JOURNAL OF APPLIED PHYSICS **124**, 152135 (2018)

A phase-change memory model for neuromorphic computing

S. R. Nandakumar,[1,2] Manuel Le Gallo,[1] Irem Boybat,[1,3] Bipin Rajendran,[2,a)] Abu Sebastian,[1,b)] and Evangelos Eleftheriou[1]
[1]*IBM Research—Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland*
[2]*New Jersey Institute of Technology, Newark, New Jersey 07102, USA*
[3]*Ecole Polytechnique Federale de Lausanne, 1015 Lausanne, Switzerland*

(Received 31 May 2018; accepted 1 October 2018; published online 17 October 2018)

Phase-change memory (PCM) is an emerging non-volatile memory technology that is based on the reversible and rapid phase transition between the amorphous and crystalline phases of certain phase-change materials. The ability to alter the conductance levels in a controllable way makes PCM devices particularly well-suited for synaptic realizations in neuromorphic computing. A key attribute that enables this application is the progressive crystallization of the phase-change material and subsequent increase in device conductance by the successive application of appropriate electrical pulses. There is significant inter- and intra-device randomness associated with this cumulative conductance evolution, and it is essential to develop a statistical model to capture this. PCM also exhibits a temporal evolution of the conductance values (drift), which could also influence applications in neuromorphic computing. In this paper, we have developed a statistical model that describes both the cumulative conductance evolution and conductance drift. This model is based on extensive characterization work on 10 000 memory devices. Finally, the model is used to simulate the supervised training of both spiking and non-spiking artificial neuronal networks. *Published by AIP Publishing.* https://doi.org/10.1063/1.5042408

I. INTRODUCTION

Phase-change memory (PCM) is arguably the most advanced emerging non-volatile memory technology.[1] PCM is based on the property of certain materials, such as $Ge_2Sb_2Te_5$, that exhibit a significant difference in resistivity depending on whether they are in the ordered crystalline phase or the disordered amorphous phase. In a PCM device, a tiny volume of such a material is sandwiched between two metal electrodes. A typical device structure is shown in the cross-sectional TEM image in Fig. 1(a). By the application of suitable electrical pulses and subsequent Joule heating, it is possible to reversibly alter the phase-configuration of the material within the device. Pulses that result in an increase in the size of the amorphous region are typically referred to as RESET pulses. In this case, the application of the pulse results in melting of a critical volume of the material which is then rapidly quenched to induce glass transition. The pulses that reduce the size of the amorphous region are referred to as SET pulses. Here, the temperature reached within the device is favorable for crystallization [see Fig. 1(b)].[2] Typically, the SET pulses that induce partial crystallization of the material are referred to as partial SET pulses and all these pulses are collectively referred to as programming pulses.

The electrical resistance/conductance of the device will depend on the resulting phase-configuration. In fact, it is possible to achieve a continuum of resistance values in a single device and this can be exploited for neuromorphic applications. For example, as shown in Fig. 1(c), PCM devices organized in a crossbar configuration can be used to emulate the synaptic elements in an artificial neural network.[3,4] The synaptic weights are captured by the conductance values of the PCM devices. The inputs from one layer of neurons are weighted by these conductance values (via Ohm's law) and the resulting current along the columns serves as inputs to the next layer of neurons. During the training of a neural network, the initial conductance values are typically chosen randomly, which are then modified (synaptic plasticity) via some appropriate learning rule. The programming pulses can be used to alter the conductance values during the training process. Unlike RESET pulses, which cause an abrupt transition to lower conductance values, the successive application of a partial SET pulse results in a more progressive increase in the conductance value. This cumulative evolution of conductance is highly beneficial for neuromorphic applications. Hence, often in PCM, only the partial SET pulses are used to implement synaptic plasticity rules.[4] To avoid the use of RESET pulses, PCM devices are organized in a differential configuration.[5] A comprehensive understanding of this accumulative behavior across a large number of devices is central to the realization of large-scale neural networks. Besides crystallization, there are other structural dynamics at play in PCM devices. These devices exhibit a temporal evolution of conductance values after the application of each programming pulse. This is attributed to a spontaneous structural relaxation of the material[6] and could also play a key role in neuromorphic computing.

In this article, we present a comprehensive model of PCM devices that captures the accumulative behavior, conductance drift, and read noise. Extensive experimental characterization of 10 000 PCM devices has been performed

[a)]Electronic mail: bipin@njit.edu
[b)]Electronic mail: ase@zurich.ibm.com

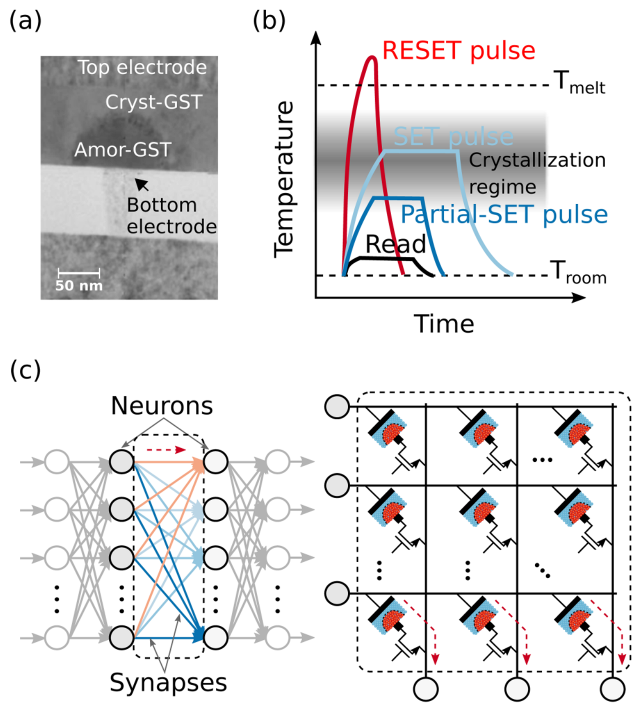

FIG. 1. (a) TEM image of a mushroom-type PCM device. Amorphous dome (Amor-GST) inside the crystalline (Cryst-GST) dielectric is visible. (b) Pictorial representation of the programming pulses and the resulting relative temperature for RESET, SET, partial SET, and read operation in PCM. (c) Schematic illustration of the application of PCM devices as synaptic elements in neuromorphic computing. A crossbar array of PCM devices could be used to represent the connection strengths in a neural network layer.

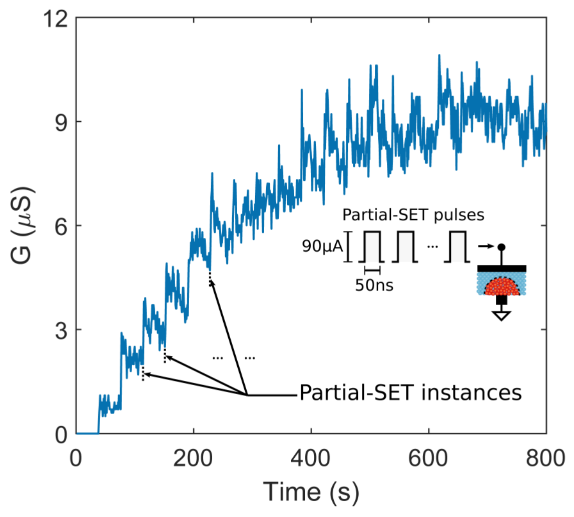

FIG. 2. The measured conductance evolution in a single device in accordance with the application of 20 consecutive partial SET pulses. The time instances of programming events are illustrated by the arrows. After the application of each pulse, the device conductance is measured 50 times.

to develop this statistical model. Finally, we demonstrate the efficacy of this model by using it to match experimentally observed array level characteristics and to train spiking and non-spiking artificial neural networks.

II. DEVICE CHARACTERIZATION AND MODELING

For device characterization, we used mushroom-type PCM devices fabricated in the 90 nm technology node.[7] The phase-change material is doped $Ge_2Sb_2Te_5$ (GST). A prototype chip comprising 3×10^6 devices was used in the study.[8] Individual devices are addressed via word lines and bit lines, and the devices have access transistors in series. The devices are programmed using current pulses of designated amplitude and width generated in the peripheral circuits. The conductances are read by applying a 0.3 V read pulse and the resulting current is read using an 8-bit analog to digital converter (ADC). The ADC is calibrated to span a conductance range between 0.1 μS and 27 μS.

First, the device conductances were initialized to a distribution close to 0.1 μS using iterative programming.[9] Subsequently, we applied 20 partial SET pulses of 90 μA amplitude and 50 ns duration. After the application of each pulse, devices are read 50 times. In addition, an immediate conductance measurement is performed approximately 100 ns after the programming pulse. However, subsequent measurements are obtained at time intervals in the order of seconds. As a result, consecutive programming pulses were applied with an average interval of 38.6 s for the 10 000 devices. The resulting conductance evolution, except for the immediate read after 100 ns, for one such representative device is shown in Fig. 2. In Secs. II A–II C, we will use all the measurements from the 10 000 devices (barring a few whose conductances which were outside the ADC limits) to develop the statistical model.

A. Accumulative behavior

First, we characterized the accumulative behavior arising from the successive application of partial SET pulses. To decouple the accumulative behavior from conductance drift, the 50th read measurement was used. The distribution of the conductance values as a function of the pulse number is shown in Fig. 3(a). It can be seen that the average conductance change is high at low conductance values and it gradually reduces as the conductance values increase. It can also be seen that there is significant randomness associated with the conductance values. This is mostly attributed to the inherent randomness associated with the crystallization process.[10,11] In fact, the inter- and intra- device variability in the array has been observed to be of comparable magnitude.[12–14]

To obtain a quantitative description of this behavior, we studied how the conductance change arising from the application of a single SET pulse depends on the conductance state of the device prior to the application of the pulse as well as the device's programming history. The devices were split into different groups based on their conductance values. Each group corresponds to a conductance interval of 1 μS. For each group, the mean ($\mu_{\Delta G}$) and standard deviation ($\sigma_{\Delta G}$) of the conductance change due to the application of a single programming pulse is plotted against the mean conductance (μ_G) of each group [see Figs. 3(b) and 3(c)]. The data points are generated only for those groups with 100 or more devices. This is repeated for the conductance values measured after the application of each programming pulse. In Figs. 3(b) and 3(c), each color corresponds to a single

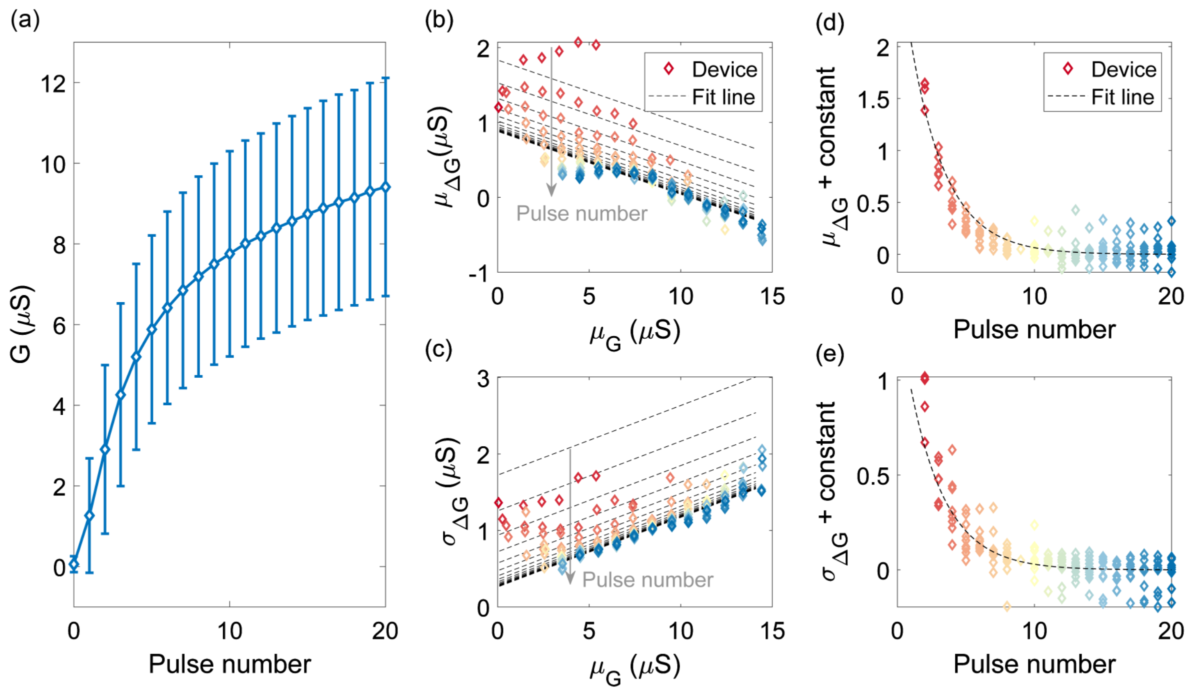

FIG. 3. (a) The statistics of cumulative conductance evolution as a function of the number of partial SET pulses. The error bars indicate one standard deviation. (b) The mean $\mu_{\Delta G}$ and (c) standard deviation $\sigma_{\Delta G}$ of conductance change as a function of the average initial conductance μ_G for each programming pulse. The initial conductance distribution for each programming pulse is divided into smaller intervals and μ_G, $\mu_{\Delta G}$, and $\sigma_{\Delta G}$ are determined separately for each interval. Each data point in (b) and (c) corresponds to an average of measurements from at least 100 devices. Also depicted are the fit lines used to obtain the model parameters. [(d), (e)] The same data points of $\mu_{\Delta G}$ and $\sigma_{\Delta G}$ are plotted as a function of the pulse number with a constant added for data points corresponding to a single μ_G interval. The dependency of $\mu_{\Delta G}$ and $\sigma_{\Delta G}$ on pulse number is approximated using an exponential function with a decay constant of 2.6.

programming pulse with the red color indicating the first pulse and the blue color the 20th pulse. We observe that there is a negative correlation between $\mu_{\Delta G}$ and μ_G suggesting a linear decrease in the conductance change as the device conductance increases. In addition, in a particular conductance range, the conductance change observed seems to decrease with increasing number of applied pulses. This behavior can be captured using a linear fit of a negative slope to map the relation between $\mu_{\Delta G}$ and μ_G for any particular pulse number. Furthermore, the dependency on the pulse number is encoded in the y intercept of this linear fit. It can be seen that for any given conductance value, the extent of conductance change induced by a single partial SET pulse reduces significantly with increasing number of applied pulses. This could be captured using an exponential empirical relation [Figs. 3(d) and 3(e)].

It can be seen that the behavior of $\sigma_{\Delta G}$ is also very similar to that of $\mu_{\Delta G}$ except that there is a positive correlation with the μ_G in this case. Therefore, the mean and standard deviation of the ΔG is modeled, respectively, using lines of negative and positive slopes and with an intercept which is an exponential function of the pulse number (p) as in the following equations [also in Figs. 3(b) and 3(c)]:

$$\mu_{\Delta G} = m_1 G + (c_1 + A_1 e^{-p/\alpha}), \tag{1}$$

$$\sigma_{\Delta G} = m_2 G + (c_2 + A_2 e^{-p/\alpha}), \tag{2}$$

where the fit parameters m_1, m_2, c_1, c_2, A_1, A_2, and α are -0.084, 0.091, 0.880, 0.260, 1.40, 2.15, and 2.6, respectively.

B. Conductance drift and read noise

In this section, we model the conductance drift in the devices arising from structural relaxation. For this, we use the 50 read measurements obtained after the application of each SET pulse. The mean conductance evolution after each programming event as a function of time is plotted in Fig. 4(a). The response is fitted using the model,[15,16]

$$G(t) = G(T_0)\left(\frac{t}{T_0}\right)^{-\nu}. \tag{3}$$

According to Eq. (3), if the device conductance, $G(T_0)$, is known at time T_0 after programming, the conductance at any time t can be estimated with the knowledge of the drift coefficient, ν. The estimated ν from the fit lines have a mean value of 0.04 [Fig. 4(a) inset]. Note that the logarithmic dependence on time suggests that after programming, the conductance drift slows down with time. We observe that the partial SET pulses result in a state that drifts, with a drift coefficient that decreases with increasing conductance $\mu_G(T_0)$. The application of a partial SET pulse re-initiates structural relaxation and conductance drift. Hence, we speculate that each partial SET pulse creates a new unstable glass state because of the atomic rearrangement that occurs upon its application, which then structurally relaxes to an energetically more favorable amorphous state.[6,17]

In addition to the conductance drift, there are also significant fluctuations in the conductance values (read noise) mostly arising from the $1/f$ noise exhibited by amorphous

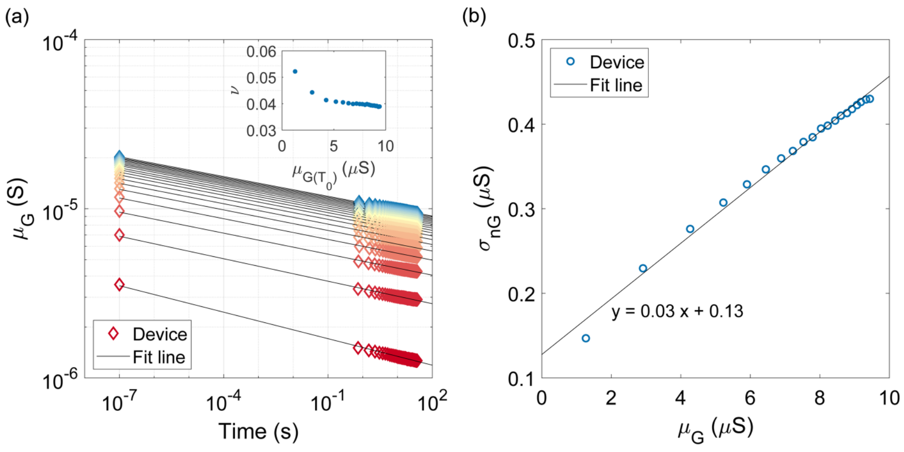

FIG. 4. (a) The average conductance evolution after each programming pulse obtained by the 50 read operation is fitted using Eq. (3). The estimated drift-coefficient, ν, is shown in the inset. (b) The read noise measured from the device array is plotted as a function of the average device conductance. The linear fit used to estimate read noise for the model in a state-dependent manner is also shown.

phase-change materials.[18] To model this, we estimated the noise from the last ten reads from the fifty read measurements. The objective was to decouple the read noise from the conductance drift. The standard deviation of the zero-mean read noise is plotted as a function of the mean conductance [see Fig. 4(b)]. It can be seen that the read noise increases with the device conductance. The read noise standard deviation, σ_{nG}, for the device conductance range is modeled using the linear relation,

$$\sigma_{nG} = m_3 G + c_3, \tag{4}$$

where $m_3 = 0.03$ and $c_3 = 0.13$.

C. The overall model description and validation

In this section, we combine the various elements of the model describing the accumulative behavior, conductance drift, and read noise to generate a complete statistical model and validate it based on the experimental data. The objective is to capture the evolution of conductance values for a large collection of devices after a certain time T_0 after programming with an arbitrary number of partial SET pulses. More specifically, we would like to determine the device conductance $G(t)$ at any time t, which has been initialized to approx. 0.1 μS, and is subjected to a sequence of 90 μA, 50 ns programming pulses with arbitrary time intervals between them.

To simulate this, three quantities are recorded per device: (a) $G_i(T_0)$, the conductance after T_0 time after the ith programming pulse for $i = 0, 1, 2\ldots$, (b) P_{mem}, a quantity that captures the programming history, and (c) t_p, the time of the last programming event. t_p is initialized to zero. Based on the chosen initial conductance value $G_0(T_0)$, P_{mem} is initialized to $P_{mem,0} = e^{-p_0/\alpha}$, where p_0 is the effective number of pulses applied to reach the initial conductance $G_0(T_0)$. p_0 is zero for initialization around 0.1 μS and p_0 for higher values of conductance is determined from the average conductance evolution curve shown in Fig. 3(a). The effective number of pulses versus conductance can be approximated empirically as

$$p_0 = 0.027\mu_G^3 - 0.15\mu_G^2 + 0.81\mu_G, \tag{5}$$

for conductance ranging from 0.1 μS to around 8 μS. After initialization, for the Nth programming event, P_{mem} is first updated as $P_{mem,N} = P_{mem,N-1}e^{-1/\alpha}$ for $N = 1, 2, \ldots$. Then $G(t)$, which has seen N programming pulses can be determined as follows:

$$\mu_{\Delta G_N} = m_1 G_{N-1}(T_0) + (c_1 + A_1 P_{mem}), \tag{6}$$

$$\sigma_{\Delta G_N} = m_2 G_{N-1}(T_0) + (c_2 + A_2 P_{mem}), \tag{7}$$

$$\Delta G_N = \mu_{\Delta G_N} + \sigma_{\Delta G_N}\chi, \tag{8}$$

$$G_N(T_0) = G_{N-1}(T_0) + \Delta G_N, \tag{9}$$

$$G(t) = G_N(T_0)\left(\frac{t - t_p}{T_0}\right)^{-\nu} + n_G. \tag{10}$$

Here, χ represents a Gaussian random number of mean zero and variance 1. Another zero mean Gaussian random variable n_G, captures the conductance fluctuations arising from PCM noise, whose standard deviation is calculated based on the instantaneous conductance state as dictated by the linear fit in Eq. (4) [also in Fig. 4(b)]. All the model parameters are listed in Table I. Please note that the conductance values predicted by the model are in μS.

First, the model is used to validate the same experimental data that was used to generate the model parameters. In particular, the model is used to generate the distribution of conductance values as a function of the number of programming pulses. As shown in Fig. 5, the mean and variance match remarkably well with experimental data. It can also be seen that the distributions themselves are remarkably similar. Figures 6(a)–6(c) show the conductance distribution from the 50th read after initialization, after the application of 5 programming pulses, and after the application of 20 programming

TABLE I. The model parameters.

Symbol	Value	Symbol	Value	Symbol	Value
m_1	−0.084	c_1	0.880	A_1	1.40
m_2	0.091	c_2	0.260	A_2	2.15
α	2.6	T_0	38.6 s	ν	0.04
m_3	0.03	c_3	0.13		

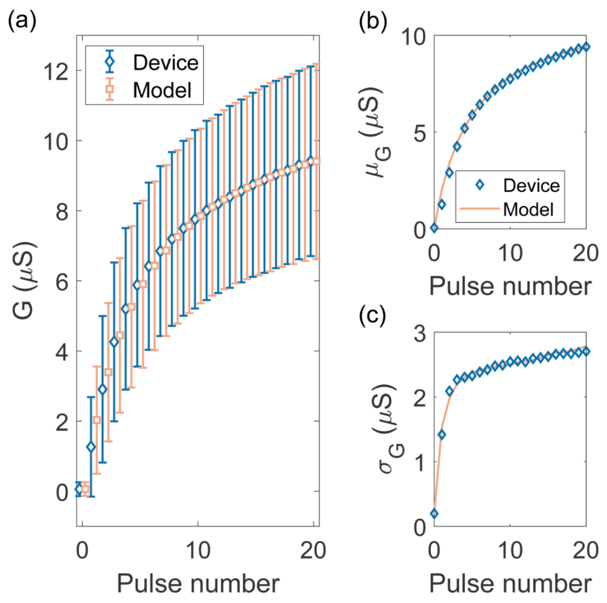

FIG. 5. (a) The distribution of conductance values obtained using the model as a function of the number of partial SET pulses and match with experimental device data. (b) The mean of the conductance as a function of the pulse number. (c) The standard deviation of the conductance as a function of the pulse number.

pulses, respectively. The model also matches the correlation coefficient observed between G and ΔG for the pulses applied. From Figs. 6(e) and 6(f), it can be seen that the statistical model also captures the individual device behavior remarkably well.

Additional measurements were performed where the devices are programmed with 20 programming pulses, however, with varying time intervals between the application of each pulse. The time interval was determined based on the number of reads performed and in the current experiment, each read process took approximately 1 s for the 10 000 devices. Figure 7 shows the programming events in time (top) and the resulting evolution of the mean conductance of the 10 000 devices (bottom). The spikes in the programming event plot correspond to the application of partial SET pulse and the device conductances are read at all other time instances. As discussed earlier, it can be seen that with the application of each programming pulse, the drift process is re-initiated. Another interesting observation is that the net change in conductance seems to be independent of structural relaxation. There is some evidence that structural relaxation slows down crystal growth rate.[2] But at least in these devices and these time scales, this does not seem to be significant. The final conductance values at the end of programming seem to converge to similar conductance levels independent of the rate of programming. Hence, our proposed model is able to capture this behavior remarkably well with the additional incorporation of Eq. (3).

III. SPIKING NEURAL NETWORK WITH MODELED PCM-SYNAPSES

The developed model could be used to simulate training behavior of neural networks and other possible learning systems which require adaptive weights. To illustrate this, we train a spiking neural network (SNN) and a non-spiking artificial neural network (ANN) with PCM based synapses whose conductance modulations are emulated by the model and discuss the effect of device behavior such as limited

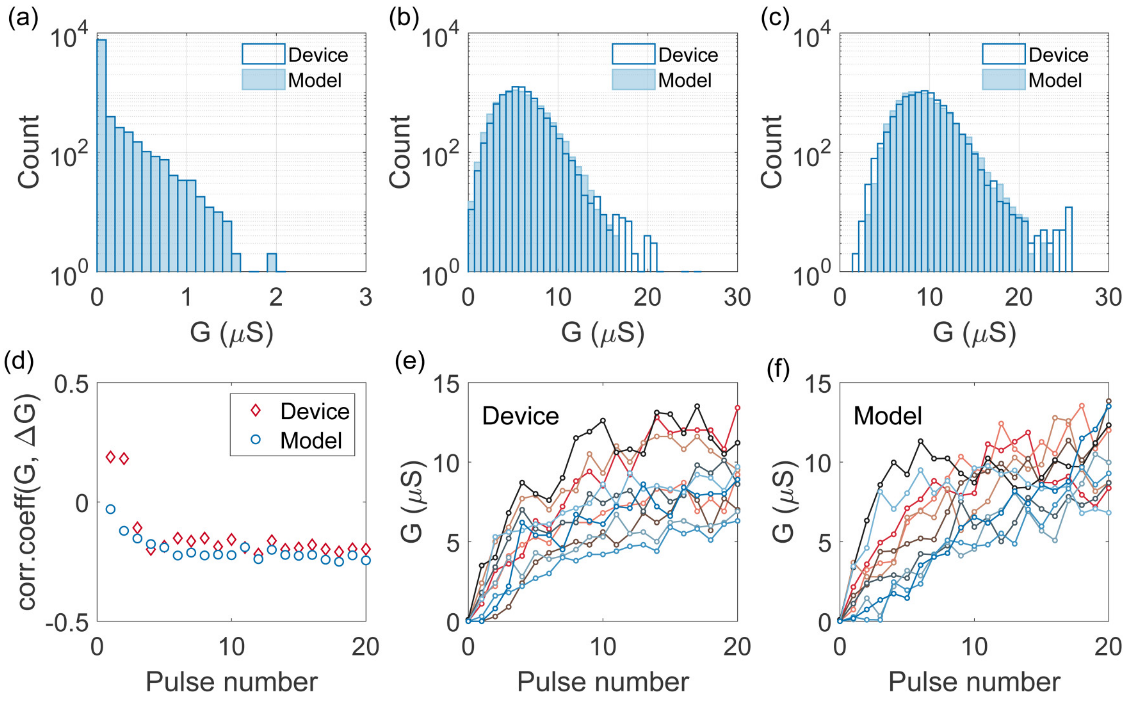

FIG. 6. The distribution of conductance values after (a) initialization, (b) the application of 5 programming pulses, and (c) the application of 20 programming pulses. It can be seen that there is a remarkable agreement between the experimental distribution and that predicted by the model. (d) The correlation coefficient between G and ΔG after the application of each programming pulse calculated based on the model and is compared with the experimental measurement. The conductance evolution of individual devices as measured experimentally (e) and as predicted by the model (f).

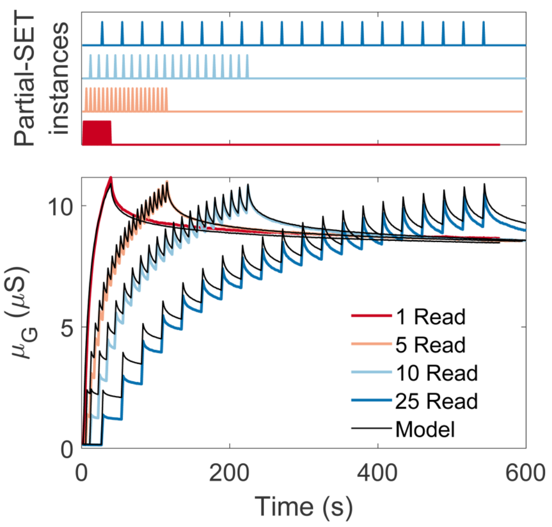

FIG. 7. The 10 000 PCM devices are programmed using sequences of 20 current pulses of 90 μA and 50 ns width. The number of read operations performed after each programming event is varied resulting in different time intervals between the programming events. The resulting conductance evolution during all the read operations is illustrated. The proposed model captures the experimentally observed behavior remarkably well.

granularity and stochasticity in the training. The PCM devices are assumed to be arranged in a crossbar array representing the connection strength between adjacent layers of neurons as in Fig. 1(c). The crossbar arrangement enables them to perform the weighted summation necessary for the dataflow through the network in constant time irrespective of the layer size.

SNNs are third generation neural networks that attempt to mimic biological neural network behavior. Biological neurons integrate their input over time in analog domain, while communicating with other neurons via spikes, enabling highly energy efficient signal encoding and processing. In SNNs, this neuronal behavior is typically emulated using a leaky-integrate and fire (LIF) model. The LIF neuron could be represented as a leaky capacitor that integrates incoming currents, with the integration reset when the voltage across the capacitor exceeds a threshold, and a spike is sent to the downstream neurons. This continuous time behavior makes the training and inference of SNNs in conventional digital hardware extremely inefficient. Non-volatile memory array based synapse networks with dedicated neuronal circuits at the periphery could potentially provide a more efficient non-von Neumann architecture for SNN implementation and training.

An example SNN is shown in Fig. 8(a). It has one LIF output neuron receiving 500 spikes streams via input synapses. The network is expected to generate a desired spike pattern from the inputs which is generated here from a Poisson random process for illustrative purposes. The network spike input and the desired output spike response are illustrated in Fig. 8(b). In response to each spike input, synapses generates currents modeled by the expression $I_{syn} = W \times (e^{-t/\tau_1} - e^{-t/\tau_2})$ as a function of time, t, where W is the synaptic strength. The task of the supervised learning algorithm is to adjust the weights such that the observed spikes match a desired pattern. One weight adjustment rule for SNNs that has been demonstrated recently is the NormAD algorithm,[19] which provides the network weight updates ΔW as

$$\Delta W = r \int_0^T e(t) \frac{\hat{d}(t)}{\|\hat{d}(t)\|} dt, \tag{11}$$

where r is the learning rate, T is the duration of the training pattern, and $e(t)$ is the difference between the desired and observed output spike trains. $\hat{d}(t)$ is obtained by convolving the synaptic current (I_{syn}) with an approximate impulse response of the LIF circuit.

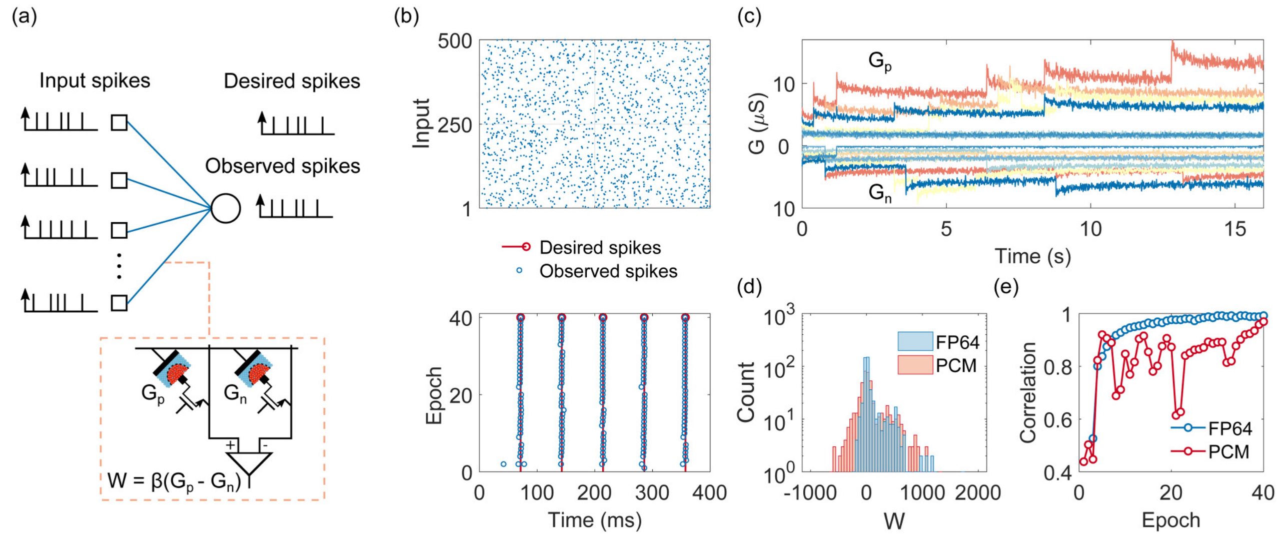

FIG. 8. (a) The spiking neural network used for the training simulation. Two PCM devices in the differential configuration used for the synapse is also shown. (b) Raster plot of input spike streams (top) and the desired spike streams from the output neuron (bottom). The observed spikes during each training epoch are plotted over the desired spike trains. The observed spikes are within 0.7 ms of the desired instances after 40 epochs of training. (c) Conductance evolution of a few synapses during the training illustrating the drift and read noise of PCM devices. (d) The final weight distribution from the PCM synapses along with the weights from a floating point (FP-64 bit) training is shown for reference. (e) The correlation between the desired and observed spike trains after 40 epochs with the PCM synapse is similar to that using floating point synapses.

The network synapses are realized using the PCM model. Because of the abrupt RESET behavior of the device, each synapse is realized using two PCM devices G_p and G_n in differential configuration such that $W = \beta(G_p - G_n)$, where β is a scaling factor.[5] Hence, synaptic potentiation is achieved by applying a partial SET pulse to G_p and depression is achieved by applying a partial SET pulse to G_n. This unidirectional programming often causes the device pairs to saturate preventing any further weight update. Hence, an occasional weight refresh is performed, based on the following criteria. If G_p or $G_n > G_x$, and $|G_p - G_n| < 0.25\, G_x$, where G_x is a threshold, both the devices are RESET, and the conductance difference is programmed to the device which had the higher conductance. For the hardware implementation, the RESET pulse shape could be determined from the PCM programming curve.[2,9,20] Here, the stochastic RESET behavior is simulated using an abrupt conductance transition to a distribution of mean $1\,\mu$S and standard deviation $0.5\,\mu$S.

For the training, the device conductances are initialized to a distribution around $2\,\mu$S with a standard deviation of $0.5\,\mu$S. During each training epoch, the 400 ms long spike sequences are presented to the network and the weight updates are computed. The scaling factor β is chosen to match the PCM based weight distribution to that obtained from an equivalent network trained with floating-point synaptic weights [Fig. 8(d)]. While this scaling enables PCM based synapses to represent the desired weight range, the achievable weight updates are limited by the device granularity. Furthermore, we assume the states of the individual devices are unknown to determine the optimum programming pulse. Hence, the estimated weight updates are converted to programming pulses by assuming an average conductance change of $0.75\,\mu$S for each partial SET pulse (as the $7.5\,\mu$S conductance range used in our study is typically reached within 10 pulses). The resulting conductance evolutions during training for a few synapses are shown in Fig. 8(c). As we see here, the PCM synapses drift and have read noise while computing the weight updates. The conductance programming is without any read-verify operation and is stochastic, which will simplify the system implementation and accelerate the training process. The training performance is evaluated based on a correlation between the desired and observed spike trains[19] and is plotted in Fig. 8(e). The corresponding numbers from a training assuming floating point synapses are shown for reference. In spite of the stochastic nature of PCM weight updates and conductance drift after programming, the SNN incorporating these devices exhibit training performance that is at par with the baseline software network.

IV. DEEP LEARNING WITH MODELED PCM-SYNAPSES

Now, we discuss the training of an artificial neural network (ANN) whose synapses are realized using the PCM differential configuration. The network is designed for the benchmark handwritten digit recognition task, based on 28×28 gray-scale images from the MNIST dataset. The dataset has 60 000 training images and 10 000 test images. In this exercise, we attempt to modify the standard backpropagation training algorithm to account for the limited device granularity and its effect is analyzed.

The network used for the task is shown in Fig. 9(a), which has two fully connected weight layers. The input layer neurons are linear; the hidden layer and the output layer neurons perform a logistic function on their inputs. The neuron responses are often termed as their activations. The training is performed using the backpropagation algorithm, an adaptation of gradient descent for multi-layer ANNs. The first stage of training, known as forward propagation, involves presenting the image pixels at the input layer and determining the output layer response. Out of the ten output neurons, the one corresponding to the input image is expected to have the highest neuron activation. The actual response is compared with the original class label and an aggregate network error is determined. The algorithm tries to minimize the error by adjusting the weights in the network. For this, the gradient of the error function with respect to each weight in the network is determined using the backpropagation algorithm. This involves sending the error computed in the last layer to the previous layers successively through the corresponding synaptic weights. If x_i is the neuron activation of the pre-neuron from forwarding propagation and δ_j, the error computed at the input of post-layer neuron of any weight during backpropagation, then desired weight-update for the synapse between these neurons can be computed as

$$\Delta W_{ij} = \eta . x_i . \delta_j, \tag{12}$$

where η is a suitably chosen learning rate. The weighted summation or the matrix-vector multiplication necessary for the forward and backward propagation can be realized using the same crossbar array of devices, by feeding the vector as voltages, respectively, along the word line or bit line and reading the matrix-multiplication results as currents along the corresponding bit line or word line, respectively.

During each epoch of the training, the network is presented with 60 000 training images and weight updates are computed using Eq. (12) after each image. Software training of the ANN with high precision floating point weights gives around 98% classification accuracy on the test set. Typically, $\Delta W/W < 10^{-3}$ for most of the ΔWs during this training. However, the PCM devices used for the synaptic implementation has a state-dependent and stochastic conductance update with very limited precision. Hence, when such non-volatile memory arrays are used for neural network training, transferring the desired weight updates becomes a major challenge. There are different proposals in the literature to solve this issue of low precision synapses by either using an additional memory for gradient accumulation[21,22] or using more complex synapse structures.[14,23] However, the additional overheads in these approaches constrain the maximum computational efficiency achievable in crossbar array based training architectures for neural networks. Here, we analyze the effect of the PCM response in training, where the weights are realized using two PCMs in differential configuration and the weight updates are implemented using single-shot programming pulses applied to the device model. Hence, the

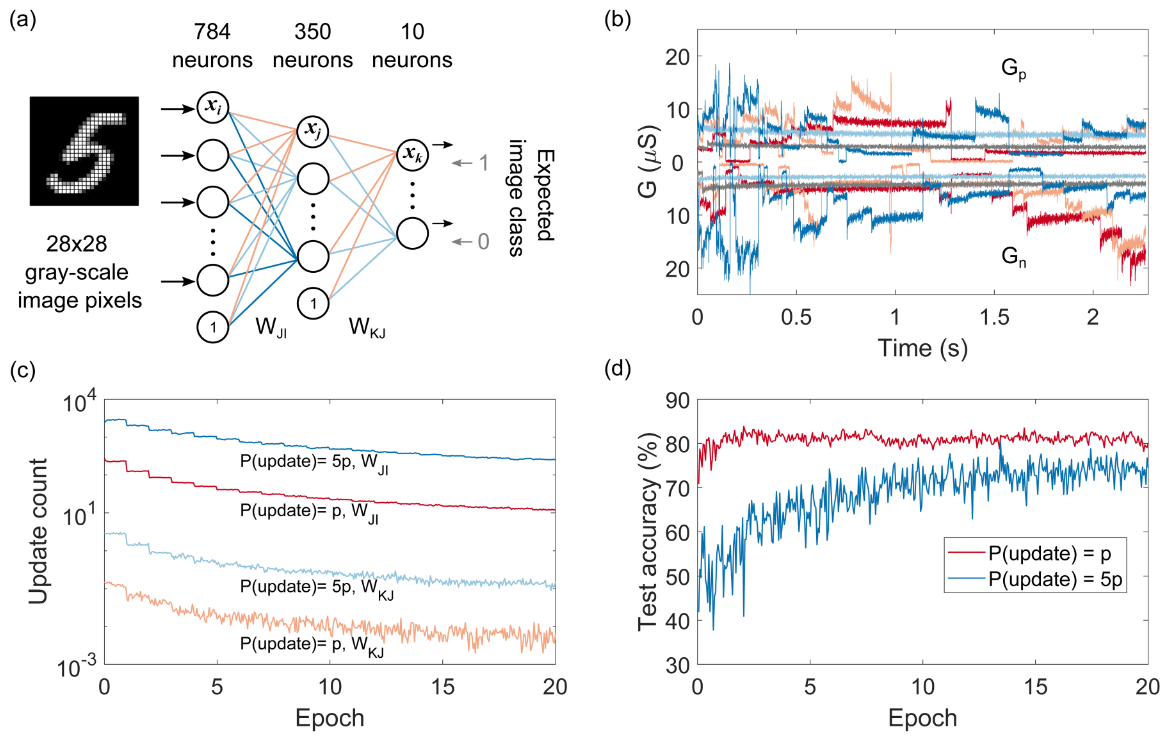

FIG. 9. (a) The artificial neural network used for handwritten digit classification based on the MNIST dataset. (b) The network weights are implemented using 2 PCMs in differential configuration [$W = \beta(G_p - G_n)$]. The conductance evolutions of the device pairs from few synapses are shown for the training duration. (c) The neuron activations and errors have been scaled to determine the weight update probability. The average number of weight updates per image in the two weight layers during the training for two chosen scaling factors are shown. (d) The test accuracies for the two update probabilities. The P(update)=p updated a smaller number of synapses using limited precision and stochastic PCM models and achieved higher test accuracies faster.

main ambiguity is in converting the desired weight updates into programming pulses. Due to the large disparity in the desired granularity of ΔW and the observed ΔG from the device, a linear mapping between the two will lead the network to rarely experience any weight update. To solve this issue, we used a scaled version of the x and δ to represent the probability to apply a programming pulse to the connected device. By adjusting the scaling factor, we could control the number of devices getting programmed in the network during each update. For illustration, we conducted two training experiments, where the update probabilities are p and $5p$, where p is a suitably chosen scaling factor. During training, we assumed, 1 μs computation delay per crossbar array resulting in a total training time of 2.26 s for 20 epochs. The conductance drift and read noise were re-evaluated after every training image during the simulations and a weight refresh was performed after every 1000 images [Fig. 9(b)]. Due to the probabilistic nature of the weight updates, out of the 278 260 weights in the ANN, only a small fraction of the total weights received updates after every image. The average number of devices updated after each image is shown in Fig. 9(c) for the two update probabilities. The corresponding classification accuracy of the network on the test set, which is not used for training is in Fig. 9(d). The training experiment that received lesser programming updates achieved higher accuracy and converged faster. In contrast to the high-precision software based training which has the flexibility to choose arbitrary learning rates, the weight updates in a low-precision device (such as the PCM synapse) are limited to the programming granularity of the device within its conductance range. The probabilistic sparse update scheme we use here could be viewed as an alternative approach to implement backpropagation, where instead of controlling the update of individual devices, the distance traveled on the error surface is chosen by controlling the number of devices being updated at any time. However, the limited device precision, non-linearity, and stochasticity seem to limit the maximum test accuracy achievable in this network to approximately 83%, which is comparable to experimentally observed training result in a similar network using PCM devices.[24]

While the training performance obtained in the simulation may seem subpar to the high-precision training, it is worth noting that biological synapses are stochastic and have state-dependent conductance update similar to the nano-scale non-volatile memory devices.[25] Some studies also suggest that they have a limited precision (∼4.6 bit).[26] Training algorithms designed assuming floating point precision for the network weights are not optimized for the limited precision weights. Considering the possible computational advantages of non-volatile memory based neural network implementations, adaptations or innovations of algorithms are necessary accounting the underlying architecture and hardware limitations. In such studies, the model we presented, which takes into account the device dynamics and variabilities, will be highly useful.

V. DISCUSSION

Now, we analyze the possible computational advantages of PCM based implementation of neural networks compared to the existing von Neumann architectures. We will also discuss how our modeling approach is applicable to other phase change material systems and how it might be affected by device scaling.

In a crossbar based matrix-vector multiplying unit, the PCM device acts as both local memory and an analog multiplier. The array structure enables them to perform standard $\mathcal{O}(N^2)$ complex matrix-vector multiplications in $\mathcal{O}(1)$ complexity with reduced data movement irrespective of the matrix size. The PCM devices are estimated to have 2- to 3-bit digital precision[27] and higher if some stochasticity could be tolerated. The area of a PCM cell with an access transistor is $\sim 25\,F^2$ (where F corresponds to the minimum lithographic pitch in a technology node), which could be reduced to $\sim 6\,F^2$ with a suitable diode based access device.[28] On the other hand, one bit SRAM area is $\geq 120\,F^2$ and the area of a 16-bit multiply-accumulate (MAC) required for neural network architectures is at least three orders of magnitude higher.[28,29] This results in trade-offs between the number of parallel computing units and on-chip memory for hardware implementations of neural networks using conventional CMOS technology. Since the available silicon real estate per die is limited, energy-hungry off-chip memory access becomes essential for storing the network parameters.[30] On the other hand, due to their in-memory computation capability, crossbar arrays are estimated to outperform modern graphics processing units (GPUs) by four orders of magnitude.[31] This is particularly advantageous for SNNs, as they need continuous time simulation, which calls for more analog and inherently parallel architectures.[32] The computational efficiency of the crossbar array could be maintained to a large extent in the ANN training if its weight-update stage could also be performed directly on the array devices, based on the coincidence of stochastic pulses that represent the neuronal activations and backpropagated errors. However, this necessitates ~ 10-bit update precision for the device to achieve state-of-the-art training accuracies.[31] Most of the non-volatile memory devices today are binary, while a few devices including the PCM offers a few extra bits of precision.

However, recent results suggest that lack of precision in the non-volatile memory could be compensated by an accompanying higher-precision unit.[21–23] The granularity of the synapse could also be improved by using multiple devices per synapse.[14] In order to study such approaches for larger and more complex neural network problems, compact models that reliably capture the device statistics are required. The model presented in this paper serves this purpose. Furthermore, the insights developed from such training explorations could also be used to determine the specifications for future devices.

The model we presented here is based on doped $Ge_2Sb_2Te_5$ in a mushroom structure fabricated in the 90 nm technology. This model is largely data-driven and is not based on specific material properties. The key aspects of the model are the negative correlation of the $\mu_{\Delta G}$ and positive correlation of the $\sigma_{\Delta G}$ with the average current state μ_G [Figs. 3(b) and 3(c)]. An intuitive explanation for these observations is as follows. Phase change memory devices have a chalcogenide sandwiched between a top electrode and a bottom electrode, with one of the contacts making a narrow contact (in both mushroom and pore PCM structures) with the dielectric acting as a point of heating. The temperature distribution within a PCM cell is decided by many factors such as thermal and electrical resistance and the specific heat capacity of the cell materials. Typically, the temperature distribution along the vertical symmetry axis is a skewed parabola with the maximum temperature located slightly above the heating contact in properly designed devices.[33,34] Furthermore, the crystal growth velocity in PCM increases monotonically until it reaches a peak crystallization temperature.[2] Though this peak temperature may depend on the material, in the partial SET pulse driven gradual conductance change operation, the programming pulses are chosen to operate below this temperature. Therefore, when a device initialized with a RESET operation is subjected to partial SET pulses, the point of maximum crystal growth will be around the point of maximum temperature. This results in large conductance change for the first few pulses. For further programming pulses of the same amplitude, assuming that the temperature distribution remains more or less unchanged and that the crystalline-amorphous boundary has moved to lower temperature regions due to earlier crystal growth, every subsequent programming will result in smaller conductance increase. This conductance saturation is captured in the model by the point where $\mu_{\Delta G}$ versus μ_G crosses zero. This zero-crossing behavior also makes the model bounded in its conductance range, even when simulated with a large sequence of pulses. The increase in the programming noise at higher conductance states may be attributed to the higher variability in the number of trap-states within the reduced volume of the amorphous region as sub-threshold conduction in these devices are trap-mediated.[35]

While $Ge_2Sb_2Te_5$ is the most commonly used material in PCM devices, other chalcogenide alloys have been explored for improved properties such as faster crystallization, reduced drift, and lower programming currents.[36,37] However, the qualitative description of the temperature distribution and the modeling approach we presented here is expected to remain valid in different phase change material systems provided the conductance modulation is driven by Joule heating and have similar crystal growth dynamics. For example, comparable partial SET conductance accumulation behavior has been reported in GeTe based PCMs.[38] Hence, the model we presented could be tuned to capture the gradual crystallization behavior if sufficient statistics on device characteristics are available. Phase change memory devices have also been demonstrated to be scalable via *ab initio* simulations[39,40] and experiments.[41] The temperature profile within the scaled devices remain more or less the same under constant voltage scaling,[42] and hence similar state-dependent conductance modulation behavior under partial SET programming pulses could be expected in them as well. However, the scaling of the electrode contact area reduces the amorphous volume involved in the conductance modulation, which could result in reduced granularity and higher stochasticity. For a given trap density, the changes in this smaller amorphous region could lead to higher programming noise. The ability of a PCM cell to provide gradual conductance state will also depend on the cell design. For example, in a mushroom cell, if the peak temperature point is too far away from the heater electrode, the amorphous region will not cover the heater unless very high powers are applied, effectively making multi-level operation almost impossible.[33] Therefore, the model could possibly be tuned to adapt to different phase change materials and technology nodes with sufficient data, provided the devices are not binary and have state dependent gradual conductance change.

VI. CONCLUSION

Phase-change memory devices are poised to play a key role in neuromorphic computing, in particular as synaptic elements in artificial neural networks. A cumulative increase in conductance value with the successive application of partial SET pulses is one of the key enablers for this functionality. In this article, through extensive characterization of thousands of PCM devices, we have developed an accurate statistical model that captures this accumulative behavior. This model also captures other attributes such as conductance drift arising from the structural relaxation of the phase-change materials. We demonstrated the efficacy of the model for training of artificial neural networks and discussed the importance of such statistical models for neuromorphic system emulations. The proposed model can be a powerful tool for the exploration of various neuromorphic algorithms.

ACKNOWLEDGMENTS

This work was supported in part by the European Union's Horizon 2020 Research, Innovation Program through the project MNEMOSENE under Grant 780215 and the U.S. National Science Foundation under Grant 1710009.

[1]G. Burr *et al.*, "Recent progress in phase-change memory technology," IEEE J. Emerg. Sel. Top. Circuits Syst. **6**, 146–162 (2016).
[2]A. Sebastian, M. Le Gallo, and D. Krebs, "Crystal growth within a phase change memory cell," Nat. Commun. **5**, 4314 (2014).
[3]B. L. Jackson, B. Rajendran, G. S. Corrado, M. Breitwisch, G. W. Burr, R. Cheek, K. Gopalakrishnan, S. Raoux, C. T. Rettner, A. Padilla *et al.*, "Nanoscale electronic synapses using phase change devices," ACM J. Emerg. Technol. Comput. Syst. **9**, 12 (2013).
[4]G. W. Burr *et al.*, "Neuromorphic computing using non-volatile memory," Adv. Phys. X **2**, 89–124 (2017).
[5]M. Suri *et al.*, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *Proc. IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2011), pp. 4.4.1–4.4.4.
[6]M. Le Gallo, D. Krebs, F. Zipoli, M. Salinga, and A. Sebastian, "Collective structural relaxation in phase-change memory devices," Adv. Electron. Mater. **4**, 1700627 (2018).
[7]M. Breitwisch, T. Nirschl, C. Chen, Y. Zhu, M. Lee, M. Lamorey, G. Burr, E. Joseph, A. Schrott, J. Philipp *et al.*, "Novel lithography-independent pore phase change memory," in *IEEE Symposium on VLSI Technology* (IEEE, 2007), pp. 100–101.
[8]G. F. Close *et al.*, "Device, circuit and system-level analysis of noise in multi-bit phase-change memory," in *IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2010), pp. 29.5.1–29.5.4.
[9]N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, and E. Eleftheriou, "Programming algorithms for multilevel phase-change memory," in *IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2011), pp. 329–332.
[10]T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," Nat. Nanotechnol. **11**, 693–699 (2016).
[11]M. Le Gallo, T. Tuma, F. Zipoli, A. Sebastian, and E. Eleftheriou, "Inherent stochasticity in phase-change memory devices," in *2016 46th European Solid-State Device Research Conference (ESSDERC)* (IEEE, 2016), pp. 373–376.
[12]I. Boybat, M. Le Gallo, T. Moraitis, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Stochastic weight updates in phase-change memory-based synapses and their influence on artificial neural networks," in *2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)* (IEEE, 2017), pp. 13–16.
[13]N. Gong, T. Ide, S. Kim, I. Boybat, A. Sebastian, V. Narayanan, and T. Ando, "Signal and noise extraction from analog memory elements for neuromorphic computing," Nat. Commun. **9**, 2102 (2018).
[14]I. Boybat, M. Le Gallo, S. R. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," Nat. Commun. **9**, 2514 (2018). 1711.06507.
[15]A. Pirovano, A. Lacaita, F. Pellizzer, S. Kostylev, A. Benvenuti, and R. Bez, "Low-field amorphous state resistance and threshold voltage drift in chalcogenide materials," IEEE Trans. Electron Devices **51**, 714–719 (2004).
[16]D. Ielmini, D. Sharma, S. Lavizzari, and A. L. Lacaita, "Reliability impact of chalcogenide-structure relaxation in phase-change memory (PCM) cells-Part I: Experimental study," IEEE Trans. Electron Devices **56**, 1070–1077 (2009).
[17]M. Boniardi and D. Ielmini, "Physical origin of the resistance drift exponent in amorphous phase change materials," Appl. Phys. Lett. **98**, 243506 (2011).
[18]M. Nardone, V. Kozub, I. Karpov, and V. Karpov, "Possible mechanisms for 1/f noise in chalcogenide glasses: A theoretical description," Phys. Rev. B **79**, 165206 (2009).
[19]N. Anwani and B. Rajendran, "Normad-normalized approximate descent based supervised learning rule for spiking neurons," in *International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2015), pp. 1–8.
[20]M. Le Gallo, A. Sebastian, G. Cherubini, H. Giefers, and E. Eleftheriou, "Compressed sensing recovery using computational memory," in *IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2017), pp. 28.3.1–28.3.4.
[21]M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. of the 28th International Conference on Neural Information Processing Systems—Volume 2, NIPS'15* (MIT Press, Cambridge, MA, 2015), pp. 3123–3131.
[22]S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Mixed-precision architecture based on computational memory for training deep neural networks," in *IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2018), pp. 1–5.
[23]S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, and G. W. Burr, "Equivalent-accuracy accelerated neural-network training using analogue memory," Nature **558**, 60–67 (2018).
[24]G. W. Burr *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 Trans. Electron Devices **62**, 3498–3507 (2015).
[25]G.-q. Bi and M.-m. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," J. Neurosci. **18**, 10464–10472 (1998).
[26]T. M. Bartol, C. Bromer, J. P. Kinney, M. A. Chirillo, J. N. Bourne, K. M. Harris, and T. J. Sejnowski, "Hippocampal spine head sizes are highly precise," bioRxiv 016329 (2015).
[27]A. Athmanathan, M. Stanisavljevic, N. Papandreou, H. Pozidis, and E. Eleftheriou, "Multilevel-cell phase-change memory: A viable technology," IEEE J. Emerg. Sel. Top. Circuits Syst. **6**, 87–100 (2016).
[28]G. W. Burr, M. J. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. A. Lastras, A. Padilla, B. Rajendran, S. Raoux, and R. S. Shenoy, "Phase change memory technology," J. Vac. Sci. Technol. B Nanotechnol. Microelectron. Mater. Process. Meas. Phenom. **28**, 223–262 (2010). arXiv:1001.1164.
[29]T. T. Hoang, M. Sjalander, and P. Larsson-Edefors, "A high-speed, energy-efficient two-cycle multiply-accumulate (MAC) architecture and its application to a double-throughput MAC unit," IEEE Trans. Circuits Syst. I Reg. Pap. **57**, 3073–3081 (2010).
[30]V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," Proc. IEEE **105**, 2295–2329 (2017). arXiv:1703.09039.

[31]T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: Design considerations," Front. Neurosci. **10**, 1–19 (2016).
[32]S. Nandakumar, S. R. Kulkarni, A. V. Babu, and B. Rajendran, "Building brain-inspired computing systems: Examining the role of nanoscale devices," IEEE Nanotechnol. Mag. **12**, 19–35 (2018).
[33]D.-H. Kim, F. Merget, M. Först, and H. Kurz, "Three-dimensional simulation model of switching dynamics in phase change random access memory cells," J. Appl. Phys. **101**, 064512 (2007).
[34]C. Ma, J. He, J. Lu, J. Zhu, and Z. Hu, "Modeling of the temperature profiles and thermoelectric effects in phase change memory cells," Appl. Sci. **8**, 1238 (2018).
[35]D. Ielmini and Y. Zhang, "Analytical model for subthreshold conduction and threshold switching in chalcogenide-based memory devices," J. Appl. Phys. **102**, 054517 (2007). arXiv:1602.01885.
[36]S. Raoux, F. Xiong, M. Wuttig, and E. Pop, "Phase change materials and phase change memory," MRS Bull. **39**, 703–710 (2014).
[37]F. Rao, K. Ding, Y. Zhou, Y. Zheng, M. Xia, S. Lv, Z. Song, S. Feng, I. Ronneberger, R. Mazzarello, W. Zhang, and E. Ma, "Reducing the stochasticity of crystal nucleation to enable subnanosecond memory writing," Science **358**, 1423–1427 (2017).
[38]M. Suri, O. Bichler, D. Querlioz, B. Traoré, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Physical aspects of low power synapses based on phase change memory devices," J. Appl. Phys. **112**, 054904 (2012).
[39]J. Liu, X. Xu, L. Brush, and M. P. Anantram, "A multi-scale analysis of the crystallization of amorphous germanium telluride using ab initio simulations and classical crystallization theory," J. Appl. Phys. **115**, 023513 (2014).
[40]J. Liu, "Microscopic origin of electron transport properties and ultrascalability of amorphous phase change material germanium telluride," IEEE Trans. Electron Devices **64**, 2207–2215 (2017).
[41]F. Xiong, E. Yalon, A. Behnam, C. Neumann, K. Grosse, S. Deshmukh, and E. Pop, "Towards ultimate scaling limits of phase-change memory," in *2016 IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2016), pp. 4.1.1–4.1.4.
[42]S. Kim and H.-S. Wong, "Analysis of temperature in phase change memory scaling," IEEE Electron Device Lett. **28**, 697–699 (2007).