This manuscript was published in: Nature Communications, volume 8, Article number 1115 (2017)
Link to published article: https://www.nature.com/articles/s41467-017-01481-9
DOI: https://doi.org/10.1038/s41467-017-01481-9
This file also includes the supplementary information that accompanied the manuscript.

Temporal correlation detection using computational phase-change memory

Abu Sebastian,[1, a)] Tomas Tuma,[1] Nikolaos Papandreou,[1] Manuel Le Gallo,[1] Lukas Kull,[1] Thomas Parnell,[1] and Evangelos Eleftheriou[1]
IBM Research – Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland.

Conventional computers based on the von Neumann architecture perform computation by repeatedly transferring data between their physically separated processing and memory units. As computation becomes increasingly data-centric and the scalability limits in terms of performance and power are being reached, alternative computing paradigms with collocated computation and storage are actively being sought. A fascinating such approach is that of computational memory where the physics of nanoscale memory devices are used to perform certain computational tasks within the memory unit in a non-von Neumann manner. We present an experimental demonstration using one million phase-change memory devices organized to perform a high-level computational primitive by exploiting the crystallization dynamics. Its result is imprinted in the conductance states of the memory devices. The results of using such a computational memory for processing real-world data-sets show that this co-existence of computation and storage at the nanometer scale could enable ultra-dense, low-power, and massively-parallel computing systems.

I. INTRODUCTION

In today's computing systems based on the conventional von Neumann architecture (Fig. 1(**a**)), there are distinct memory and processing units. The processing unit comprises the arithmetic and logic unit (ALU), a control unit and a limited amount of cache memory. The memory unit typically comprises dynamic random-access memory (DRAM), where information is stored in the charge state of a capacitor. Performing an operation (such as an arithmetic or logic operation), f, over a set of data stored in the memory, A, to obtain the result, $f(A)$, requires a sequence of steps in which the data must be obtained from the memory, transferred to the processing unit, processed, and stored back to the memory. This results in a significant amount of data being moved back and forth between the physically separated memory and processing units. This costs time and energy, and constitutes an inherent bottleneck in performance.

To overcome this, a tantalizing prospect is that of transitioning to a hybrid architecture where certain operations, such as f, can be performed at the same physical location as where the data is stored (Fig. 1(**b**)). Such a memory unit that facilitates collocated computation is referred to as computational memory. The essential idea is not to treat memory as a passive storage entity, but to exploit the physical attributes of the memory devices to realize computation exactly at the place where the data is stored. One example of computational memory is a recent demonstration of the use of DRAM to perform bulk bit-wise operations[1] and fast row copying[2] within the DRAM chip. A new class of emerging nanocale devices, namely, resistive memory or memristive devices with their non-volatile storage capability is particularly well suited for computational memory. In these devices, information is stored in their resistance/conductance states[3–6]. An early proposal for the use of memristive devices for in-place computing was the realization of certain logical operations using a circuit based on TiO_x-based memory devices[7]. The same memory devices were used simultaneously to store the inputs, perform the logic operation, and store the resulting output. Subsequently, more complex logic units based on this initial concept have been proposed[8–10]. In addition to performing logical operations, resistive memory devices, when arranged in a cross-bar configuration, can be used to perform matrix-vector multiplications in an analog manner. This exploits the multi-level storage capability as well as Ohm's law and Kirchhoff's law. Hardware accelerators based on this concept are now becoming an important subject of research[11–17]. However, in these applications, the cross-bar array of resistive memory devices serves as a non-von Neumann computing core and the results of the computation are not necessarily stored in the memory array.

Besides the ability to perform logical operations and matrix-vector multiplications, another tantalizing prospect of computational memory is that of realizing higher-level computational primitives by exploiting the rich dynamic behavior of its constituent devices. The dynamic evolution of the conductance levels of those devices upon application of electrical signals can be used to

[a)]Electronic mail: ase@zurich.ibm.com

perform in-place computing. A schematic illustration of this concept is shown in Fig. 1(**c**). 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 computation is imprinted in the memory array. One early demonstration of this concept was that of finding factors of numbers using phase-change memory (PCM) devices, a type of resistive memory devices[18–20]. However, this procedure is rather sensitive to device variabilities and thus experimental demonstrations were confined to a small number of devices. Hence, a large-scale experimental demonstration of a high-level computational primitive that exploits the memristive device dynamics and is robust to device variabilities across an array is still lacking.

In this paper, we present an algorithm to detect temporal correlations between event-based data streams using computational memory. The crystallization dynamics of PCM devices is exploited, and the result of the computation is imprinted in the very same memory devices. We demonstrate the efficacy and robustness of this scheme by presenting a large-scale experimental demonstration using an array of one million PCM devices. We also present applications of this algorithm to process real-world data-sets such as weather data.

II. RESULTS

Dynamics of phase-change memory devices

A PCM device consists of a nanometric volume of phase-change material sandwiched between two electrodes. A schematic illustration of a PCM device with mushroom-type device geometry is shown in Fig. 2(**a**))[21]. In an as-fabricated device, the material is in the crystalline phase. When a current pulse of sufficiently high amplitude is applied to the PCM device (typically referred to as the RESET pulse), a significant portion of the phase-change material melts owing to Joule heating. When the pulse is stopped abruptly, the molten material quenches into the amorphous phase because of the glass transition. In the resulting RESET state, the device will be in the low-conductance state as the amorphous region blocks the bottom electrode. The size of the amorphous region is captured by the notion of an effective thickness, u_a that also accounts for the asymmetric device geometry[22]. PCM devices exhibit a rich dynamic behavior with an interplay of electrical, thermal and structural dynamics that forms the basis for their application as computational memory. The electrical transport exhibits a strong field and temperature dependence[23]. Joule heating and the thermal transport pathways ensure that there is a strong temperature gradient within the PCM device. Depending on the temperature in the cell, the phase-change material undergoes structural changes, such as phase transitions and structural relaxation[24,25].

In our demonstration, we focus on a specific aspect of the PCM dynamics: the crystallization dynamics capturing the progressive reduction in the size of the amorphous region due to the phase transition from amorphous to crystalline (Fig. 2(**b**)). When a current pulse (typically referred to as the SET pulse) is applied to a PCM device in the RESET state such that the temperature reached in the cell via Joule heating is high enough, but below the melting temperature, a part of the amorphous region crystallizes. At the nanometer scale, the crystallization mechanism is dominated by crystal growth due to the large amorphous–crystalline interface area and the small volume of the amorphous region[24]. The crystallization dynamics in such a PCM device can be approximately described by

$$\frac{\mathrm{d}u_\mathrm{a}}{\mathrm{d}t} = -v_\mathrm{g}(T_\mathrm{int}), \tag{1}$$

where v_g denotes the temperature-dependent growth velocity of the phase-change material; $T_\mathrm{int} = R_\mathrm{th}(u_\mathrm{a})P_\mathrm{inp} + T_\mathrm{amb}$ is the temperature at the amorphous–crystalline interface, and $u_\mathrm{a}(0) = u_{\mathrm{a}_0}$ is the initial effective amorphous thickness[24]. T_amb is the ambient temperature, and R_th is the effective thermal resistance that captures the thermal resistance of all possible heat pathways. Experimental estimates of R_th and v_g are shown in Fig. 2(**c**) and Fig. 2(**d**), respectively[24]. From the estimate of R_th as a function of u_a, one can infer that the hottest region of the device is slightly above the bottom electrode and that the temperature within the device decreases monotonically with increasing distance from the bottom electrode. The estimate of v_g shows the strong temperature dependence of the crystal growth rate. Up to approx. 550 K, the crystal growth rate is negligible whereas it is maximum at approx. 750 K. As a consequence of Equation (1), u_a progressively decreases upon the application of repetitive SET pulses, and hence the low-field conductance progressively increases. In subsequent discussions, the RESET and SET pulses will be collectively referred to as write pulses. It is also worth noting that in a circuit-theoretic representation, the PCM device can be viewed as a generic memristor, with u_a serving as an internal state variable[26–28].

Statistical correlation detection using computational memory

In this section, we show how the crystallization dynamics of PCM devices can be exploited to detect statistical correlations between event-based data streams. This can be applied in various fields such as the Internet of Things (IoT), life sciences, networking, social networks, and large scientific experiments. For example, one could generate an event-based data stream based on the presence or absence of a specific word in a collection of tweets. Real-time processing of event-based data streams from dynamic vision sensors is another promising application area[29]. One can also view correlation detection as a key constituent of unsupervised learning where one of the objectives is to find correlated clusters in data streams.

In a generic formulation of the problem, let us assume that there are N discrete-time binary stochastic processes arriving at a correlation detector (see Fig. 3(**a**)). Let $\mathbf{X}_i = \{X_i(k)\}$ be one of the processes. Then $X_i(k)$ is a random variable with probabilities

$$P[X_i(k) = 1] = p \tag{2}$$
$$P[X_i(k) = 0] = 1 - p, \tag{3}$$

for $0 \leq p \leq 0.5$. Let $\mathbf{X}_j$ be another discrete-time binary stochastic process with the same value of parameter p. Then the correlation coefficient of the random variables $X_i(k)$ and $X_j(k)$ at time instant k is defined as

$$c = \frac{Cov[X_i(k), X_j(k)]}{\sqrt{Var[X_i(k)]Var[X_j(k)]}}. \tag{4}$$

Processes $\mathbf{X}_i$ and $\mathbf{X}_j$ are said to be correlated if $c > 0$ and uncorrelated otherwise. The objective of the correlation detection problem is to detect, in an unsupervised manner, an unknown subset of these processes that are mutually correlated.

As shown in Supplementary Note 1 and schematically illustrated in Fig. 3(**b**), one way to solve this problem is by obtaining an estimate of the uncentered covariance matrix corresponding the processes denoted by

$$\hat{R}_{ij} = \frac{1}{K}\sum_{k=1}^{K} X_i(k)X_j(k). \tag{5}$$

Next, by summing the elements of this matrix along a row or column, we can obtain certain numerical weights corresponding to the processes denoted by $\hat{W}_i = \sum_{j=1}^{N} \hat{R}_{ij}$. It can be shown that if $\mathbf{X}_i$ belongs to the correlated group with correlation coefficient $c > 0$, then

$$E[\hat{W}_i] = (N-1)p^2 + p + (N_\mathrm{c} - 1)cp(1-p). \tag{6}$$

N_c denotes the number of processes in the correlated group. In contrast, if $\mathbf{X}_i$ belongs to the uncorrelated group, then

$$E[\hat{W}_i] = (N-1)p^2 + p. \tag{7}$$

Hence by monitoring $\hat{W}_i$ in the limit of large K, we can determine which processes are correlated with $c > 0$. Moreover, it can be seen that with increasing c and N_c, it becomes easier to determine whether a process belongs to a correlated group.

We can show that this rather sophisticated problem of correlation detection can be solved efficiently using a computational memory module comprising PCM devices by exploiting the crystallization dynamics. By assigning each incoming process to a single PCM device, the statistical correlation can be calculated and stored in the very same device as the data passes through the memory. The way it is achieved is depicted schematically in Fig. 3(**c**): At each time instance k, a collective momentum, $M(k) = \sum_{j=1}^{N} X_j(k)$, that corresponds to the instantaneous sum of all processes is calculated. The calculation of $M(k)$ incurs little computational effort as it just counts the number of non-zero events at each time instance. Next, an identical SET pulse is applied potentially in parallel to all the PCM devices for which the assigned binary process has a value of 1. The duration or amplitude of the SET pulse is chosen to be a linear function of $M(k)$. For example, let the duration of the pulse $\delta t(k) = CM(k) = C\sum_{j=1}^{N} X_j(k)$. For the sake of simplicity, let us assume that the interface temperature, T_int, is independent of the amorphous thickness, u_a. As the pulse amplitude is kept constant, $v_\mathrm{g}(T_\mathrm{int}) = \mathscr{G}$, where $\mathscr{G}$ is a constant. Then from Equation 1, the absolute value of the change in the amorphous thickness of the i^{th} phase-change device at the k^{th} discrete-time instance is

$$\delta u_{\mathrm{a}_i}(k) = \delta t(k) v_\mathrm{g}(T_\mathrm{int}) = C\mathscr{G}\sum_{j=1}^{N} X_j(k). \tag{8}$$

The total change in the amorphous thickness after K time steps can be shown to be

$$\begin{aligned}
\Delta u_{\mathrm{a}_i}(K) &= \sum_{k=1}^{K} \delta u_{\mathrm{a}_i}(k) X_i(k) \\
&= C\mathscr{G} \sum_{k=1}^{K} \sum_{j=1}^{N} X_i(k) X_j(k) \\
&= C\mathscr{G} \sum_{j=1}^{N} \sum_{k=1}^{K} X_i(k) X_j(k) \\
&= KC\mathscr{G} \sum_{j=1}^{N} \hat{R}_{ij} \\
&= KC\mathscr{G} \hat{W}_i. \qquad (9)
\end{aligned}$$

Hence, from Equations 6 and 7, if $\mathbf{X}_i$ is one of the correlated processes, then Δu_{a_i} will be larger than if $\mathbf{X}_i$ is one of the uncorrelated processes. Therefore by monitoring Δu_{a_i} or the corresponding resistance/conductance for all phase-change devices we can determine which processes are correlated.

Experimental platform

Next, we present experimental demonstrations of the concept. The experimental platform (schematically shown in Fig. 4(**a**)) is built around a prototype PCM chip that comprises 3 million PCM devices. More details on the chip are presented in the methods section. As shown in Fig. 4(**b**)), the PCM array is organized as a matrix of word lines (WL) and bit lines (BL). In addition to the PCM devices, the prototype chip integrates the circuitry for device addressing and for write and read operations. The PCM chip is interfaced to a hardware platform comprising two field programmable gate array (FPGA) boards and an analog-front-end (AFE) board. The AFE board provides the power supplies as well as the voltage and current reference sources for the PCM chip. The FPGA boards are used to implement the overall system control and data management as well as the interface with the data processing unit. The experimental platform is operated from a host computer, and a Matlab environment is used to coordinate the experiments.

An extensive array-level characterization of the PCM devices was conducted prior to the experimental demonstrations. In one experiment, 10,000 devices were arbitrarily chosen and were first RESET by applying a rectangular current pulse of 1 μs duration and 440 μA amplitude. After RESET, a sequence of SET pulses of 50 ns duration were applied to all devices, and the resulting device conductance values were monitored after the application of each pulse. The map between the device conductance and the number of pulses is referred to as accumulation curve. The accumulation curves corresponding to different SET currents are shown in Fig. 4(**c**). These results clearly show that the mean conductance increases monotonically with increasing SET current (in the range from 50 μA and 100 μA) and with increasing number of SET pulses. From Fig. 4(**d**), it can also be seen that a significant variability is associated with the evolution of the device conductance values. This variability arises from inter-device as well as intra-device variability. The intra-device variability is traced to the differences in the atomic configurations of the amorphous phase created via the melt-quench process after each RESET operation[30,31]. Besides the variability arising from the crystallization process, additional fluctuations in conductance also arise from $1/f$ noise[32] and drift variability[33].

Experimental demonstration with a million processes

In a first demonstration of correlation detection, we created the input data artificially, and generated one million binary stochastic processes organized in a two-dimensional grid (Fig. 5(**a**)). We arbitrarily chose a subset of 95,525 processes, which we mutually correlated with a relatively weak instantaneous correlation coefficient of 0.1, whereas the other 904,475 were uncorrelated. The objective was to see if we can detect these correlated processes using the computational memory approach. Each stochastic process was assigned to a single PCM device. First, all devices were RESET by applying a current pulse of 1 μs duration and 440 μA amplitude. In this experiment, we chose to modulate the SET current while maintaining a constant pulse duration of 50 ns. At each time instance, the SET current is chosen to be equal to $0.002 * M(k)$ μA, where $M(k) = \sum_{j=1}^{N} X_j(k)$ is equal to the collective momentum. This rather simple calculation was performed in the host computer. Alternatively, it could be done in one of the FPGA boards. Next, the on-chip write circuitry was instructed to apply a SET pulse with the calculated SET current to all PCM devices for which $X_i(k) = 1$. To minimize the execution time, we chose not to program the devices if the SET current was less than 25 μA. The SET pulses were applied sequentially. However, if the chip has multiple write circuitries that can operate in parallel, then it is also possible to apply the SET pulses in parallel. This process of applying SET pulses was repeated at every time instance. The maximum SET current applied to the devices during the experiment was 80 μA.

As described earlier, owing to the temporal correlation between the processes, the devices assigned to those processes are expected to go to a high conductance state. We periodically read the conductance values of all PCM devices using the on-chip read circuitry and the on-chip analog-to-digital convertor (ADC). The resulting map of the conductance values is shown in Fig. 5(**b**). Also shown is the corresponding distribution of the conductance values (Fig. 5(**c**)). This distribution shows that we can distinguish between the correlated and the uncorrelated processes. We constructed a binary classifier by slicing the histogram of Fig. 5(**c**) according to some threshold, above which processes are labelled correlated and below which processes are labelled uncorrelated. The threshold parameter can be swept across the domain, resulting in an ensemble of different classifiers, each with its own statistical characteristics (e.g., precision and recall). The area under the precision-recall curve (AUC) is an excellent metric for quantifying the performance of the classifier. The AUC is 0.93 for the computational memory approach compared to 0.095 that corresponds to a random classifier that simply labels processes as correlated with some arbitrary probability. However, the performance is still short of that of an ideal classifier with AUC equal to one and this is attributed to the variability and conductance fluctuations discussed earlier. However, it is remarkable that in spite of these issues, we are able to perform the correlation detection with significantly high accuracy. Note that there are several applications, such as sensory data processing, where these levels of accuracy would be sufficient. Moreover, we could improve the accuracy by using multiple devices to interface with a single random process and by averaging their conductance values. This concept is also illustrated in the experimental demonstration on weather data that is described next. The conductance fluctuations can also be minimized using concepts such as projected phase-change memory[34].

Note that the correlations need to be detected within a certain period of time. This arises from the finite conductance range of the PCM devices. There is a limit to the u_a and hence the maximum conductance values that the devices can achieve. The accumulation curves in Fig. 4(**d**) clearly show that the mean conductance values begin to saturate after the application of a certain number of pulses. If the correlations are not detected within a certain amount of time, the conductance values corresponding to the correlated processes saturate while those corresponding to the uncorrelated processes continue to increase. Once the correlations have been detected, the devices need to be RESET, and the operation has to be resumed to detect subsequent correlations. The application of shorter SET pulses is one way to increase this time period. The use of multiple devices to interface with the random processes can also increase the overall conductance range.

As per Equation (6), we would expect the level of separation between the distributions of correlated and uncorrelated groups to increase with increasing values of the correlation coefficient. We could confirm experimentally that the correlated groups can be detected down to very low correlation coefficients such as $c = 0.01$ (see Supplementary Note 2, Supplementary Movie 1 and Supplementary Movie 2). We also quantified the performance of the binary classifier by obtaining the precision-recall curves and could show that in all cases, the classifiers performed significantly better than a baseline, random classifier (see Supplementary Figure 2). Experiments also show that there is a potential for this technique to be extended to detect multiple correlated groups having different correlation coefficients (see Supplementary Note 3).

Experimental demonstration with weather data

A second demonstration is based on real-world data from 270 weather stations across the USA. Over a period of 6 months, the rainfall data from each station constituted a binary stochastic process that was applied to the computational memory at one-hour time steps. The process took the value 1 if rainfall occurred in the preceding one-hour time window, else it was 0 (Fig. 5(**d**)). An analysis of the uncentered covariance matrix shows that several correlated groups exist and that one of them is predominant. As expected, also a strong geographical correlation with the rainfall data exists (Fig. 5(**e**)). Correlations between the rainfall events are also reflected in the geographical proximity between the corresponding weather stations. To detect the predominant correlated group using computational memory, we used the same approach as above, but with 4 PCM devices interfacing with each weather station data. The four devices were used to improve the accuracy. At each instance in time, the SET current was calculated to be equal to $0.0013 * M(k)$ μA. Next, the PCM chip was instructed to program the 270×4 devices sequentially with the calculated SET current. The on-chip write circuitry applies a write pulse with the calculated SET current to all PCM devices for which $X_i(k) = 1$. We chose not to program the devices if the SET current was less than 25 μA. The duration of the pulse was fixed to be 50 ns, and the maximum SET current applied to the devices was 80 μA. The resulting device conductance map (averaged over the four devices per weather station) shows that the conductance values corresponding to the predominant correlated group of weather stations are comparably higher (Fig. 5(**f**)).

Based on a threshold conductance value chosen to be 2 μS (the threshold that gave the best classifier performance), we can classify the weather stations into correlated and uncorrelated weather stations. This conductance threshold was chosen to get the best classifier performance (see Supplementary Note 2). We can also make comparisons with established unsupervised classification techniques such as *k*-means clustering. It was seen that, out of the 270 weather stations, there was a match for 245 weather stations. The computational memory approach classified 12 stations as uncorrelated that had been marked correlated by the *k*-means clustering approach. Similarly, the computational memory approach classified 13 stations as correlated that had been marked uncorrelated by the *k*-means clustering approach. Given the simplicity of the computational memory approach, it is remarkable that it can achieve this level of similarity with such a sophisticated and well-established classification algorithm (see Supplementary Note 4 for more details).

III. DISCUSSION

The scientific relevance of the presented work is that we have convincingly demonstrated the ability of computational memory to perform certain high-level computational tasks in a non-von Neumann manner by exploiting the dynamics of resistive memory devices. We have also demonstrated the concept experimentally at the scale of a million PCM devices. Even though we programmed the devices sequentially in the experimental demonstrations using the prototype chip, we could also program them in parallel provided there is a sufficient number of write modules. A hypothetical computational memory module performing correlation detection need not be substantially different from conventional memory modules (see Supplementary Note 5). The main constituents of such a module will also be a memory controller and a memory chip. Tasks such as computing $M(k)$ can easily be performed in the memory controller. The memory controller can then convey the write/read instructions to the memory chip.

In order to gain insight into the potential advantages of a correlation detector based on computational memory, we have compared the hypothetical performance of such a module with that of various implementation using state-of-the-art computing hardware (see Supplementary Note 6). For this study, we have designed a multi-threaded implementation of correlation detection, an implementation that can leverage the massive parallelism offered by graphical processing units (GPUs), as well as a scale-out implementation that can run across several GPUs. All implementations were compiled and executed on an IBM Power System S822LC system. This system has 2 POWER8 CPUs (each comprising 10 cores) and 4 Nvidia Tesla P100 graphical processing units (attached using the NVLink interface). A detailed profiling of the GPU implementation reveals two key insights. Firstly, we find that the fraction of time computing the momentum $M(k)$ is around 2% of the total execution time. Secondly, we observe that the performance is ultimately limited by the memory bandwidth of the GPU device. We then proceed to estimate the time that would be needed to perform the same task using a computational memory module: we determine the time required to compute the momentum on the memory controller, as well as the additional time required to perform the in-memory part of the computation. We conclude that by using such a computational memory module, one could accelerate the task of correlation detection by a factor of 200 relative to an implementation that uses 4 state-of-the-art GPU devices. We have also performed power profiling of the GPU implementation, and conclude that the computational memory module would provide a significant improvement in energy consumption of two order of magnitude (see Supplementary Note 6).

An alternative approach to using PCM devices will be to design an application-specific chip where the accumulative behavior of PCM is emulated using complementary metal-oxide semiconductor (CMOS) technology using adders and registers (see Supplementary Note 7). However, even at a relatively large 90 nm-technology node, the areal footprint of the computational phase-change memory is much smaller than that of CMOS-only approaches, even though the dynamic power consumption is comparable. By scaling the devices to smaller dimensions and by using shorter write pulses, these gains are expected to increase several fold[35,36]. The ultra-fast crystallization dynamics and non-volatility ensure a multi-time-scale operating window ranging from a few tens of nanoseconds to years. These attributes are particularly attractive for slow processes, where the leakage of CMOS would dominate the dynamic power because of the low utilization rate.

It can be shown that a single-layer spiking neural network can also be used to detect temporal correlations[30]. The event-based data-streams can be translated into pre-synaptic spikes to a synaptic layer. Based on the synaptic weights, the postsynaptic potentials are generated and added to the membrane potential of a leaky integrate-and-fire neuron. The temporal correlations between the presynaptic input spikes and the neuronal-firing events result in an evolution of the synaptic weights due to a feedback-driven competition among the synapses. In the steady state, the correlations between the individual input streams can be inferred from the distribution of the synaptic weights or the resulting firing activity of the postsynaptic neuron. Recently, it was shown that in such a neural network, PCM devices can serve as the synaptic elements[37,38]. One could argue that the synaptic elements serve as some form of computational memory. Even though both approaches aim to solve the same problem, there are some notable differences. In the neural network approach, it is the spike-timing-dependent-plasticity rule and the network dynamics that enable correlation detection. One could use any passive multi-level storage element to store the synaptic weight. Also note that the neuronal input is derived based on the value of the synaptic weights. It is challenging to implement such a feedback architecture in a computational memory unit. Such feedback architectures are also likely to be much more sensitive to device variabilities and nonlinearities and are not well suited for detecting very low correlations[37,39].

Detection of statistical correlations is just one of the computational primitives that could be realized using the crystallization dynamics. Another application of crystallization dynamics is that of finding factors of numbers, which we referred to in the introduction[20]. Assume that a PCM device is initialized in such a way that after the application of X number of pulses, the conductance exceeds a certain threshold. To check whether X is a factor of Y, Y number of pulses are applied to the device, re-initializing the device each time the conductance exceeds the threshold. It can be seen that if after the application of Y pulses, the conductance of the device is above the threshold, then X is a factor of Y. Another fascinating application of crystallization dynamics is to realize matrix-vector multiplications. To multiple an $N \times N$ matrix, A, with a $N \times 1$vector, x, the elements of the matrix and the vector can be translated into the durations and amplitudes of a sequence of crystallizing pulses applied to an array of N PCM devices. It can be shown that by monitoring the conductance levels of the PCM devices, one obtains a good estimate of the matrix-vector product (see Supplementary Note 8). Note that such an approach consumes only N devices compared to the existing approach based on the Kirchhoff's circuit laws that requires $N \times N$ devices.

In addition to the crystallization dynamics, one could also exploit other rich dynamic behavior in PCM devices, such as the dynamics of structural relaxation. Whenever an amorphous state is formed via the melt-quench process, the resulting unstable glass state relaxes to an energetically more favorable ideal glass state[25,40–42] (see Supplementary Note 9). This structural relaxation, which codes the temporal information of the application of write pulses, can be exploited to perform tasks such as the detection of rates of processes in addition to their temporal correlations (see Supplementary Note 9). It is also foreseeable that by further coupling the dynamics of these devices, we can potentially solve even more intriguing problems. Suggestions of such memcomputing machines that could solve certain non-deterministic polynomial (NP) problems in polynomial (P) time by exploiting attributes, such as the inherent parallelism, functional polymorphism, and information overhead are being actively investigated[43,44]. The concepts presented in this work could also be extended to the optical domain using devices such as photonic PCM[45]. In such an approach, optical signals instead of electrical signals will be used to program the devices. These concepts are also not limited to PCM devices: several other memristive device technologies exist that possess sufficiently rich dynamics to serve as computational memory[46]. However, it is worth noting that PCM technology is arguably the most advanced resistive memory technology at present with a very well established multi-level storage capability[21]. The read endurance is assumed to be unlimited. There are also recent reports of more than 10^{12} RESET/SET endurance cycles[47]. Note that in our experiments, we mostly apply only the SET pulses, and in this case the endurance is expected to be substantially higher.

To summarize, the objective of our work was to realize a high-level computational primitive or machine-learning algorithm using computational memory. We proposed an algorithm to detect temporal correlations between event-based data streams that exploits the crystallization dynamics of PCM devices. The conductance of the PCM devices receiving correlated inputs evolves to a high value, and by monitoring these conductance values we can detect the temporal correlations. We performed a large-scale experimental demonstration of this concept using a million PCM devices, and could successfully detect weakly correlated processes in artificially generated stochastic input data. This experiment demonstrates the efficacy of this concept even in the presence of device variability and other non-ideal behavior. We also successfully processed real-world data sets from weather stations in the United States and obtained classification results similar to the k-means clustering algorithm. A detailed comparative study with respect to state-of-the-art von Neumann computing systems showed that computational memory could lead to orders of magnitude improvements in time/energy-to-solution compared to conventional computing systems.

IV. METHODS

Phase-change memory chip

The PCM devices were integrated into the chip in 90 nm CMOS technology[32]. The phase-change material is doped $Ge_2Sb_2Te_2$ (d-GST). The bottom electrode has a radius of approx. 20 nm and a length of approx. 65 nm, and was defined using a sub-lithographic key-hole transfer process[48]. The phase-change material is approx. 100 nm thick and extends to the top electrode. Two types of devices are available on-chip. They differ by the size of their access transistor. The first sub-array contains 2 million devices. In the second sub-array, which contains 1 million devices, the access transistors are twice as large. All experiments in this work were done on the second sub-array, which is organized as a matrix of 512 word lines (WL) and 2048 bit lines (BL). The selection of one PCM device is done by serially addressing a WL and a BL. A single selected device can be programmed by forcing a current through the BL with a voltage-controlled current source. For reading a PCM cell, the selected BL is biased to a constant voltage of 200 mV. The resulting read current is integrated by a capacitor, and the resulting voltage is then digitized by the on-chip 8-bit cyclic ADC. The total time of one read is 1 μs. The readout characteristic is calibrated by means of on-chip reference poly-silicon resistors.

Generation of 1M random processes and experimental details

Let $\mathbf{X}_r$ be a discrete binary process with probabilities $P(X_r(k) = 1) = p$ and $P(X_r(k) = 0) = 1 - p$. Using $\mathbf{X}_r$ as the reference process, N binary processes can be generated via the stochastic functions[39]

$$\theta = P(X_i(k) = 1 | X_r(k) = 1) = p + \sqrt{c}(1 - p) \tag{10}$$

$$\phi = P(X_i(k) = 1 | X_r(k) = 0) = p(1 - \sqrt{c}) \tag{11}$$

$$P(X_i(k) = 0) = 1 - P(X_i(k) = 1). \tag{12}$$

It can be shown that $E(X_i(k)) = p$ and $Var(X_i(k)) = p(1-p)$. If two processes $\mathbf{X}_i$ and $\mathbf{X}_j$ are both generated using Equations 10 to 12, then the expectation of their product is given by:

$$E(X_i(k)X_j(k)) = P(X_i(k) = 1, X_j(k) = 1) = \sum_{v \in \{0,1\}} P(X_i(k) = 1, X_j(k) = 1 | X_r(k) = v) P(X_r(k) = v).$$

Conditional on the value of the process $\mathbf{X}_r$, the two processes $\mathbf{X}_i$ and $\mathbf{X}_j$ are statistically independent by construction, and thus the conditional joint probability $P(X_i(k) = 1, X_j(k) = 1 | X_r(k) = v)$ can be factorized as follows:

$$\begin{aligned} E(X_i(k)X_j(k)) = &\sum_{v \in \{0,1\}} P(X_i(k) = 1 | X_r(k) = v) P(X_j(k) = 1 | X_r(k) = v) P(X_r(k) = v) \\ &= \theta^2 p + \phi^2 (1-p) \\ &= p^2 + cp(1-p), \end{aligned}$$

where the final equality is obtained by substituting the preceding expressions for θ and ϕ, followed by some simple algebraic manipulation. It is then straightforward to show that the correlation coefficient between the two processes is equal to c as shown below:

$$\begin{aligned} Cov(X_i(k)X_j(k)) &= E(X_i(k)X_j(k)) - E(X_i(k))E(X_j(k)) \\ &= p^2 + cp(1-p) - p^2 \end{aligned}$$

$$\frac{Cov(X_i(k)X_j(k))}{\sqrt{Var(X_i)Var(X_j)}} = c \tag{13}$$

For the experiment presented, we chose an $\mathbf{X}_r$ where $p = 0.01$. A million binary processes were generated. Of these, $N_c = 95,525$ are correlated with $c > 0$. The remaining 904,475 processes are mutually uncorrelated. Each process is mapped to one pixel of a 1000×1000 pixel black-and-white sketch of Alan Turing: white pixels are mapped to the uncorrelated processes; black pixels are mapped to the correlated processes. The seemingly arbitrary choice of N_c arises from the need to match with the black pixels of the image. The pixels turn on and off in accordance with the binary values of the processes. One phase-change memory device is allocated to each of the one million processes.

Weather data-based processes and experimental details

The weather data was obtained from the National Oceanic and Atmospheric Administration (http://www.noaa.gov/) database of quality-controlled local climatological data. It provides hourly summaries of climatological data from approximately 1600 weather stations in the United States of America. The measurements were obtained over a 6-month period from January 2015 to June 2015 (181 days, 4344 hours). We generated one binary stochastic process per weather station. If it rained in any given period of 1 hour in a particular geographical location corresponding to a weather station, then the process takes the value 1; else it will be 0. For the experiments on correlation detection, we picked 270 weather stations with similar rates of rainfall activity.

Data availability: The data that support the findings of this study are available from the corresponding author upon request.

REFERENCES

[1]Seshadri, V. *et al.* Buddy-ram: Improving the performance and efficiency of bulk bitwise operations using DRAM. *arXiv preprint arXiv:1611.09988* (2016).

[2]Seshadri, V. *et al.* Rowclone: fast and energy-efficient in-DRAM bulk data copy and initialization. In *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, 185–197 (ACM, 2013).

[3]Beck, A., Bednorz, J., Gerber, C., Rossel, C. & Widmer, D. Reproducible switching effect in thin oxide films for memory applications. *Applied Physics Letters* **77**, 139–141 (2000).

[4]Strukov, D. B., Snider, G. S., Stewart, D. R. & Williams, R. S. The missing memristor found. *Nature* **453**, 80–83 (2008).

[5]Chua, L. Resistance switching memories are memristors. *Applied Physics A* **102**, 765–783 (2011).

[6]Wong, H.-S. P. & Salahuddin, S. Memory leads the way to better computing. *Nature Nanotechnology* **10**, 191–194 (2015).

[7]Borghetti, J. *et al.* 'Memristive' switches enable 'stateful' logic operations via material implication. *Nature* **464**, 873–876 (2010).

[8]Kvatinsky, S. *et al.* Magic: Memristor-aided logic. *IEEE Transactions on Circuits and Systems II: Express Briefs* **61**, 895–899 (2014).

[9]Zha, Y. & Li, J. Reconfigurable in-memory computing with resistive memory crossbar. In *Proceedings of the 35th International Conference on Computer-Aided Design*, 120 (ACM, 2016).

[10]Vourkas, I. & Sirakoulis, G. C. Emerging memristor-based logic circuit design approaches: A review. *IEEE Circuits and Systems Magazine* **16**, 15–30 (2016).

[11]Burr, G. W. *et al.* Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices* **62**, 3498–3507 (2015).

[12]Shafiee, A. *et al.* ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *Proceedings of the 43rd International Symposium on Computer Architecture*, 14–26 (IEEE Press, 2016).

[13]Chi, P. *et al.* PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. In *Proceedings of the 43rd International Symposium on Computer Architecture*, 27–39 (IEEE Press, 2016).

[14]Bojnordi, M. N. & Ipek, E. Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 1–13 (IEEE, 2016).

[15]Burr, G. W. *et al.* Neuromorphic computing using non-volatile memory. *Advances in Physics: X* **2**, 89–124 (2017).

[16]Gallo, M. L. *et al.* Mixed-precision memcomputing. *arXiv preprint arXiv:1701.04279* (2017).

[17]Sheridan, P. M. *et al.* Sparse coding with memristor networks. *Nature nanotechnology* **12**, 784–789 (2017).

[18]Wright, C. D., Liu, Y., Kohary, K. I., Aziz, M. M. & Hicken, R. J. Arithmetic and biologically-inspired computing using phase-change materials. *Advanced Materials* **23**, 3408–3413 (2011).

[19]Wright, C. D., Hosseini, P. & Diosdado, J. A. V. Beyond von-Neumann computing with nanoscale phase-change memory devices. *Advanced Functional Materials* **23**, 2248–2254 (2013).

[20]Hosseini, P., Sebastian, A., Papandreou, N., Wright, C. D. & Bhaskaran, H. Accumulation-based computing using phase-change memories with FET access devices. *IEEE Electron Device Letters* **36**, 975–977 (2015).

[21]Burr, G. W. *et al.* Recent progress in phase-change memory technology. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **6**, 146–162 (2016).

[22]Sebastian, A., Papandreou, N., Pantazi, A., Pozidis, H. & Eleftheriou, E. Non-resistance-based cell-state metric for phase-change memory. *Journal of Applied Physics* **110**, 084505 (2011).

[23]Le Gallo, M., Kaes, M., Sebastian, A. & Krebs, D. Subthreshold electrical transport in amorphous phase-change materials. *New Journal of Physics* **17**, 093035 (2015).

[24]Sebastian, A., Le Gallo, M. & Krebs, D. Crystal growth within a phase change memory cell. *Nature Communications* **5**, 4314 (2014).

[25]Raty, J. Y. *et al.* Aging mechanisms in amorphous phase-change materials. *Nature Communications* **6** (2015).

[26]Corinto, F., Civalleri, P. P. & Chua, L. O. A theoretical approach to memristor devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **5**, 123–132 (2015).

[27]Ascoli, A., Corinto, F. & Tetzlaff, R. Generalized boundary condition memristor model. *International Journal of Circuit Theory and Applications* **44**, 60–84 (2016).

[28]Secco, J., Corinto, F. & Sebastian, A. Flux–Charge memristor model for Phase Change Memory. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2017).

[29]Liu, S.-C. & Delbruck, T. Neuromorphic sensory systems. *Current Opinion in Neurobiology* **20**, 288–295 (2010).

[30]Tuma, T., Pantazi, A., Le Gallo, M., Sebastian, A. & Eleftheriou, E. Stochastic phase-change neurons. *Nature Nanotechnology* **11**, 693–699 (2016).

[31]Le Gallo, M., Tuma, T., Zipoli, F., Sebastian, A. & Eleftheriou, E. Inherent stochasticity in phase-change memory devices. In *46th European Solid-State Device Research Conference (ESSDERC)*, 373–376 (IEEE, 2016).

[32]Close, G. *et al.* Device, circuit and system-level analysis of noise in multi-bit phase-change memory. In *IEEE International Electron Devices Meeting (IEDM)*, 29–5 (IEEE, 2010).

[33]Pozidis, H. *et al.* A framework for reliability assessment in multilevel phase-change memory. In *4th IEEE International Memory Workshop (IMW)*, 1–4 (IEEE, 2012).

[34]Koelmans, W. W. *et al.* Projected phase-change memory devices. *Nature communications* **6** (2015).

[35]Xiong, F., Liao, A. D., Estrada, D. & Pop, E. Low-power switching of phase-change materials with carbon nanotube electrodes. *Science* **332**, 568–570 (2011).

[36]Loke, D. *et al.* Breaking the speed limits of phase-change memory. *Science* **336**, 1566–1569 (2012).

[37]Tuma, T., Le Gallo, M., Sebastian, A. & Eleftheriou, E. Detecting correlations using phase-change neurons and synapses. *IEEE Electron Device Letters* **37**, 1238–1241 (2016).

[38]Pantazi, A., Woźniak, S., Tuma, T. & Eleftheriou, E. All-memristive neuromorphic computing with level-tuned neurons. *Nanotechnology* **27**, 355205 (2016).

[39]Gütig, R., Aharonov, R., Rotter, S. & Sompolinsky, H. Learning input correlations through nonlinear temporally asymmetric hebbian plasticity. *The Journal of neuroscience* **23**, 3697–3714 (2003).

[40]Boniardi, M. & Ielmini, D. Physical origin of the resistance drift exponent in amorphous phase change materials. *Applied Physics Letters* **98**, 243506 (2011).

[41]Sebastian, A., Krebs, D., Le Gallo, M., Pozidis, H. & Eleftheriou, E. A collective relaxation model for resistance drift in phase change memory cells. In *IEEE International Reliability Physics Symposium (IRPS)*, MY–5 (IEEE, 2015).

[42]Zipoli, F., Krebs, D. & Curioni, A. Structural origin of resistance drift in amorphous GeTe. *Physical Review B* **93**, 115201 (2016).

[43]Traversa, F. L. & Di Ventra, M. Universal memcomputing machines. *IEEE Transactions on Neural Networks and Learning Systems* **26**, 2702–2715 (2015).

[44]Di Ventra, M. & Pershin, Y. V. Just add memory. *Scientific American* **312**, 56–61 (2015).

[45]Ríos, C. *et al.* Integrated all-photonic non-volatile multi-level memory. *Nature Photonics* **9**, 725–732 (2015).

[46]Waser, R. & Aono, M. Nanoionics-based resistive switching memories. *Nature Materials* **6**, 833–840 (2007).
[47]Kim, W. *et al.* ALD-based confined PCM with a metallic liner toward unlimited endurance. In *IEEE International Electron Devices Meeting (IEDM)*, 4–2 (IEEE, 2016).
[48]Breitwisch, M. *et al.* Novel lithography-independent pore phase change memory. In *IEEE Symposium on VLSI Technology*, 100–101 (IEEE, 2007).

ACKNOWLEDGEMENTS

We would like to thank several colleagues from IBM Research - Zurich and the IBM T. J. Watson Research Center, USA in particular Haris Pozidis, Milos Stanisavljevic, Urs Egger and Matthew BrightSky. We would also like to thank Charlotte Bolliger for help with the preparation of the manuscript and Sheethal Alice Tharian for help with the artwork. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 682675).

AUTHOR CONTRIBUTIONS

A.S. and T.T. conceived the idea. A.S., N.P., and M.L.G. performed the experiments. A.S. and T.T. analyzed the data. T.P. and L.K. performed the comparative study with conventional computing systems. E.E. provided managerial support and critical comments. A.S. wrote the manuscript with input from all the authors.

COMPETING INTERESTS

The authors declare no competing financial interests.

FIGURE CAPTIONS

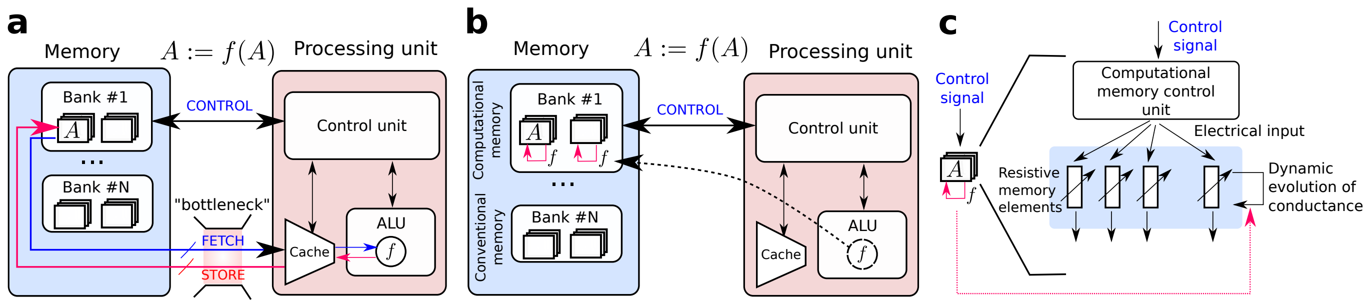


FIG. 1. **The concept of computational memory** (**a**) Schematic of the von Neumann computer architecture, where the memory and computing units are physically separated. A denotes information stored in a memory location. To perform a computational operation, $f(A)$, and to store the result in the same memory location, data is shuttled back and forth between the memory and the processing unit. (**b**) An alternative architecture where $f(A)$ is performed in place in the same memory location. (**c**) One way to realize computational memory is by relying on the state dynamics of a large collection of memristive devices. 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 can be retrieved by reading the conductance at an appropriate time instance.

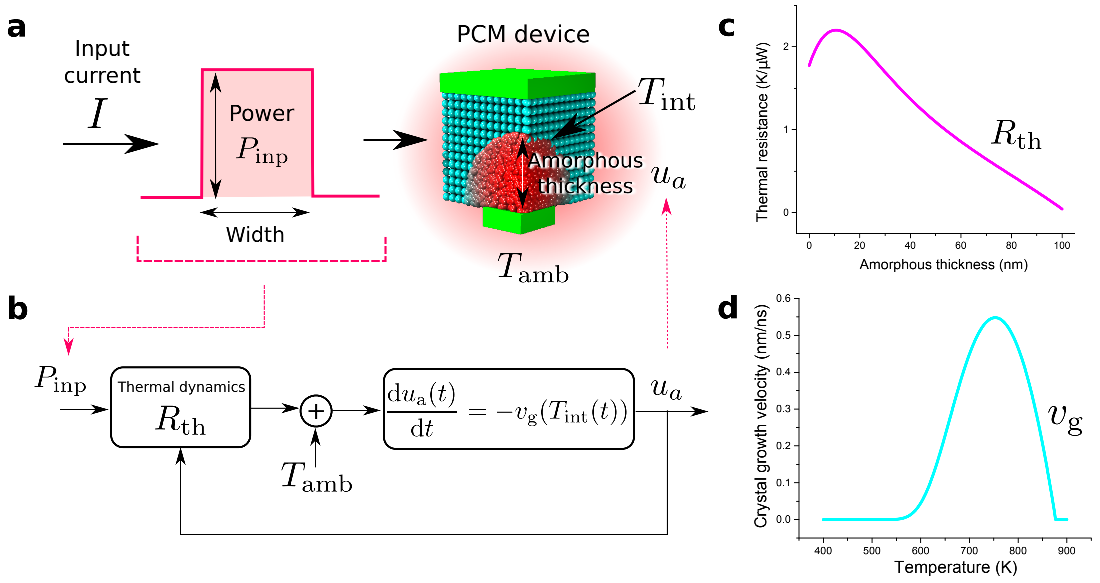


FIG. 2. **Crystallization dynamics** (**a**) Schematic of a mushroom-type phase-change memory device showing the phase configurations. (**b**) Illustration of the crystallization dynamics. When an electrical signal with power P_{inp} is applied to a PCM device, significant Joule heating occurs. The resulting temperature distribution across the device is determined by the thermal environment, in particular the effective thermal resistance, R_{th}. The effective thickness of the amorphous region, u_a, evolves in accordance with the temperature at the amorphous–crystalline interface, T_{int}, and with the temperature dependence of crystal growth, v_g. Experimental estimates of (**c**) R_{th} and (**d**) v_g.

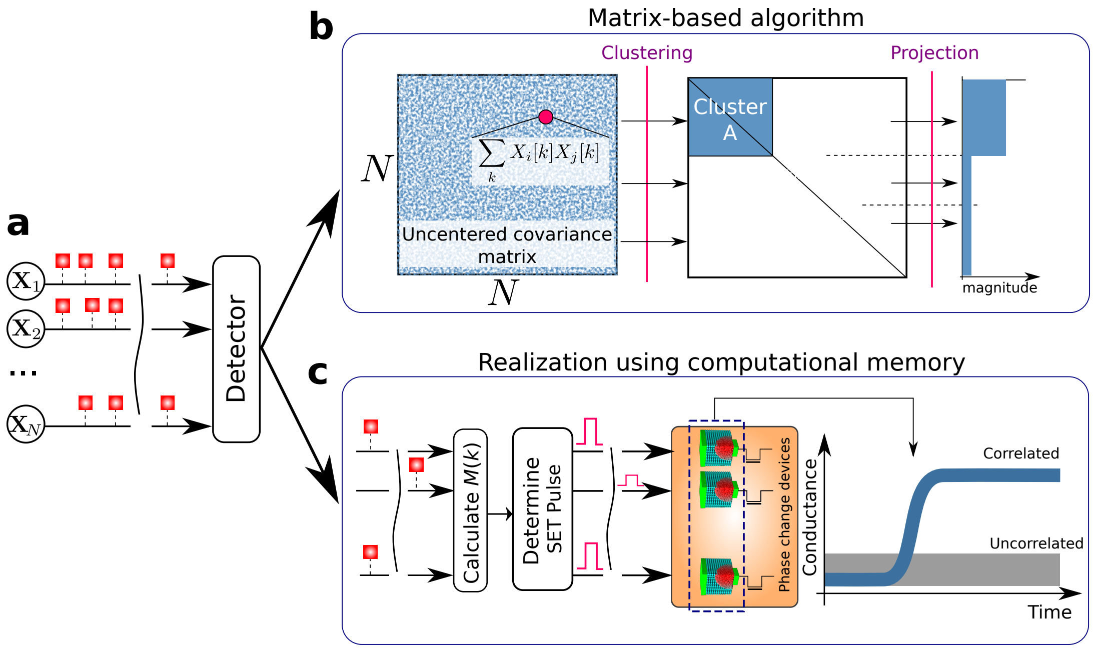


FIG. 3. **Temporal correlation detection** (**a**) Schematic of N stochastic binary processes, some correlated and the remainder uncorrelated, arriving at a correlation detector. (**b**) One approach to detect the correlated groups is to obtain an uncentered covariance matrix. By summing the elements of this matrix along a row or column, we can obtain some kind of numerical weights corresponding to the N processes and can differentiate the correlated from the uncorrelated group based on their magnitudes. (**c**) Alternatively, the correlation detection problem can be realized using computational memory. Here, each process is assigned to a single phase-change memory device. Whenever the process takes the value 1, a SET pulse is applied to the PCM device. The amplitude or the width of the SET pulse is chosen to be proportional to the instantaneous sum of all processes. By monitoring the conductance of the memory devices, we can determine the correlated groups.

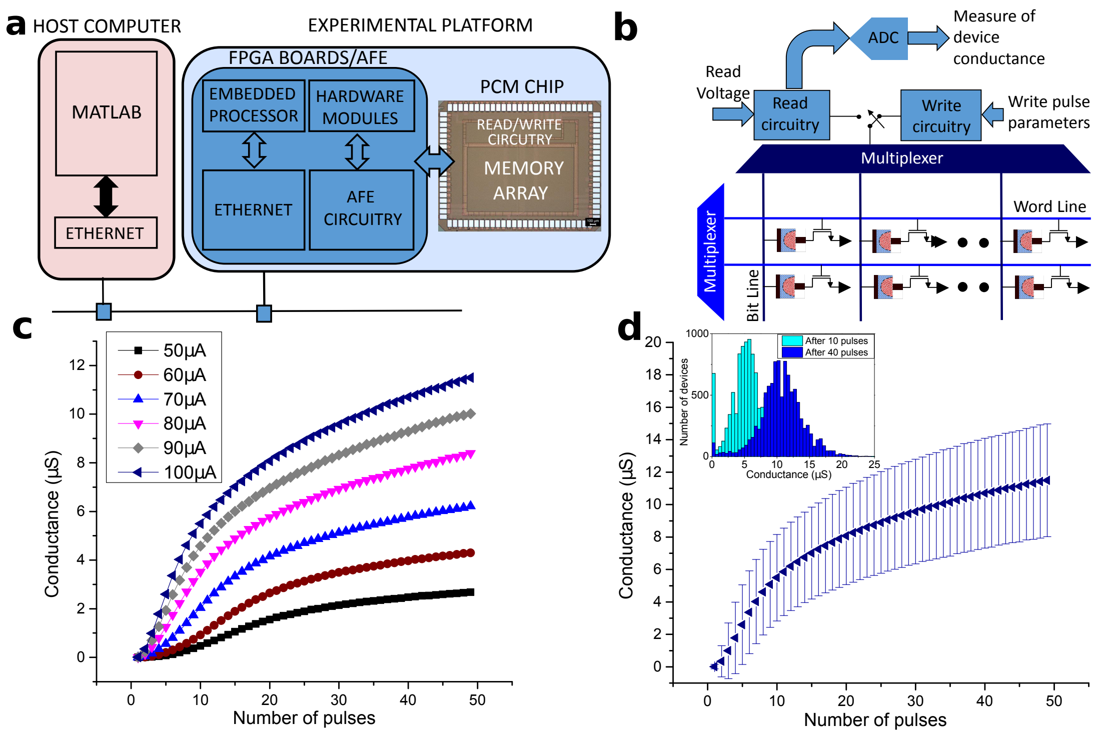


FIG. 4. **Experimental platform and characterization results** (**a**) Schematic illustration of the experimental platform showing the main components. (**b**) The phase-change memory array is organized as a matrix of word lines (WL) and bit lines (BL), and the chip also integrates the associated read/write circuitries. (**c**) The mean accumulation curve of 10,000 devices showing the map between the device conductance and the number of pulses. The devices achieve a higher conductance value with increasing SET current and also with increasing number of pulses. (**d**) The mean and standard deviation associated with the accumulation curve corresponding to the SET current of 100 μA. Also shown are the distributions of conductance values obtained after application of the 10th and 40th SET pulses.

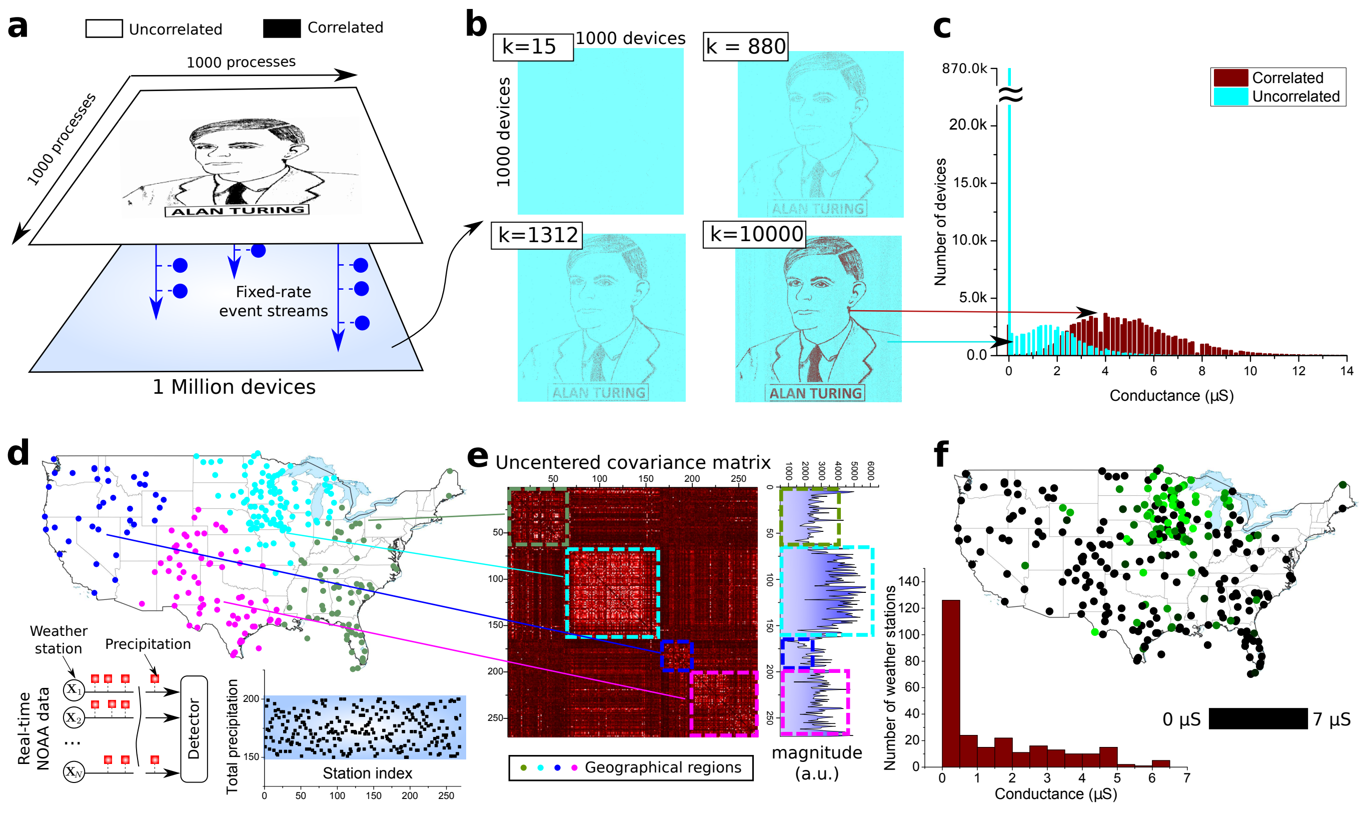


FIG. 5. **Experimental results** (**a**) A million processes are mapped to the pixels of a 1000 × 1000 pixel black-and-white sketch of Alan Turing. The pixels turn on and off in accordance with the instantaneous binary values of the processes. (**b**) Evolution of device conductance over time, showing that the devices corresponding to the correlated processes go to a high conductance state. (**c**) Distribution of the device conductance shows that the algorithm is able to pick out most of the correlated processes. (**d**) Generation of a binary stochastic process based on the rainfall data from 270 weather stations across the USA. (**e**) The uncentered covariance matrix reveals several small correlated groups, along with a predominant correlated group. (**f**) Map of the device conductance levels after the experiment shows that the devices corresponding to the predominant correlated group have achieved a higher conductance value.

SUPPLEMENTARY NOTE 1: METHODOLOGY FOR DETECTING TEMPORAL CORRELATIONS

This note provides additional details on the technique to detect correlations between random processes. Let $\mathbf{X}_i = \{X_i(k)\}$ be a discrete-time binary stochastic process. Then $X_i(k)$ is a random variable with probabilities

$$P[X_i(k) = 1] = p \tag{1}$$

$$P[X_i(k) = 0] = 1 - p, \tag{2}$$

for $0 \leq p \leq 0.5$. It can be shown that

$$E[X_i(k)] = p \tag{3}$$

$$\text{Var}[X_i(k)] = p(1-p). \tag{4}$$

Let $\mathbf{X}_i$ and $\mathbf{X}_j$ be discrete-time binary stochastic processes with the same value of parameter p. Then the correlation coefficient of the random variables $X_i(k)$ and $X_j(k)$ at time instant k is defined as

$$\begin{aligned} c &= \frac{\text{Cov}[X_i(k), X_j(k)]}{\sqrt{\text{Var}[X_i(k)]\text{Var}[X_j(k)]}} \\ &= \frac{E[X_i(k)X_j(k)] - p^2}{p(1-p)}. \end{aligned} \tag{5}$$

From Equation (5), $E[X_i(k)X_j(k)]$ denoted by $R_{ij}(k)$ is given by

$$R_{ij}(k) = p^2 + cp(1-p). \tag{6}$$

Let $k \in \{1, 2, 3, \ldots K\}$, then for stationary ergodic processes, an estimate of R_{ij} which is independent of k is given by

$$\hat{R}_{ij} = \frac{1}{K} \sum_{k=1}^{K} X_i(k)X_j(k). \tag{7}$$

It can easily be seen that

$$\begin{aligned} R_{ij} &= E[\hat{R}_{ij}] \\ &= \frac{1}{K} \sum_{k=1}^{K} E[X_i(k)X_j(k)] \\ &= \begin{cases} p^2 + cp(1-p), & \text{for } i \neq j \\ p, & \text{for } i = j. \end{cases} \end{aligned} \tag{8}$$

Assume that there are N such discrete-time binary processes, of which N_c are correlated. Moreover, let us define $\hat{W}_i = \sum_{j=1}^{N} \hat{R}_{ij}$. From Equation (8), it can be shown that if $\mathbf{X}_i$ belongs to the correlated group with correlation coefficient $c > 0$, then

$$E[\hat{W}_i] = (N-1)p^2 + p + (N_\text{c} - 1)cp(1-p). \tag{9}$$

In contrast, if $\mathbf{X}_i$ belongs to the uncorrelated group, then

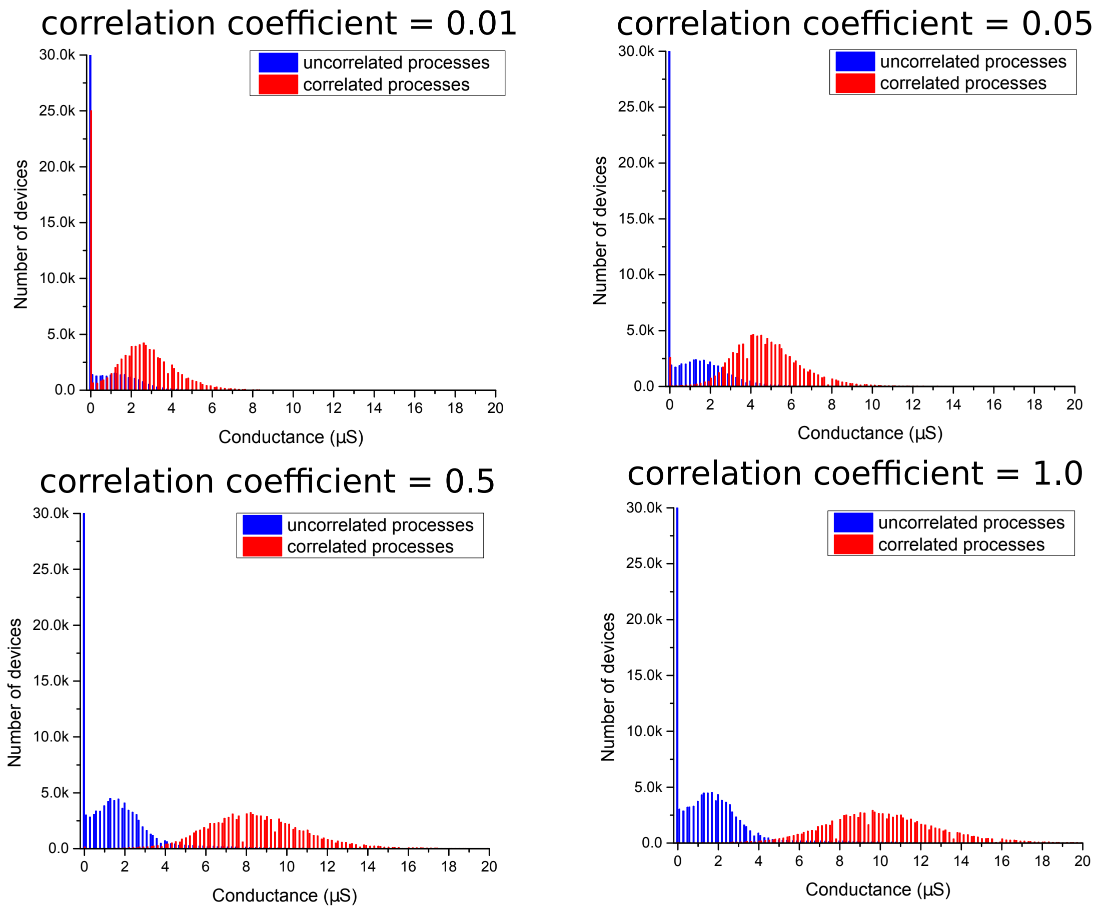
$$E[\hat{W}_i] = (N-1)p^2 + p. \tag{10}$$

It is possible to show that the variance of the estimator $\hat{W}_i$ decreases as the number of time steps K grows:

$$\begin{aligned} \text{Var}[\hat{W}_i] &= E[\hat{W}_i^2] - E[\hat{W}_i]^2 \\ &= \frac{1}{K} \sum_{j=1}^{N} \sum_{j'=1}^{N} \text{Cov}[X_i(k)X_j(k), X_i(k)X_{j'}(k)] \\ &\leq \frac{1}{K} \sum_{j=1}^{N} \sum_{j'=1}^{N} \sqrt{\text{Var}[X_i(k)X_j(k)]\text{Var}[X_i(k)X_{j'}(k)]} \leq \frac{N^2}{4K}. \end{aligned} \tag{11}$$

Hence by monitoring $\hat{W}_i$ in the limit of large K, we can determine which processes are correlated with $c > 0$. Moreover, it can be seen that with increasing c and N_c, it becomes easier to determine whether a process belongs to a correlated group.

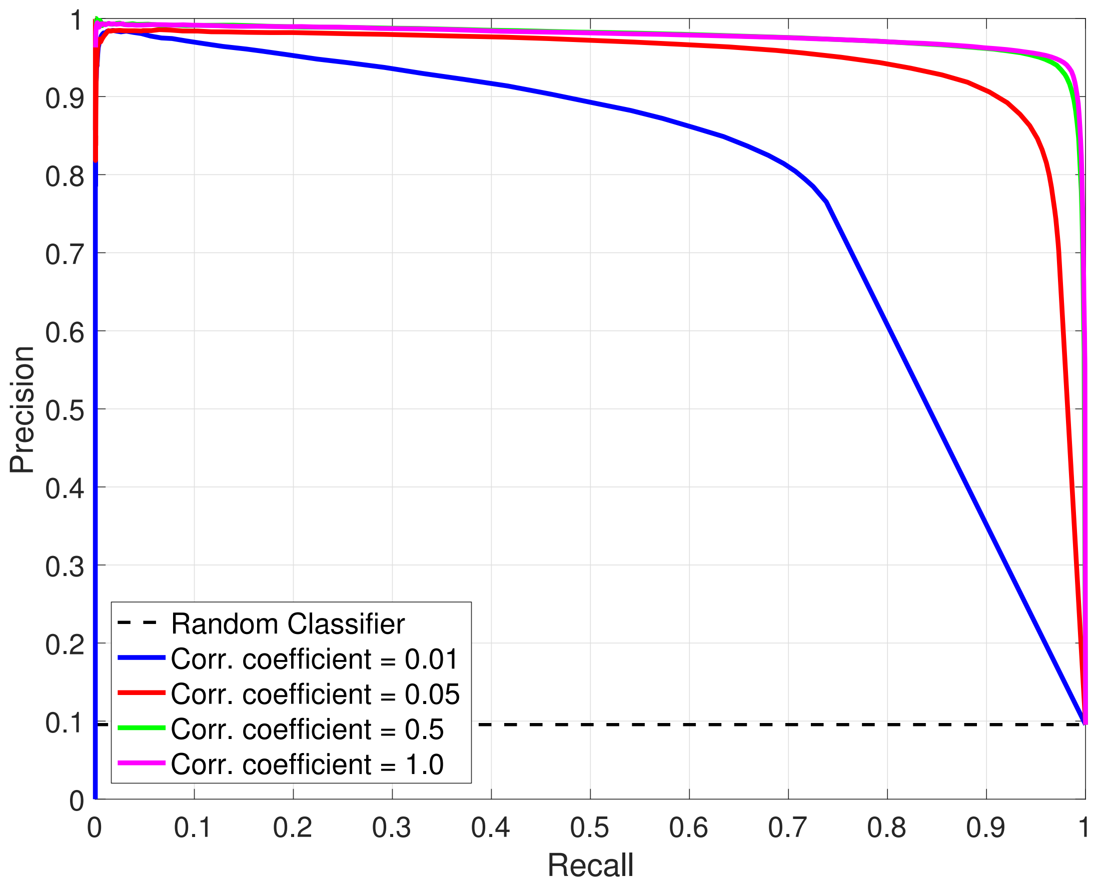
SUPPLEMENTARY NOTE 2: DEPENDENCE ON THE CORRELATION COEFFICIENT



Supplementary Figure 1. **Dependence on the correlation coefficient.** Histograms showing the distribution of device conductance values at the end of the experiment for different values of c. The separation between the correlated and the uncorrelated groups increases with increasing values of c.

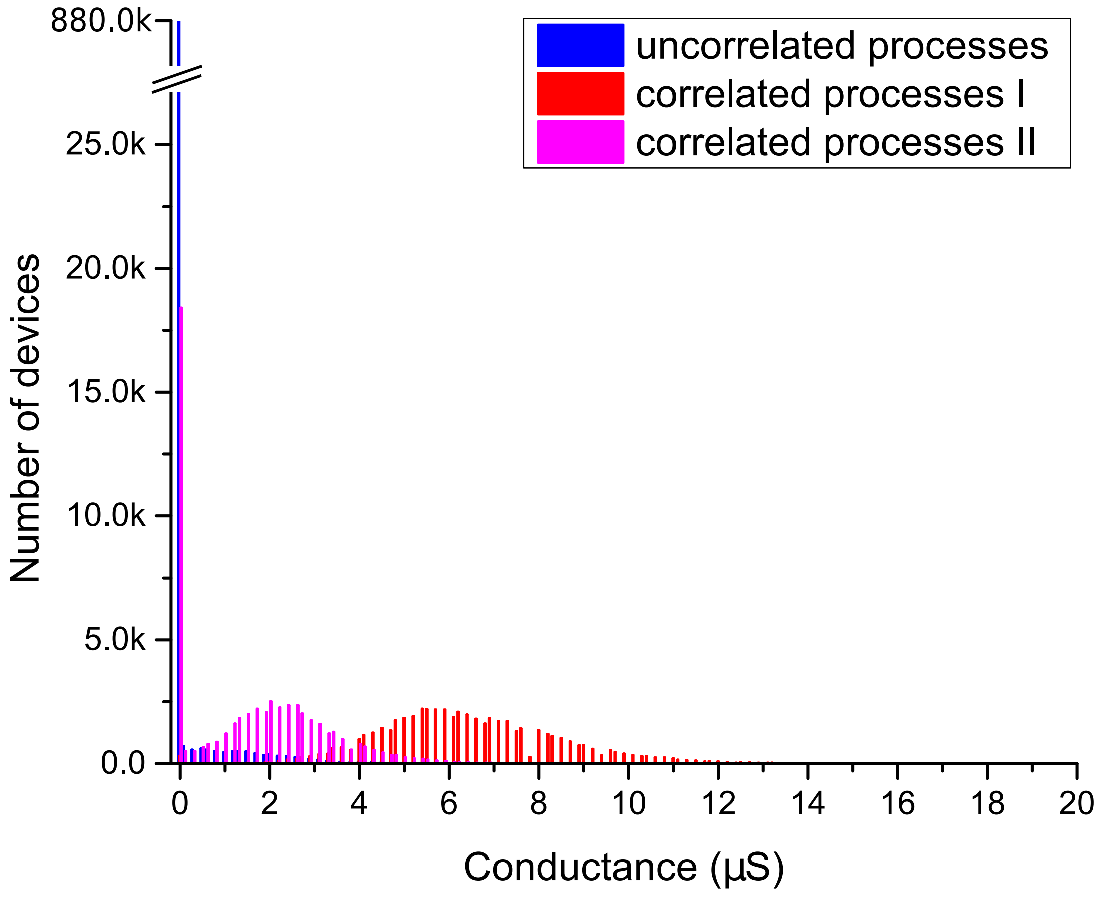
As discussed in Supplementary Note 1, the ability to detect temporal correlations heavily depends on the extent of correlation. For example, one would expect that the task becomes progressively easier with increasing correlation coefficient. Experimental results indeed show this trend. Experiments were performed with 1 million processes in total. Of these, we arbitrarily chose a subset of 95,525 processes to be mutually correlated with an instantaneous correlation coefficient, c. The remaining 904,475 processes are uncorrelated. Supplementary Figure 1 shows distribution of the device conductance values at the end of the experiment for different values of c. It can be seen that with increasing values of c, there is a larger separation between the correlated and uncorrelated groups.

To perform correlation detection, we construct a binary classifier by slicing the histograms of Supplementary Figure 1 according to some threshold, above which processes are labelled correlated and below which processes are labelled uncorrelated. The threshold parameter can be swept across the domain, resulting in an ensemble of different classifiers, each with its own statistical characteristics (e.g., precision and recall). In Supplementary Figure 2, we plot the precision–recall curves for such an ensemble for increasing values of the correlation coefficient, c. We have opted to study the precision–recall curves rather than the corresponding receiver operator characteristic (ROC) curves because it is well established that precision–recall analysis is better suited for measuring the classification performance on imbalanced datasets[1]. As a baseline, we also present the precision–recall curve for the random classifier that simply labels processes as correlated with some arbitrary probability. As expected, we observe an increasingly better quality of classification (i.e., more area under the precision–recall curve) as the correlation coefficient increases. In all cases, the area under the curve is significantly larger than that of the baseline, random classifier.



Supplementary Figure 2. Precision–recall curves as a function of correlation coefficient.

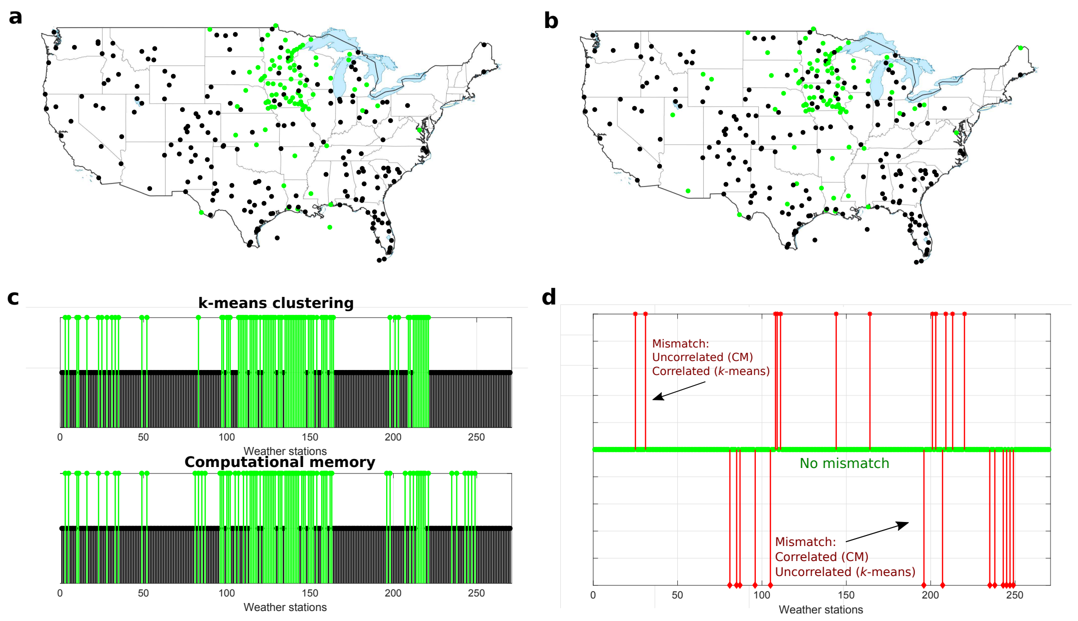
SUPPLEMENTARY NOTE 3: DETECTING MULTIPLE CORRELATIONS



Supplementary Figure 3. **Detecting multiple correlations.** The distribution of conductance levels after 2455 time steps clearly shows that the algorithm is able to detect multiple correlations.

It is also possible to detect multiple correlations using the computational memory approach. To show this, an experiment is presented with 1 million processes in total. Of these 889,007 are uncorrelated. 56,296 processes are correlated with a correlation coefficient of 0.05, whereas 54,697 processes are correlated with a correlation coefficient of 0.08. The sets of correlated processes are mutually uncorrelated. Each of the processes is assigned to a single PCM device. The resulting distribution of conductance levels after 2455 time steps is shown in Supplementary Figure 3. One can clearly see the difference in the conductance levels associated with the devices assigned to the two different correlated groups.

SUPPLEMENTARY NOTE 4: COMPARISON WITH k-MEANS CLUSTERING ALGORITHM



Supplementary Figure 4. **Comparison with k-means clustering algorithm.** (**a**) The k-means clustering algorithm was used to cluster the weather stations into a correlated and an uncorrelated group. In the resulting map of the United States, the correlated weather stations are denoted in green and the uncorrelated ones in black. (**b**) The computational memory approach was also used to group the weather stations into a correlated and an uncorrelated group. In the resulting map of the United States, the correlated weather stations are denoted in green and the uncorrelated ones in black. (**c**) Out of the 270 weather stations, there was agreement between the two approaches for 245 weather stations. (**d**) The computational memory approach classified 12 weather stations as uncorrelated that had been marked correlated by the k-means clustering approach. Similarly, the computational memory approach classified 13 weather stations as correlated that had been marked uncorrelated by the k-means clustering approach.

For the experiment using the weather data, another way to classify the processes based on their temporal correlation is via the k-means clustering algorithm. Consider a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation $x_i \in \mathbb{R}^d$, and a k-fold partitioning of the same observations $\mathcal{S} = (S_1, S_2, \ldots, S_n)$. Each individual partition S_i is a set referred to as a cluster. The optimal solution to the k-means clustering problem is given by the partitioning $\hat{\mathcal{S}}$ that satisfies the following:

$$\hat{S} = \underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2, \tag{12}$$

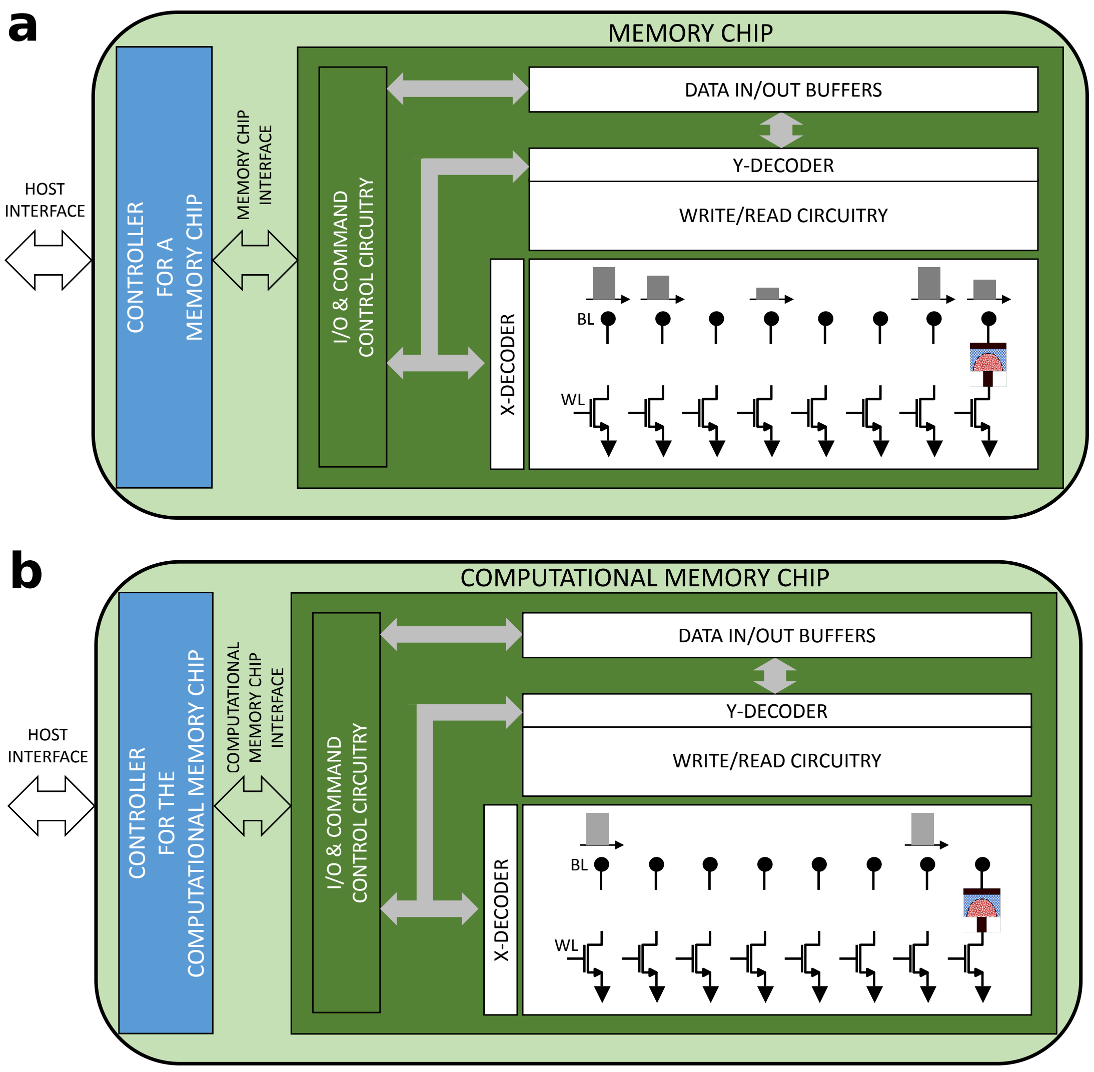
where $\mu_i \in \mathbb{R}^d$ denotes the cluster mean (or centroid). While the k-means problem has been proved to be NP-hard[2], heuristic techniques such as Lloyd's algorithm[3] are often used to find approximate solutions. Lloyd's algorithm begins by initializing each cluster centroid by selecting one of the n observations uniformly at random. It then proceeds to pass through the n observations, assigning each observation to the cluster that is closest in the sense of Euclidean distance from the observation vector to the cluster centroid. At this end of this pass, each cluster centroid is updated by computing the mean of all of the observations that were assigned to it in the preceding pass. This process is repeated for a fixed number of iterations, T, or until some convergence criterion is met. As the algorithm can converge to a local optimum, it is common practice to perform a number of outer iterations using different initial centroids to see whether a better solution can be obtained. The complexity of a single outer iteration of Lloyd's algorithm has complexity $\mathcal{O}(ndkT)$. For structured data, one typically finds that the algorithm converges with relatively few inner iterations. However, it has been shown that the algorithm can need $T = 2^{\Omega(\sqrt{n})}$ iterations in the worst case and thus exhibits super-polynomial complexity[4].

Here, we compare the k-means clustering approach with the computational memory approach. The k-means clustering algorithm was used to cluster the weather stations into two groups based on their Euclidean distance in the $\mathbb{R}^K$ space. The cluster of the weather stations with lesser mean distance from each other was denoted the correlated group, and the other cluster was denoted the uncorrelated group. Supplementary Figure 4(a) shows the map of the United States with the correlated weather stations denoted in green and the uncorrelated ones in black.

The computational memory approach was also used to group the weather stations into a correlated and an uncorrelated group. As described in the main text, we used 4 devices to interface with a single weather station. If the mean conductance value exceeds 2 μS, then that weather station was considered to belong to the correlated group. The resulting map of the United States with the correlated weather stations as detected by the computational memory approach is shown in Supplementary Figure 4(b).

Out of the 270 weather stations, there was agreement between the two approaches for 245 weather stations (Supplementary Figure 4(c)). The computational memory approach classified 12 weather stations as uncorrelated that had been marked correlated by the k-means clustering approach. Similarly, the computational memory approach classified 13 weather stations as correlated that had been marked uncorrelated by the k-means clustering approach (Supplementary Figure 4(d)). Given the simplicity of the computational memory approach, it is remarkable that it can achieve this level of similarity with such a sophisticated classification algorithm. Also note that, given the difference between the two algorithms (covariance matrix-based and k-means clustering), calculations show that the expectation was to get an agreement only for 251 weather stations. These experiments also revealed the advantages of having multiple devices interfacing to a single random process. For example, if we had used only one device per weather station, our accuracies would have been lower and the agreement with the k-means approach would have been obtained only for 229 weather stations.

SUPPLEMENTARY NOTE 5: A HYPOTHETICAL COMPUTATIONAL PHASE-CHANGE MEMORY UNIT

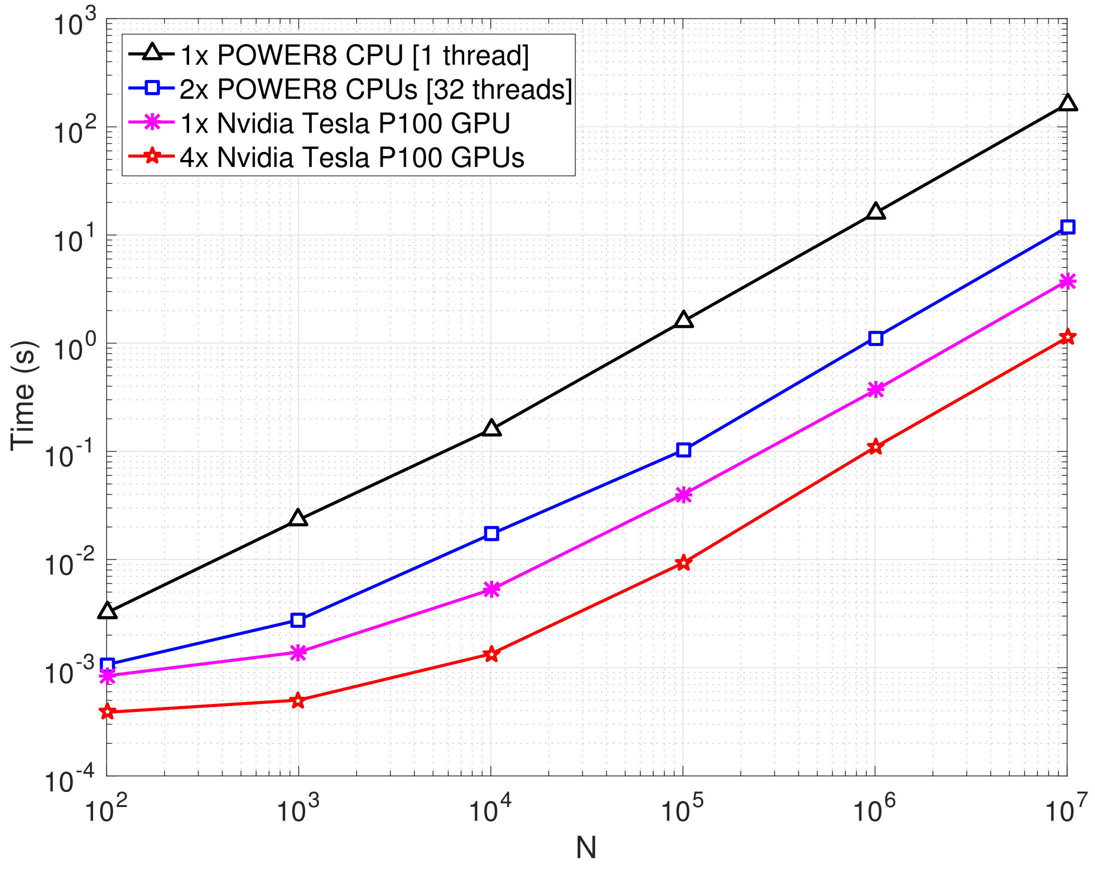

a
MEMORY CHIP
HOST INTERFACE
CONTROLLER FOR A MEMORY CHIP
MEMORY CHIP INTERFACE
I/O & COMMAND CONTROL CIRCUITRY
DATA IN/OUT BUFFERS
Y-DECODER
WRITE/READ CIRCUITRY
X-DECODER
BL
WL
b
COMPUTATIONAL MEMORY CHIP
HOST INTERFACE
CONTROLLER FOR THE COMPUTATIONAL MEMORY CHIP
COMPUTATIONAL MEMORY CHIP INTERFACE
I/O & COMMAND CONTROL CIRCUITRY
DATA IN/OUT BUFFERS
Y-DECODER
WRITE/READ CIRCUITRY
X-DECODER
BL
WL

Supplementary Figure 5. **Schematic representation of the various building blocks.** (**a**) Conventional PCM-based memory unit and (**b**) hypothetical computational phase-change memory unit. BL stands for Bit Line and WL for Word Line.

Here we present some details as to how a hypothetical computational phase-change memory unit would look. In particular, we would like to highlight the similarity of such a computational memory chip with a conventional PCM-based memory unit. Supplementary Figure 5(a) presents a schematic representation of the various building blocks of the conventional memory unit. The main constituents are a memory controller chip and a memory chip. The memory unit typically interfaces to a host via a host interface. During the write operation, the controller may apply some type of encoding to the data before sending the corresponding write commands to the memory based on the Input/Output (I/O) interface of the chip. During the read operation, the data received from the memory chip is decoded and error correction may be performed before the data is passed to the host. The memory chip comprises the I/O and command control circuitry, the I/O data buffers, the address-decoding circuitry, the read/write circuitry and the memory array. During the write operation, the data is loaded to the input buffers and the write circuity will be invoked to program the memory devices in parallel or sequentially. The devices can be addressed directly or in an incremental fashion (Bit Lines and World Lines can be used to access an individual memory cell or a group of memory cells). During the read operation, the memory cells are read in parallel or sequentially, and the digitized information is passed on to the memory controller.

Supplementary Figure 5(b) presents a schematic representation of the various building blocks of a potential computational memory unit. Here again, the main constituents are a memory controller chip and a memory chip, and the chip can interface to a host via an appropriate host interface. For example, the event-based data streams arrive at the computational memory chip via this interface. Based on this, the instantaneous collective momentum can be calculated in the controller. A digital bit sequence that indicates the programming current or pulse width corresponding to the collective momentum is passed to the memory chip along with the addressing information of the memory devices that need to be programmed. The bit sequence and addressing information are formatted according to some encoding scheme determined by the memory chip interface. The devices that need to be programmed will be programmed with the same programming conditions, whereas the devices that need not be programmed can be indicated using a designated bit sequence.

SUPPLEMENTARY NOTE 6: COMPARISON WITH IMPLEMENTATIONS WITH CPUS AND GPUS



Supplementary Figure 6. Time required to compute the coefficients W_i for $K = 10^4$ for increasing number of processes N.

In this note we consider how one can efficiently compute the coefficients W_i using state-of-the-art CPU and GPU hardware and how such implementations compare with the proposed approach that uses computational memory. For a set for binary processes $X_i(k)$, the coefficient W_i can be computed as follows:

$$W_i = \sum_{k=1}^{K} X_i(k) \sum_{j=1}^{N} X_j(k) = \sum_{k=1}^{K} X_i(k) M(k) \tag{13}$$

The samples from all N processes at a given time step k are stored contiguously in memory, and a single bit is used to represent each sample $X_i(k)$, thus the samples from 32 different processes are packed inside one 4-byte unsigned integer data type. The total memory required to store the data at all K time steps is given by $KN/8$ bytes: for $K = 10^4$ and $N = 10^6$ this amounts to around 1.2 GB of data. The CPU-based implementation begins by initializing the estimates $W_i = 0$, for $i = 1, 2, \dots, N$. The implementation then proceeds to iterate over the time steps $k = 1, 2, \dots, K$. At each time step, we first compute the collective momentum $M(k)$: a 4-byte floating point number resulting from the summation of all bits $X_i(k)$ for $i = 1, 2, \dots N$. To perform this addition operation, each bit $X_i(k)$ must first be extracted from the corresponding unsigned integer data type, and cast to a floating point number. Compiler flags were set to ensure that the additions are effectively vectorized, thus making full use of the arithmetic resources provided by the CPU. Once the collective momentum has been computed, we then perform a second pass through the processes, again casting each $X_i(k)$ to a floating point number, multiplying by $M(k)$, and adding the result to the current value of W_i. In this manner, the coefficients W_i are effectively accumulated in memory.

To harness the multi-core architecture of modern CPUs, the computation defined in equation (13) can easily be divided up amongst T threads. The time steps $k = 1, 2, \dots, K$ are divided into T equal groups, and each thread is responsible for computing the part of the summation (13) corresponding to one of such groups. Once all threads have finished working, the partial results are reduced to obtain the final value for the coefficients W_i.

In a similar manner, the computation can be mapped to GPU hardware by launching a kernel function consisting of K thread blocks. Each thread block is responsible for the computation of a single summand in (13) and for the computation of every summand in (13) (i.e., one thread block for every time step). The GPU schedules the thread blocks for execution on its streaming multi-processors as it sees fit. Each thread block computes the collective momentum $M(k)$ in a multi-threaded manner: each individual thread is responsible for only a part of the necessary summation over all processes, the partial results are stored in shared memory, and finally the results are reduced to obtain the resulting value for $M(k)$. The thread block then proceeds to pass over the processes again, this time multiplying each sample $X_i(k)$ by the momentum $M(k)$ and then using an atomic addition operation to accumulate the resulting values for W_i in the main memory of the GPU. It is possible to further distribute the computation across multiple GPUs by partitioning the time steps $k = 1, 2, \ldots, K$ into distinct groups, allowing each GPU to compute a partial result for the W_i for the set of time steps assigned to it, and then collecting and reducing the results from all GPUs at the end. As on the CPU, the binary data for the processes are packed into 4-byte unsigned integer data types, and all arithmetic is performed using 4-byte floating point data types. Thus, as before, each bit $X_i(k)$ must be unpacked and cast to a floating point number before it is used arithmetically.

To quantify the time required for such a calculation using state-of-the-art computing hardware, we measured the performance of various implementations that can be executed on an IBM* Power* System S822LC system. This system has 2 POWER8* CPUs (each comprising 10 cores) and 4 Nvidia Tesla P100 GPUs (attached using the NVLink interface). We measured the performance of a CPU-based implementation using only a single thread as well as of an implementation that uses 32 threads spread across the 2 CPUs. We also studied the performance of an implementation that uses a single GPU, as well as one that uses all 4 GPUs. The results are shown in Supplementary Figure 6 for $K = 10^4$ and increasing values of N. We observe linear scaling with N in all cases and a significant improvement from using the multi-threaded and GPU-based implementations. For instance, by using 4 GPUs, it was possible to compute all coefficients W_i for $N = 10^7$ processes in a single second.

To gain more insight into the computational complexity of the GPU-based correlation detector, in Supplementary Table 1 we present a profiling of the computational kernel that was obtained using the nvprof tool provided by NVIDIA. We present data for the experiment using a single P100 GPU and $N = 10^7$ processes with $K = 10^4$ time steps. We compare the resulting profiles for the case where the computational kernel only computes the collective momentum, with the numbers obtained for the full computational kernel as described above. Firstly, we note that while the kernel takes only 80 ms to compute the collective momentum, it requires 3.9 s to perform the full calculation (i.e., only 2% of the execution time is spent computing the collective momentum). Next, we note that when moving from the momentum-only kernel to the full kernel, most of the instruction counts either stay relatively constant or increase by a factor of two. There are three noticeable exceptions: the full kernel requires 102 billion floating point multiplications (the momentum computation requires zero), 26 billion global load transactions (only 800 million were necessary for the momentum computation) and 204 billion L2 transactions related to atomic requests (again, no atomic requests were necessary for the momentum computation). This huge increase in the number of L2 requests due to atomic operations leads to saturation of the GPU device's memory bandwidth and is thus the performance bottleneck for the kernel (this is confirmed by running the bottleneck analysis provided by NVIDIA's visual profiling tool).

To provide a point of comparison with the proposed approach that uses computational memory, we note that the time required to compute W_i for $N = 10^7$ processes will be dominated either by the time required to write to the PCM cells, t_1, or the time required to compute the collective momentum $M(k)$ on the memory controller, t_2. As all PCM cells can be programmed in parallel, the time spent writing to PCM is given by $t_1 = Kt_{\text{PCM}}$ where t_{PCM} is the PCM write latency. Assuming a write latency of 100 ns, we conclude that, for $K = 10^4$, the time spent writing to the PCM will be around 1 ms. Note that owing to the sparsity of the processes, only a fraction of the $N = 10^7$ devices receive a SET pulse, and hence the devices could be programmed in parallel. Even otherwise, it is possible to slightly offset the programming pulses in time without a significant penalty. The time t_2 is determined by how quickly one can count the number of ones in the binary sequence $(X_1(k), X_2(k), \ldots, X_N(k))$ on the memory controller. Assuming that one implements this addition using a tree structure in either an FPGA or an ASIC device, one can compute the collective momentum at a given time step in $L = \log_2(N)$ clock cycles. Assuming a relatively low clock frequency of 50 MHz, this would correspond to around 0.5 μs per time step for $N = 10^7$, and thus the total time spent computing the collective momentum for $K = 10^4$ steps would be approximately 5 ms. Hence, we conclude that the computation of the coefficients W_i can be accelerated by approximately a factor of 200 (relative to the implementation using four P100 GPUs) by using computational memory and an FPGA or ASIC-based memory controller.

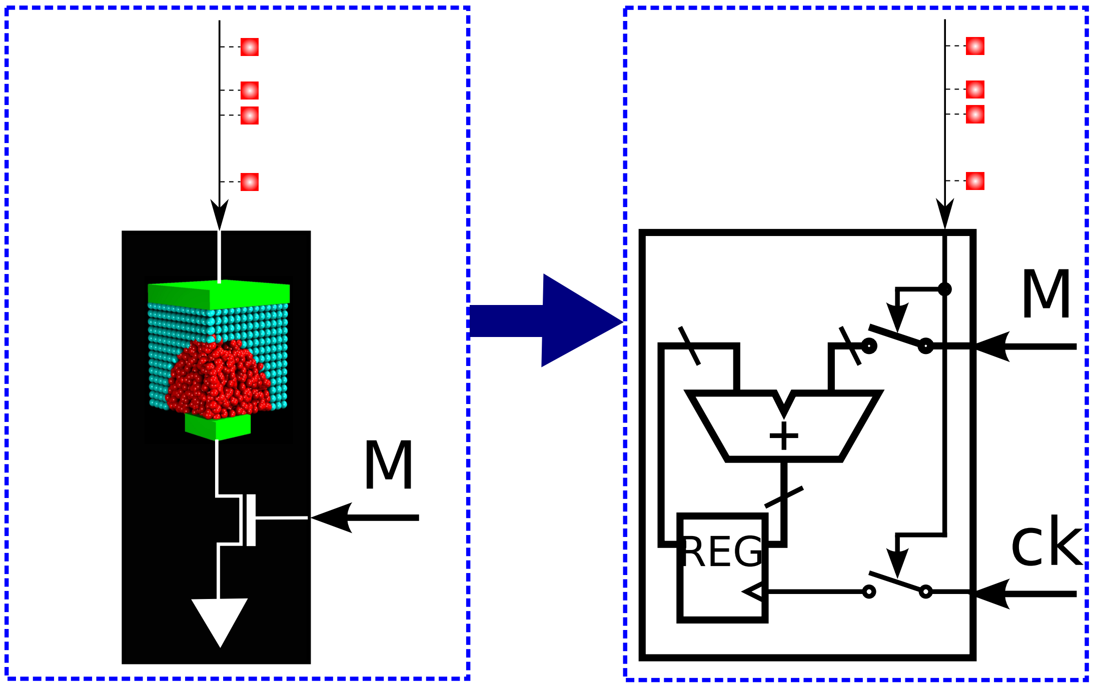
We can also make a comparative study between the GPU implementation and computational memory with respect to energy to solution. From Supplementary Table 1, we expect the energy to solution for the full kernel to be 148.2 J (The difference between average power and idle power multiplied by the execution time). The energy consumed for just the momentum calculation is 1.68 J. The energy consumption associated with the computational memory unit is difficult to quantify given that we do not have a complete system yet. However, we can make an estimate based on the existing devices and their energy consumption. The energy consumed per device is approximately 580 pJ for one RESET operation and approximately 1.5 pJ for one SET operation. Hence the total energy consumed in the devices for the experiment corresponding to Figure 5 in the manuscript is 58.7 mJ. If we extrapolate this result for the number of processes being $N = 10^7$ and retaining the number of time instances as $K = 10^4$, then the energy consumed is estimated to be approx. 587 mJ. These calculations are based on devices fabricated in the 90 nm technology node. The RESET energy, which dominates the overall energy consumption, will reduce substantially if we consider the state-of-the-art PCM devices fabricated in lower technology nodes. However, this estimate does not consider the overhead associated with read/write circuitry, data converters, control and command circuitry etc. But, even if we assume a substantial

Metric	**Momentum only**	**Full kernel**
Integer Instructions	128 B	272 B
Bit-Convert Instructions	102 B	205 B
Control-Flow Instructions	3.2 B	6.4 B
Floating Point Operations (Single Precision Add)	102 B	102 B
Floating Point Operations (Single Precision Mul)	0	**102 B**
Shared Load Transactions	100 k	110 k
Shared Store Transactions	60 k	60 k
Global Load Transactions	800 M	**26 B**
L2 Transactions (Texture Reads)	400 M	800 M
L2 Transactions (Atomic Requests)	0	**204 B**
L2 Throughput (Atomic Requests)	0 GB/s	**788.27 GB/s**
Kernel Execution Time	80 ms	3.9 s
Power (Idle)	35 W	35 W
Power (Avg.)	56 W	73 W

Supplementary Table 1. Profiling of the computational kernel for $K = 10^4$ and $N = 10^7$ on a single P100 GPU.

overhead, the energy consumed for just the momentum calculation is expected to dominate the overall energy consumption, assuming that the momentum is computed using GPU. In this case, we expect an almost two orders of magnitude improvement in the overall energy consumption. If the momentum computation is realized in an FPGA or ASIC-based controller as proposed earlier, one may expect even more substantial improvements.

SUPPLEMENTARY NOTE 7: EMULATION OF COMPUTATIONAL MEMORY IN CMOS

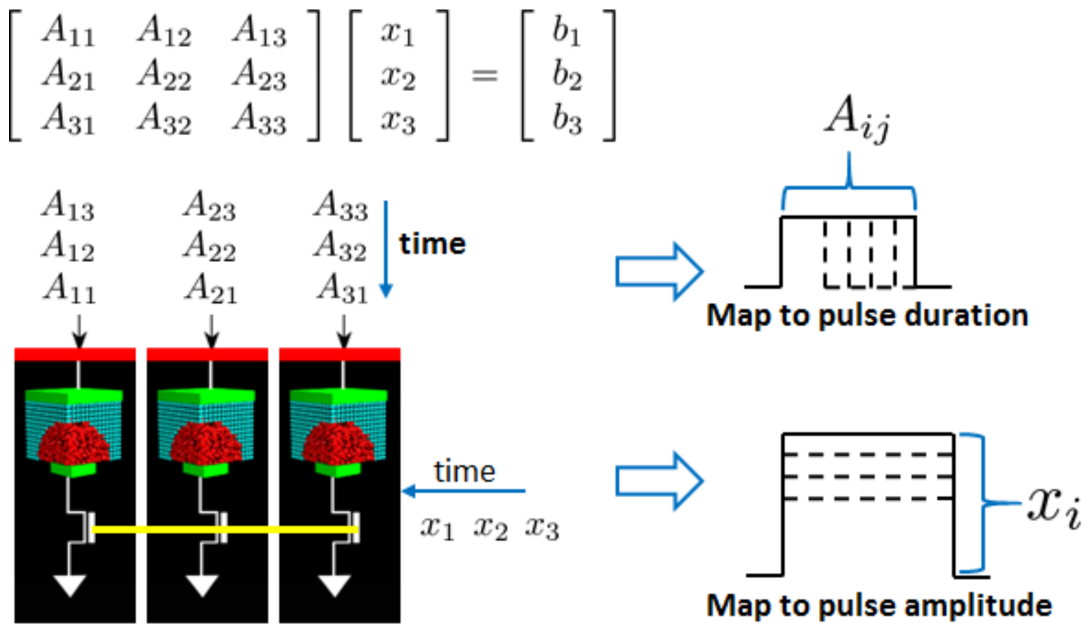


Supplementary Figure 7. **Emulation of the accumulation behavior using CMOS technology.** One could emulate the accumulative behavior of the PCM using adders and registers. However, such a device is volatile and has a much larger footprint.

One could make an argument for the design of an application-specific chip where the accumulative behavior of PCM is emulated using complementary metal-oxide semiconductor (CMOS) technology using adders and registers (see Supplementary Figure 7). The required resolution of the adder and register is $\log_2(N \times K)$. For 10^6 parallel processes and 4000 time steps, 32 bits are sufficient, but precision could be reduced to save area and power. Comparing at the same technology node, a digital CMOS adder would be on the order of a few 100 transistors, which would require an area that is two orders of magnitude larger than a PCM device with a single access transistor. The much smaller area of a PCM device than that of the CMOS equivalent circuit allows a more than two orders of magnitude higher density on chip. This enables significantly larger problems to be solved in a PCM chip without moving data between the PCM chip and external memory.

Furthermore, the non-volatility of PCM devices enables efficient low-power devices, in which the leakage of CMOS would dominate the dynamic power because of the low utilization rate. We can exploit the enormous storage capacity and non-volatility of PCM devices. The non-volatility makes them particularly attractive for very slow processes. Note that the comparison on current PCM technology discounts the fact that the energy figures corresponding to the computational memory will improve by orders of magnitude as we scale to smaller dimensions[5] and faster programming speeds[6]. We do not foresee such a scaling scenario for CMOS technology[7].

SUPPLEMENTARY NOTE 8: OTHER COMPUTATIONAL PRIMITIVES USING CRYSTALLIZATION DYNAMICS

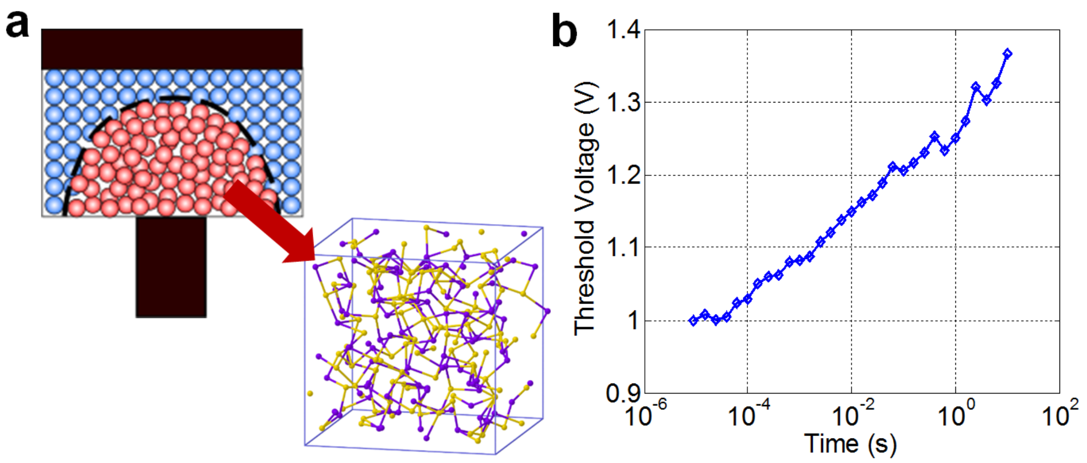


$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$
A_{13} A_{12} A_{11}
A_{23} A_{22} A_{21}
A_{33} A_{32} A_{31}
time
time
x_1 x_2 x_3
A_{ij}
Map to pulse duration
x_i
Map to pulse amplitude

Supplementary Figure 8. **Applications beyond correlation detection.** To multiply a matrix, A, with a vector, x, one PCM device is assigned to each row of A. The elements of A and x are mapped to the pulse characteristics (duration and amplitude, respectively) of the programming pulse. Owing to crystallization dynamics, the result gets stored as the conductance values of the devices.

Temporal correlation detection is one of the computational primitives realizable using crystallization dynamics. Here, we show how we can exploit the crystallization dynamics to perform matrix-vector multiplications (Supplementary Figure 8). Matrix-vector multiplications, such as the one where we would like to multiply a matrix, A, with a vector, x, to obtain the vector, b, arise in a wide range of engineering and scientific applications. As shown in Supplementary Figure 8, one PCM device is assigned to each row of matrix A. Crystallizing pulses of duration proportional to the elements of the row of the matrix are then applied sequentially in time to the corresponding PCM device. The corresponding programming pulse amplitude (programming current) is modulated in accordance with the elements of the vector, x. For example, the first pulse applied to the PCM device corresponding to the first row of the matrix would have an amplitude proportional to x_1 and a duration proportional to A_{11}. Because of the crystallization dynamics, the extent of crystallization and hence the conductance of the device will be proportional to the magnitude of the corresponding element of vector b. The result of the calculation will be "imprinted" as resistance or conductance value of the devices. This is yet another clear illustration of collocated computing and storage. Note that such an approach could scale remarkably well as it would only need N devices for an $N \times N$ matrix.

SUPPLEMENTARY NOTE 9: COMPUTATIONAL MEMORY USING THE DYNAMICS OF STRUCTURAL RELAXATION IN ADDITION TO CRYSTALLIZATION DYNAMICS



Supplementary Figure 9. Structural relaxation and its impact on electrical transport. (**a**) When an amorphous phase is formed by the melt-quench process, the atomic configurations are frozen into an unstable glass state, which relaxes over time to a more stable "ideal" glass state. This is known as structural relaxation and is well captured by a collective relaxation model[11]. (**b**) Experimental data shows that at constant temperature, the structural relaxation manifests itself as a linear increase in the threshold switching voltage, V_{th}, with the log of time.

Phase-change devices exhibit a rich dynamic behavior captured by a feedback interconnection of electrical, thermal and structural dynamics. Besides the crystallization dynamics, we can also exploit other types of dynamics for computational memory. One such dynamic behavior is structural relaxation. When an amorphous phase is formed by the melt-quench process, the atoms are frozen into an unstable glass state (Supplementary Figure 9(**a**)). Subsequently, these atomic configurations relax over time to a more stable "ideal" glass state[8]. The exact nature of this structural relaxation and the nature of the "ideal" glass state are being actively researched[9,10]. We recently showed that the dynamics of structural relaxation can be modelled accurately via a collective relaxation model[11]. The amorphous structure collectively rearranges, whereby every local configuration is changed repeatedly to achieve an overall lower energy state. The relaxation proceeds in a sequence of transitions between neighboring states. The closer to the equilibrium the systems is, the higher is the barrier for subsequent relaxation. If $\Sigma(t) \in [0\ 1]$ denotes an order parameter that indicates the distance of the amorphous state from the ideal glass state at any point in time, t, then $\Sigma(t)$ evolves according to

$$\frac{d\Sigma(t)}{dt} = -\kappa e^{\left(-\frac{E_s}{k_{\mathrm{B}}T(t)}\right)} e^{\left(\frac{\Sigma(t)E_s}{k_{\mathrm{B}}T(t)}\right)}. \tag{14}$$

It can be shown that at constant temperature, $\Sigma(t)$ varies linearly with the log of time.

Experimental measurements such as the one shown in Supplementary Figure 9(**b**) indicate that the threshold-switching field depends on $\Sigma(t)$ as given by

$$E_{\text{th}}(t) = E^0_{\text{th}} + \beta(1 - \Sigma(t)). \tag{15}$$

Hence the threshold switching voltage evolves according to

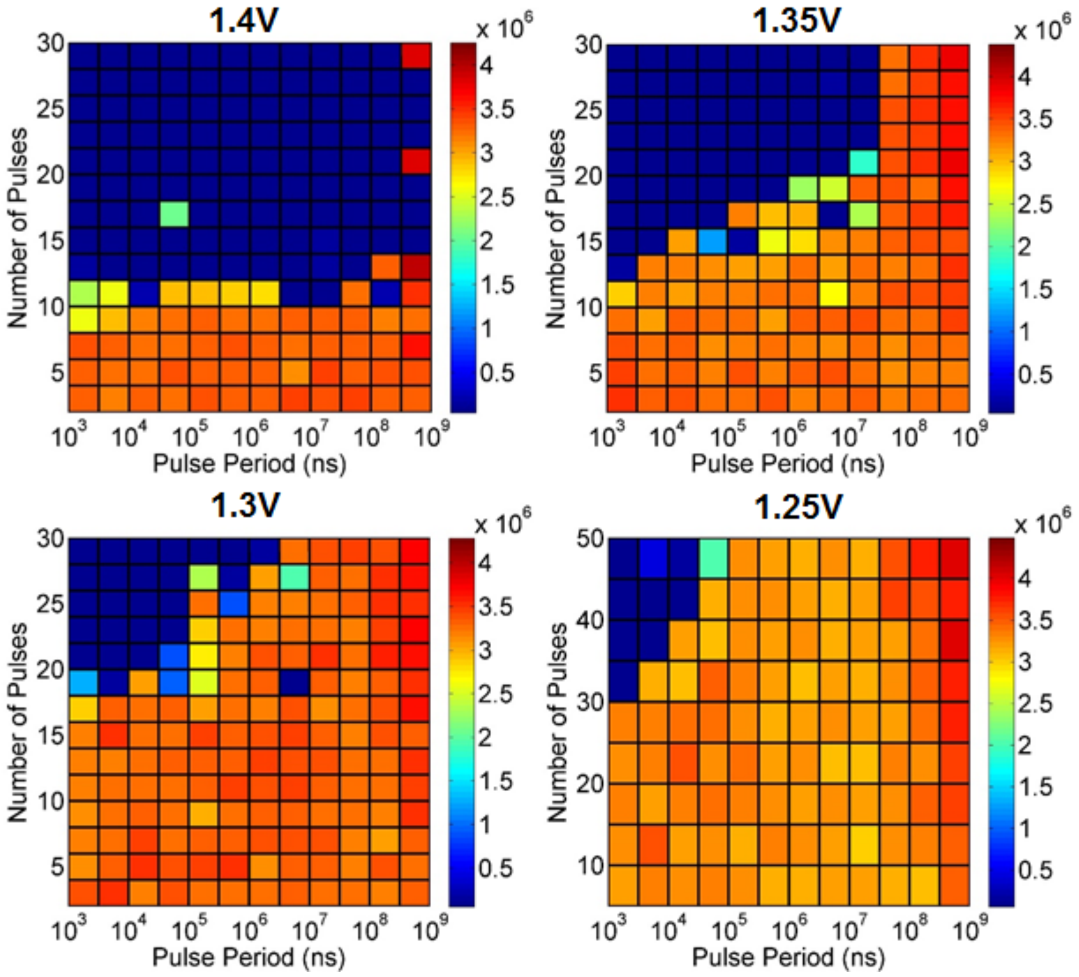
$$V_{\text{th}}(t) = E_{\text{th}}(t)u_{\text{a}}. \tag{16}$$

Therefore, when programming a PCM device, the applied voltage, V, has to be greater than $V_{\text{th}}(t)$ to pass sufficient current through the PCM device and to induce crystallization. This means that the temperature at the amorphous–crystalline interface, T_{int}, is given by

$$T_{\text{int}} = R_{\text{th}}(u_{\text{a}})\frac{V^2}{R_{\text{ON}}} + T_{\text{amb}},\ \text{if } V > V_{\text{th}}(t) \tag{17}$$

$$= T_{\text{amb}}\ \text{If } V \le V_{\text{th}}(t), \tag{18}$$

where R_{ON} is the high-field ON resistance of the PCM device (largely independent of the amorphous thickness). When the applied voltage, V, is lower than $V_{\text{th}}(t)$, there is no significant Joule heating possible and T_{int} will remain close to T_{amb}. So if we



Supplementary Figure 10. **Influence of structural dynamics on the accumulation behavior**. Experimental data showing the dependence of the number of pulses to crystallize on the pulse period. The number of pulses required to reach a threshold conductance level is obtained as a function of the pulse period. When the crystallizing pulse amplitude is large compared with $V_{\text{th}}(t)$ (e.g., 1.4 V), the number of pulses to crystallize is mostly independent of the pulse period. But when the crystallizing pulses have amplitudes comparable to $V_{\text{th}}(t)$, there is a significant dependence on the pulse period. For example when the crystallizing pulse amplitude is 1.25 V, if the pulse period is greater than 10^5 ns, then it is not possible to fully crystallize the phase-change material.

want to exploit the dynamics of structural relaxation in addition to the crystallization dynamics, the key idea is to operate with programming voltages close to $V_{\text{th}}(t)$.

An experimental demonstration of this concept is presented in Supplementary Figure 10. Crystallizing pulses of 50 ns duration are applied to a phase-change device. The number of pulses required to reach a threshold conductance level is obtained. This experiment is repeated by increasing the period of the pulse sequence (or decreasing the time duration between two consecutive pulses). If the voltage of the crystallizing pulse is sufficiently large compared with the threshold switching voltage (e.g., $V = 1.4$ V), then the number of pulses required for crystallization is almost independent of the pulse period. However, it can be seen that as V becomes smaller, there will be a strong dependence on the pulse period. A larger weighting period between the consecutive application of two pulses could result in substantial structural relaxation such that the applied voltage becomes smaller than the threshold switching voltage. This behavior arising from the dynamics of structural relaxation can be used to discriminate between high-rate and low-rate input processes, such as in rate-coded processes, in addition to the ability to detect temporal correlations.

*IBM, Power, and POWER8 are trademarks of the of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product or service names may be trademarks or service marks of IBM or other companies.

SUPPLEMENTARY REFERENCES

[1]Davis, J. & Goadrich, M. The relationship between Precision–Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine learning*, 233–240. ACM, (2006).

[2]Aloise, D., Deshpande, A., Hansen, P., & Popat, P. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning* **75**(2), 245–248 (2009).

[3]Lloyd, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory* **28**(2), 129–137 Mar (1982).

[4]Arthur, D. & Vassilvitskii, S. How slow is the k-means method? In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, 144–153 (ACM, New York, NY, USA, 2006).

[5]Xiong, F., Liao, A. D., Estrada, D., & Pop, E. Low-power switching of phase-change materials with carbon nanotube electrodes. *Science* **332**(6029), 568–570 (2011).

[6]Loke, D., Lee, T., Wang, W., Shi, L., Zhao, R., Yeo, Y., Chong, T., & Elliott, S. Breaking the speed limits of phase-change memory. *Science* **336**(6088), 1566–1569 (2012).

[7]Horowitz, M. Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 10–14. IEEE, (2014).

[8]Ielmini, D., Lavizzari, S., Sharma, D., & Lacaita, A. Temperature acceleration of structural relaxation in amorphous $Ge_2Sb_2Te_5$. *Applied Physics Letters* **92**(19), 193511 (2008).

[9]Raty, J. Y., Zhang, W., Luckas, J., Chen, C., Mazzarello, R., Bichara, C., & Wuttig, M. Aging mechanisms in amorphous phase-change materials. *Nature Communications* **6** (2015).

[10]Zipoli, F., Krebs, D., & Curioni, A. Structural origin of resistance drift in amorphous GeTe. *Physical Review B* **93**(11), 115201 (2016).

[11]Sebastian, A., Krebs, D., Le Gallo, M., Pozidis, H.,& Eleftheriou, E. A collective relaxation model for resistance drift in phase change memory cells. In *IEEE International Reliability Physics Symposium (IRPS)*, MY–5. IEEE, (2015).