxation of the melt-quenched amorphous phase. As discussed earlier, there are also temperature-induced conductance variations. A promising solution towards addressing these challenges is that of projected phase-change memory (Projected PCM) [101, 102]. In these devices, there is a non-insulating projection segment in parallel to the phase-change material segment. By exploiting the highly non-linear I – V characteristics of phase-change materials, one could ensure that during the SET/RESET process, the projection segment has minimal impact on the operation of the device. However, during read, the device conductance is mostly determined by the projection segment that appears parallel to the amorphous phase-change segment. Recently, it was shown that it is possible to achieve remarkably high precision in-memory scalar multiplication (equivalent to 8-bit fixed point arithmetic) using projected PCM devices [103]. These projected PCM devices also facilitate array-level temperature compensation schemes. The limited endurance of PCM devices (the number of times the PCM devices can be SET and RESET) is another key challenge. The endurance is orders of magnitude higher (approx. $10^9$–$10^{12}$) [104] for PCM compared to other nonvolatile memory devices such as Flash memory (approx. $10^3$–$10^5$) [105]. But this could be inadequate for certain applications involving many write operations such as stateful logic operations. The limited endurance and various other non-idealities associated with the accumulative behavior such as limited dynamic range, nonlinearity and stochasticity can be partially circumvented with multi-PCM synaptic architectures. Recently, a multi-PCM synaptic architecture was proposed that employs an efficient counter-based arbitration scheme [106]. However, to improve the accumulation behavior, more research is required on the effect of device geometries [107] as well as the randomness associated with crystal growth [108].

To conclude, advances in AI are driving much of the innovations in next generation computing hardware. In-memory computing using post-CMOS memory devices is poised to have a significant impact on improving the energy/area efficiency as well as the latency compared to conventional computing systems with physically separated processing and memory units. Phase-change memory being the most advanced resistive memory technology could play a key role in in-memory computing. The well understood device physics, volumetric switching and the potential for scaling to nanoscale dimensions are key advantages of PCM. There are also challenges to be overcome, such as temporal variations of conductance values, relatively high RESET current as well as device fabrication issues when scaling to very small dimensions. However, in spite of these we believe that PCM-based in-memory computing cores could usher in a new era of non-von Neumann accelerators/coprocessors for AI applications.

Acknowledgments

We would like to thank the in-memory computing team at IBM Research—Zurich for their contributions to this article. We would also like to thank our colleagues at the IBM TJ Watson Research Center and IBM Research—Almaden as well as our academic collaborators from the New Jersey Institute of Technology, University of Patras, RWTH Aachen, University of Oxford, University of Exeter, ETH Zürich, and EPFL. This work was supported in part by the European Research Council through the European Union’s Horizon 2020 Research and Innovation Program under Grant 682675.

ORCID iDs

Abu Sebastian https://orcid.org/0000-0001-5603-5243
Manuel Le Gallo https://orcid.org/0000-0003-1600-6151

References

[1] Sebastian A, Tuma T, Papandreou N, Le Gallo M, Kull L, Parnell T and Eleftheriou E 2017 Nat. Commun. 8 1115
[2] Ferrucci D A 2012 IBM J. Res. Dev. 56 1
[3] Silver D et al 2017 Nature 550 354
[4] Jouppi N P et al 2017 ACM/IEEE 44th Annual Int. Symp. on Computer Architecture (Piscataway, NJ: IEEE) pp 1–12
[5] Fleischer B et al 2018 IEEE Symp. on VLSI Circuits (Piscataway, NJ: IEEE) pp 35–6
[70] Nandakumar S R, Le Gallo M, Boybat I, Rajendran B, Sebastian A and Eleftheriou E 2017 (arXiv:1712.01192)
[71] Nandakumar S, Le Gallo M, Boybat I, Rajendran B, Sebastian A and Eleftheriou E 2018 J. Appl. Phys. 124 152135
[72] Burr G W et al 2015 IEEE Trans. Electron Devices 62 3498
[73] Gokmen T and Vaslov Y 2016 Frontiers Neurosci. 10 333
[74] Tsai H, Ambrogio S, Narayanan P, Shelby R M and Burr G W 2018 J. Phys. D: Appl. Phys. 51 283001
[75] Haensch W, Gokmen T and Puri R 2019 Proc. IEEE 107 108
[76] Ambrogio S et al 2018 Nature 558 60
[77] Tuma T, Pantazi A, Le Gallo M, Sebastian A and Eleftheriou E 2016 Nat. Nanotechnol. 11 693
[78] Alaghi A and Hayes J P 2013 ACM Trans. Embedded Comput. Syst. (TECS) 12 92
[79] Gupta S, Agrawal A, Gopalakrishnan K and Narayanan P 2015 Int. Conf. on Machine Learning pp 1737–46
[80] Gaba S, Sheridan P, Zhou J, Choi S and Lu W 2013 Nanoscale 5 5872
[81] Balatti S, Ambrogio S, Wang Z and Ielmini D 2015 IEEE J. Emerg. Sel. Top. Circuits Syst. 5 214
[82] Vodenicarevic D et al 2017 Phys. Rev. Appl. 8 054045
[83] Yang K, Fick D, Henry M B, Lee Y, Blaauw D and Sylvester D 2014 IEEE Int. Solid-State Circuits Conf. Digest of Technical Papers (Piscataway, NJ: IEEE) pp 280–1
[84] Piccinini E, Brunetti R and Rudan M 2017 IEEE Trans. Electron Devices 64 2185
[85] Rukhin A, Soto J, Nechvatal J, Smid M and Barker E 2001 A statistical test suite for random and pseudorandom number generators for cryptographic applications Technical Report Booz-Allen and Hamilton Inc Mclean Va
[86] Peres Y 1992 Ann. Stat. 20 590
[87] Balatti S, Ambrogio S, Carbou R, Milo V, Wang Z, Calderoni A, Ramaswamy N and Ielmini D 2016 IEEE Trans. Electron Devices 63 2029
[88] Jiang H et al 2017 Nat. Commun. 8 882
[89] Suri M et al 2011 Int. Electron Devices Meeting (IEDM) pp 4.4.1–4
[90] Kuzum D, Jeyasingh R G, Lee B and Wong H-S P 2011 Nano Lett. 12 2179
[91] Jackson B L et al 2013 ACM J. Emerg. Technol. Comput. Syst. (JETC) 9 12
[92] Tuma T, Le Gallo M, Sebastian A and Eleftheriou E 2016 IEEE Electron Device Lett. 37 1238
[93] Sidler S, Pantazi A, Woźniak S, Leblebici Y and Eleftheriou E 2017 International Conference on Artificial Neural Networks (Berlin: Springer) pp 281–8
[94] Moraitis T, Sebastian A and Eleftheriou E 2018 IEEE Nanotechnol. Mag. 12 45
[95] Rios C, Stegmaier M, Hosseini P, Wang D, Scherer T, Wright C D, Bhaskaran H and Pernice W H 2015 Nat. Photon. 9 725
[96] Rios C, Youngblood N, Cheng Z, Gallo M L, Pernice W H, Wright C D, Sebastian A and Bhaskaran H 2019 Sci. Adv. 5 eaau5759
[97] Feldmann J, Stegmaier M, Gruhler N, Rios C, Bhaskaran H, Wright C and Pernice W 2017 Nat. Commun. 8 1256
[98] Xiong F, Liao A D, Estrada D and Pop E 2011 Science 332 568
[99] Bruns G, Merkelbach P, Schlockermann C, Salina M, Wuttig M, Happ T, Philipp J and Kund M 2009 Appl. Phys. Lett. 95 043108
[100] Salinga M, Kersting B, Ronneberger I, Jonnalagadda V P, Vu X T, Le Gallo M, Giannopoulos I, Cojocaru-Mirédin O, Mazzarello R and Sebastian A 2018 Nat. Mater. 17 681
[101] Kim S, Sosa N, BrightSky M, Mori D, Kim W, Zhu Y, Suu K and Lam C 2013 Int. Electron Devices Meeting (Piscataway, NJ: IEEE) pp 30–7
[102] Koelmans W W, Sebastian A, Jonnalagadda V P, Krebs D, Dellmann L and Eleftheriou E 2015 Nat. Commun. 6 8181
[103] Giannopoulos I, Sebastian A, Le Gallo M, Jonnalagadda V, Sousa M, Boon M and Eleftheriou E 2018 IEEE Int. Electron Devices Meeting (Piscataway, NJ: IEEE) p 27
[104] Kim W et al 2017 Int. Electron Devices Meeting (Piscataway, NJ: IEEE) p 4
[105] Pletka R, Kollisidas I, Ioannou N, Tomic S, Papandreou N, Parnell T, Pozidis H, Fry A and Fisher T 2018 ACM Trans. Storage 14 33
[106] Boybat I, Le Gallo M, Nandakumar S, Moraitis T, Parnell T, Tuma T, Rajendran B, Leblebici Y, Sebastian A and Eleftheriou E 2018 Nat. Commun. 9 2514
[107] La Barbera S, Ly D R, Navarro G, Castellani N, Cueto O, Bourgeois G, De Salvo B, Nowak E, Querlioz D and Vianello E 2018 Adv. Electron. Mater. 4 1800223
[108] Zhang W, Mazzarello R, Wuttig M and Ma E 2019 Nat. Rev. Mater. 4 150–68

13