Synaptogen: A Cross-Domain Generative Device Model for Large-Scale Neuromorphic Circuit Design

Tyler Hennen, Leon Brackmann, Tobias Ziegler, Sebastian Siegel, Stephan Menzel, *Senior Member, IEEE*, Rainer Waser, Dirk J. Wouters, *Member, IEEE*, and Daniel Bedau

***Abstract*— We present a fast generative modeling approach for resistive memories that reproduces the complex statistical properties of real-world devices. By training on extensive measurement data of an integrated 1T1R array (6000 cycles of 512 devices), an autoregressive stochastic process accurately accounts for the cross-correlations between device switching parameters, while nonlinear transformations ensure agreement with the joint cycle-to-cycle (C2C) and device-to-device (D2D) write distributions. In addition to a high-level programming version, the model is also implemented in Verilog-A to enable efficient simulation of analog circuits. This statistically comprehensive model can be used to simulate crossbar sizes with up to 1024 × 1024 devices, and benchmarks show that it achieves read/write speeds several orders of magnitude higher than a variability-aware physics-based compact model and over 10× faster than even a simplified and deterministic compact model.**

***Index Terms*— Circuit modeling, neural network hardware, resistive circuits, statistics, stochastic circuits.**

I. Introduction

A PRESSING challenge for large-scale simulations of neuromorphic systems is the availability of suitable synaptic device models for resistive memories such as ReRAM [1]. For applications, it is important to accurately capture the complex stochastic behavior of the devices, and to handle modern neural network sizes, models should also be fast enough to simulate millions of cells at once.

Many stochastic models, both empirical and physics-based, have been used in different simulation domains; from compact models compatible with circuit simulators [2], [3], [4], [5], [6] to highly abstracted models for simulating analog machine learning (ML) tasks in high-level programming languages [7], [8]. To improve accuracy, there has been progress in using measurement data to infer distributions of model parameters [9], [10]. However, models often use simplistic variability mechanisms such as sampling independent parameters from normal distributions. While able to approximate device variance via ad hoc fitting procedures, models often have limited ability to capture all the statistical details important for network performance. Furthermore, even device models which have freely available implementations are not applicable across different simulation domains, and although they can easily bottleneck many-device simulations, performance benchmarks are rarely reported.

In a previous work, we showed that a computationally lightweight generative model can be trained on electrical characteristics of a fabricated device to provide high-speed simulations of large numbers ($>10^9$) of cells with unprecedented statistical accuracy [11]. While we earlier focused solely on large-scale simulations using a high-level programming language, here we present a circuit-level model implemented in the hardware description language (HDL) Verilog-A, necessary to bridge the divide between the ML and analog circuit simulation domains. The model was expanded to cover a device configuration with access transistors (1T1R), to more accurately model gradual RESET transitions for multilevel programming, and to incorporate many devices into the measurement, training, and generation processes.

The stochastic modeling closely captures the distributions, cross-correlations, and history dependence of ReRAM switching parameters as the devices are cycled [cycle-to-cycle (C2C)], as well as how those cycling statistics vary between the different devices on the chip [device-to-device (D2D)]. This open-source device model is far more

Manuscript received 10 April 2024; revised 17 June 2024; accepted 8 July 2024. Date of publication 19 July 2024; date of current version 23 August 2024. This work was supported in part by the German Federal Ministry of Education and Research through the Projects NEUROTEC II under Grant 16ME0399 and Grant 16ME0398K and in part by NeuroSys under Grant 03ZU1106AA and Grant 03ZU1106AB. The review of this article was arranged by Editor S. Alam. *(Corresponding author: Tyler Hennen.)*

Tyler Hennen, Leon Brackmann, Tobias Ziegler, and Dirk J. Wouters are with the Institut für Werkstoffe der Elektrotechnik 2 (IWE2), RWTH Aachen University, 52074 Aachen, Germany (e-mail: t.hennen@iwe.rwth-aachen.de).

Sebastian Siegel and Stephan Menzel are with the Peter Grünberg Institute (PGI-7), Forschungszentrum Jülich, 52428 Jülich, Germany.

Rainer Waser is with IWE2, 52074 Aachen, Germany, and also with PGI-7, 52428 Jülich, Germany.

Daniel Bedau is with Western Digital Corporation, San Jose, CA 95119 USA (e-mail: daniel.bedau@wdc.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TED.2024.3427616.

Digital Object Identifier 10.1109/TED.2024.3427616

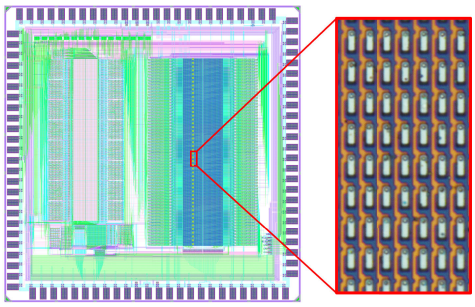

Fig. 1. ReRAM chip layout in the MAD200 process design kit (left) and an optical image of the fabricated 1T1R ReRAM array (right).

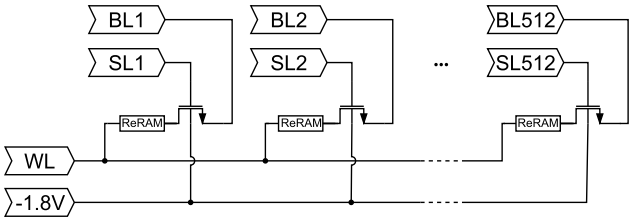

Fig. 2. Simplified circuit diagram showing a connected vector of 512 1T1R ReRAM devices which were individually selected for measurement of model training data. Transistor bodies were biased to −1.8 V as bipolar voltage sweeps were applied to the WL and current was measured at the respective BL (0 V).

statistically comprehensive than the existing compact models and outperforms them by several orders of magnitude in read/write benchmarks. In circuit simulations, we demonstrate weight programming and readout of crossbar arrays with up to 256 × 256 and 1024 × 1024 devices, respectively, the feasibility of which has not been previously shown.

II. Methods

A. Electrical Measurements

An integrated ReRAM chip was obtained through the manufacturing broker Circuits Multi-Projects (CMP) and used for electrical measurements. A 512 × 32 1T1R crossbar array was part of a custom layout within the Memory Advanced Demonstrator 200 mm (MAD200) design environment (Fig. 1). Select logic and access transistors were implemented in the HCMOS9A STMicroelectronics 130-nm CMOS process, and ReRAM devices with material stack TiN/HfO_2/Ti were deposited in a postprocess by CEA-LETI [12]. Each ReRAM device in the array was connected in series with an integrated common-source N-channel field-effect transistor in a standard 1T1R configuration. The 512 bit lines (BLs) were multiplexed to one output pin, the corresponding 512 select lines (SLs) were multiplexed to another output pin, whereas each of the 32 word lines (WLs) was directly routed to individual output pins. The packaged chip was mounted on a custom printed circuit board (PCB) providing a PC interface via the digital outputs of a data acquisition board whereby devices can be individually addressed for measurement. In this work, a total of 512 devices sharing a single WL in the array were sequentially selected to collect training data (Fig. 2).

High-speed measurements were performed using external generating and sampling equipment connected to the PCB over 50-Ω lines. To collect bipolar switching cycles continuously

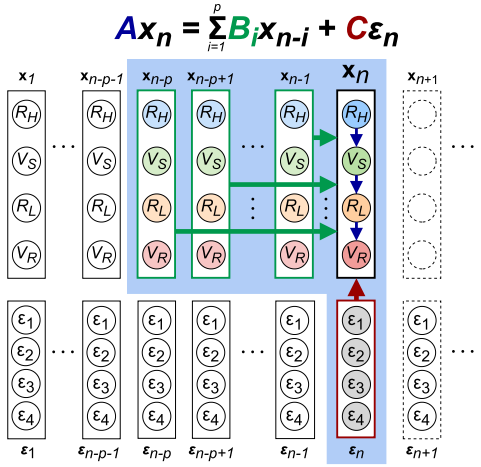

Fig. 3. Graphical depiction of the VAR(p) base process used to reproduce memory cycling statistics. Past features within cycle range p have a linear deterministic impact on future values, and a 4-D white noise process ϵ_n contributes stochasticity to each feature.

with a single driving signal, an unusual 1T1R biasing was necessary. The chip substrate (and FET body) was biased to −1.8 V relative to signal ground, and the gate was biased to 1.35 V while a bipolar driving signal was applied to the WL and current was measured at the BL through a 50-Ω shunt to 0 V. Devices were first electroformed by a 3-V amplitude triangle pulse with 1-ms duration before being cycled by a continuous triangle waveform between −1.5 and 2 V with 1-ms period. Preconditioning cycles were initially applied to each cell before collecting 6000 switching current versus voltage (I, V) traces for each of the 512 devices.

B. Statistical Modeling

The core concept of the generative device model is to first extract important features (i.e., resistance and voltage threshold levels) from each cycle of the training data, and then learn to efficiently generate new samples with very similar statistical properties. Using the generated features as a guide, we approximate the $I(V)$ dependence for simulated cells according to the voltage sequence applied to them.

1) Feature Generation: The chosen features to model are extracted from the raw data and organized into vector time series

$$\boldsymbol{x}_{n,m} = \begin{bmatrix} R_H \\ V_S \\ R_L \\ V_R \end{bmatrix}_{n,m} \tag{1}$$

for each cycle number $n \in [1, N]$ and device number $m \in [1, M]$. The feature vectors are arranged from top to bottom in the order that they occur in the measurement; R_H is the resistance of the high-resistance state (HRS), V_S is the voltage of the SET transition, R_L is the resistance of the low-resistance state (LRS), and V_R is the voltage at the start of the RESET transition. The details of this feature extraction are documented in [11].

Feature vector generation is based on a discrete vector autoregressive (VAR) stochastic process (Fig. 3),

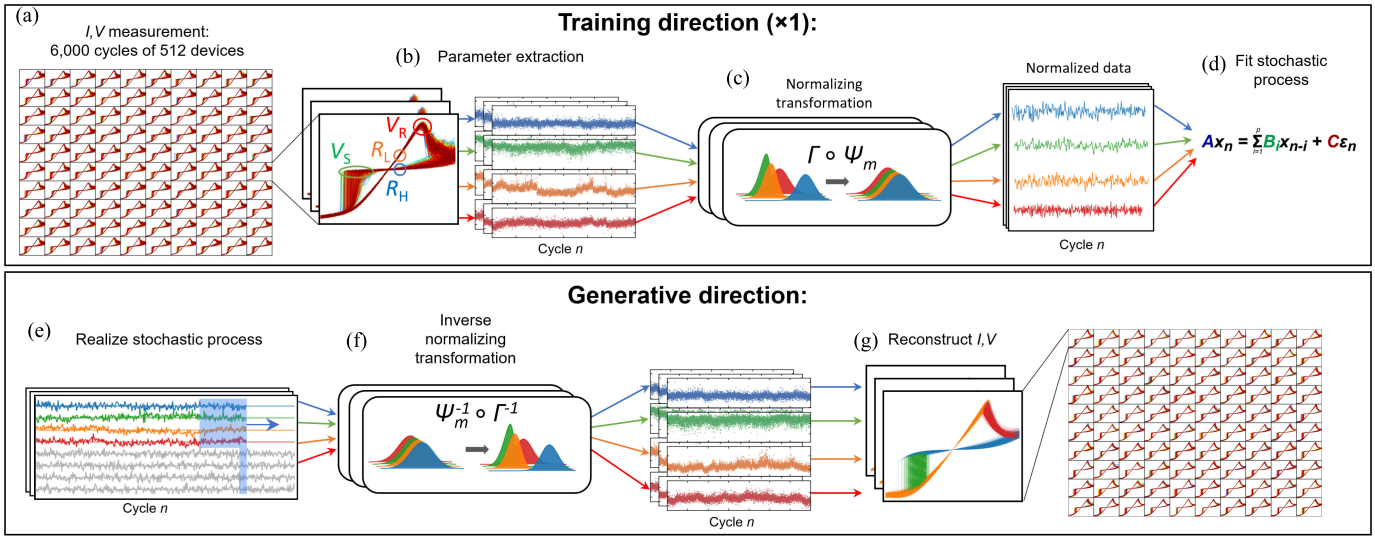

Fig. 4. Overview of the generative modeling approach. Training direction: (a) collect I, V data (N cycles × M devices), (b) extract feature vectors, (c) learn a distribution of normalizing transformations, and (d) fit a stochastic process (VAR) to the normalized data. Generative direction: (e) realize an independent VAR process for each simulated cell, (f) apply device-specific random de-normalizing transformations to the VAR outputs, and (g) as voltages are applied, reconstruct I, V dependence of each cell.

which captures the cycle history dependence and the correlations between features [13]. A VAR(p) models the nth feature vector as a linear function of past values within cycle range p and is driven by 4-D white noise $\boldsymbol{\epsilon}_n$. The stochastic process has the easily computable form

$$\boldsymbol{A}\boldsymbol{x}_n = \sum_{i=1}^{p} \boldsymbol{B}_i \boldsymbol{x}_{n-i} + \boldsymbol{C}\boldsymbol{\epsilon}_n \tag{2}$$

where $\boldsymbol{A}$, $\boldsymbol{B}_i$, and $\boldsymbol{C}$ are 4 × 4 weight matrices subject to training.

To map the normally distributed output of the VAR process to the joint empirical distribution measured across cycles and across devices, we apply a sequence of invertible transformations. The parameters of these transformations are learned in a single training pass in which the generative process is carried out in reverse (Fig. 4). Thereby, the marginal distributions of the extracted features are normalized in two steps. First, the device-specific mean and variance over the sampled cycles are standardized using an affine transformation $\boldsymbol{\Psi}_m$ (Fig. 5). Then, to further shape the intermediate probability densities into normal distributions, the affine transformation is followed by a parameterized, nonlinear quantile transform $\boldsymbol{\Gamma}$. A VAR(p) process is then fit to the normalized data using least-squares regression.

In the generative direction, the learned transformations are inverted and applied to independent realizations of the VAR process for each simulated device. The normalizing map $\boldsymbol{\Gamma}$ is defined such that its inverse consists of elementwise polynomial evaluations

$$\boldsymbol{\Gamma}^{-1}(\boldsymbol{x}) = \begin{bmatrix} \gamma_1(R_H) \\ \gamma_2(V_S) \\ \gamma_3(R_L) \\ \gamma_4(V_R) \end{bmatrix} \tag{3}$$

where γ_i are the fourth-degree polynomials and are visualized in Fig. 6.

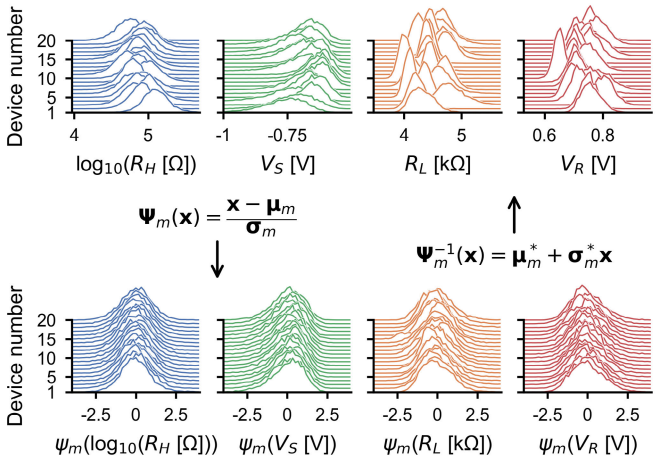

Fig. 5. Ridgeline plots (stacked histograms) show a standardizing affine transformation applied to a representative sample of 20 devices. The forward transformation $\boldsymbol{\Psi}_m$ is applied in the training direction as a first step to normalize the C2C feature distributions. Here, $\boldsymbol{\mu}_m$ and $\boldsymbol{\sigma}_m$ are the sample means and standard deviations of the feature vectors for device m across all the cycles, respectively. The inverse transformation is used in the generative direction, where $\boldsymbol{\mu}_m^*$ and $\boldsymbol{\sigma}_m^*$ are sampled from a distribution estimated from the entire training set.

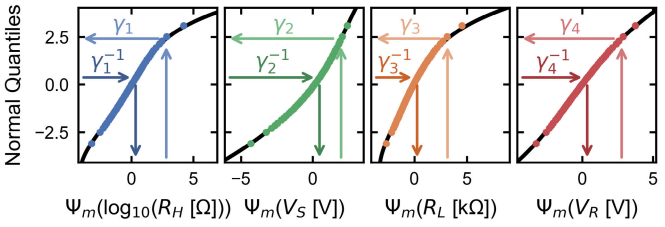

Fig. 6. Elementwise nonlinear quantile transform $\boldsymbol{\Gamma}$ adapted to the training data. The inverse transformations are polynomial functions γ_i designed for fast evaluation during the generative process. The nonlinearity allows the model to reproduce the training data's nonnormal and asymmetric distributions.

To restore device-specific offsets and scales to the generated features, we invert $\boldsymbol{\Psi}_m$ by approximating the distribution of an 8-D block vector of sampled C2C means (μ) and

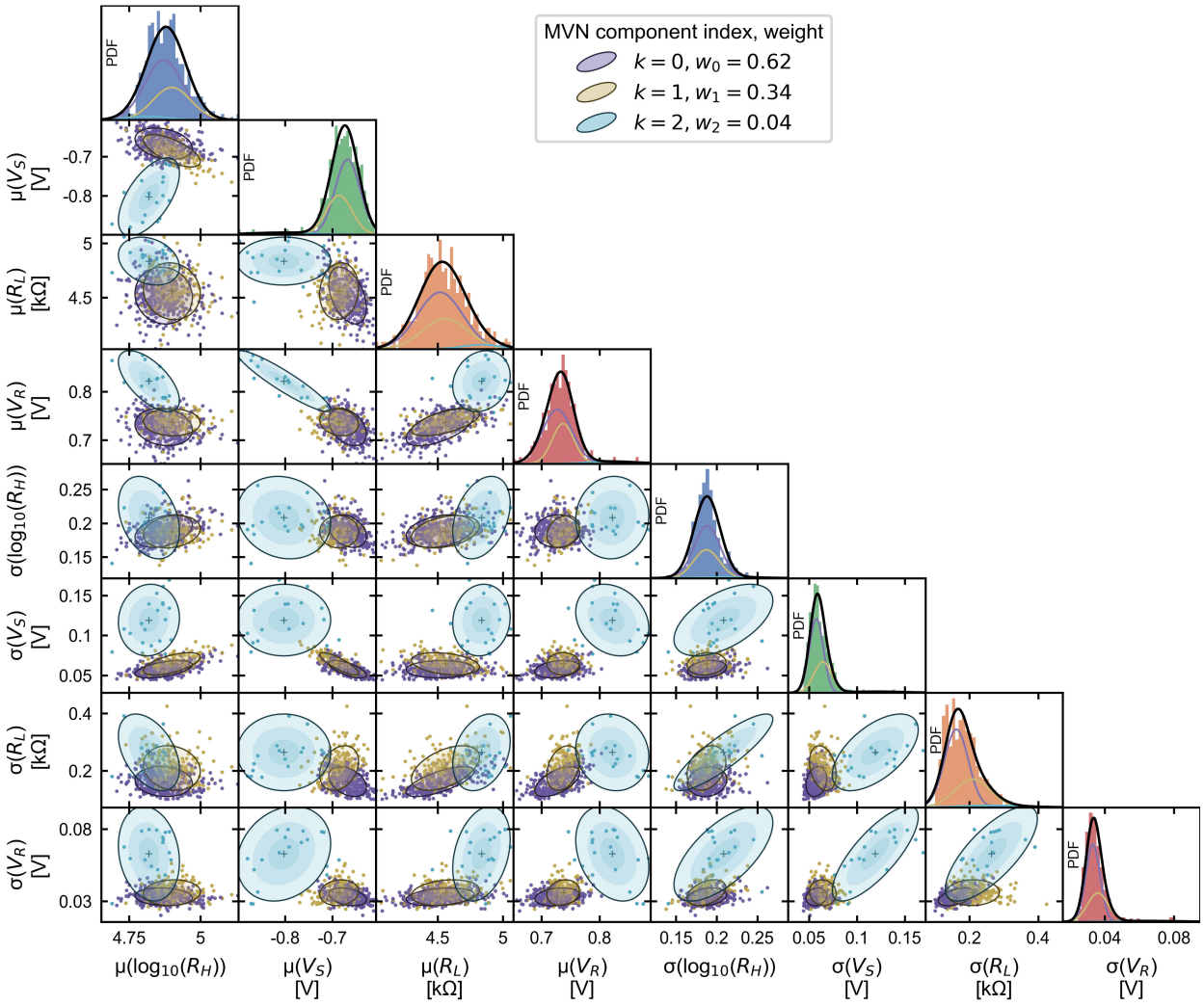

Fig. 7. Correlative scatterplot of the feature means (μ) and standard deviations (σ) over all the cycles of devices in the training set. The 512 datapoints are fit and classified by a GMM with three Gaussian components (purple, yellow, and teal), which allows sampling of new mean and standard deviation vectors in the generative process. Component $k = 2$ (teal) captures the multivariate structure of a device defect occurring in 4% of devices. The diagonal subplots show that the weighted addition of the marginal probability distribution functions (PDFs) of the three components closely fits the histograms of the input data.

standard deviations (σ)

$$\boldsymbol{S}_m = \begin{bmatrix} \boldsymbol{\mu}_m \\ \boldsymbol{\sigma}_m \end{bmatrix} = \begin{bmatrix} \mu(R_{H,n}) \\ \mu(V_{S,n}) \\ \mu(R_{L,n}) \\ \mu(V_{R,n}) \\ \sigma(R_{H,n}) \\ \sigma(V_{S,n}) \\ \sigma(R_{L,n}) \\ \sigma(V_{R,n}) \end{bmatrix}_m . \tag{4}$$

This distribution is represented by a superposition of multivariate normal (MVN) distributions, which is known as a Gaussian mixture model (GMM). A GMM is cheap to sample from and allows a close fit of the covariance structure of the main cluster of $\boldsymbol{S}_m$ datapoints. The GMM also captures the structure of statistical abnormalities that occur (i.e., defective devices), which may have a disproportionate impact on system performance. A three-component GMM, denoted

$$\boldsymbol{S}_m^* = \begin{bmatrix} \boldsymbol{\mu}_m^* \\ \boldsymbol{\sigma}_m^* \end{bmatrix} \tag{5}$$

is fit to the empirical distribution by the expectation–maximization algorithm using k-means initialization and is visualized in Fig. 7.

Note that all-positive features with logarithmically skewed distributions may be logged before training and exponentiated after generation to assist with the normalization and to prevent generation of negative values. In the present case, this log transformation is used only for the R_H feature.

2) Modeling the I(V) Dependence: The nonlinear $I(V)$ state for each cell is modeled as a linear combination of two static, global limiting polynomials $I_H(V)$ and $I_L(V)$ whose coefficients are estimated from the training data. This way, the model can reproduce a wide variety of asymmetric nonlinearities present in both HRS and LRS and can also interpolate between them to represent intermediate resistance states. The degrees of the polynomials are untrained hyperparameters and can be adapted as necessary to suit the amount of nonlinearity in the training data. In the present case, $I_H(V)$ and $I_L(V)$ were chosen to have degree 5 and 6, respectively, empirically large enough to capture the inherent nonlinearity of the HRS and also the shape of the series transistor curve visible in the LRS.

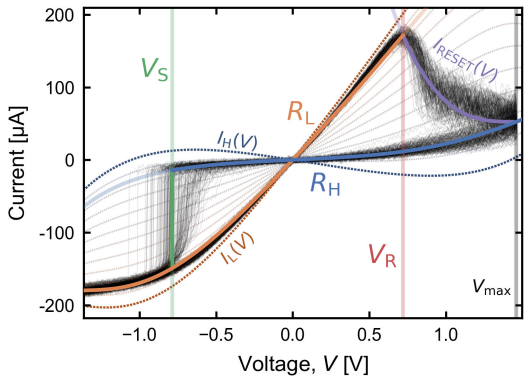

Fig. 8. Exemplary I, V cycle reconstructed from its feature vector representation. States are bounded by polynomials $I_H(V)$ and $I_L(V)$. Intermediate states (weights) are programmed by applying a voltage between V_R and V_{max}. Experimental traces (black) are plotted in the background for reference.

Resistance levels of the devices are tracked by continuous state variables $r_m \in (0, 1)$, which represent the degree of mixing between the predefined limiting polynomials. The current as a function of voltage for device m assumes the form

$$I_m(r_m, V) = r_m I_H(V) + (1 - r_m) I_L(V). \tag{6}$$

The state variable corresponding to each generated resistance level R is calculated using the function

$$r(R) = \frac{I_L(V_0) - V_0 R^{-1}}{I_L(V_0) - I_H(V_0)} \tag{7}$$

which uniquely sets the static resistance of the device (evaluated at $V_0 = 0.2$ V) equal to R.

Transitions of the state variables occur when the voltage applied to a device exceeds the threshold levels for SET or RESET in its current cycle. The transitions connect each generated resistance state to the following one, as illustrated in Fig. 8. Below the SET threshold $V_{S,n}$, there is an instantaneous transition from resistance state $R_{H,n}$ to $R_{L,n}$. After SET has occurred, voltages above $V_{R,n}$ gradually shift the resistance state from $R_{L,n}$ to $R_{H,n+1}$. This gradual RESET proceeds such that the device current has an empirical functional form

$$I_{\text{RESET}}(V) = a(V_{\max} - V)^{\eta} + c \tag{8}$$

where

$$a = \frac{I_{\text{LRS},n}(V_{R,n}) - I_{\text{HRS},n+1}(V_{\max})}{\left(V_{\max} - V_{R,n}\right)^{\eta}} \tag{9}$$

and

$$c = I_{\text{HRS},n+1}(V_{\max}) \tag{10}$$

satisfy the transition boundary conditions. Here, $V_{\max}$ is the maximum voltage applied in the experimental sweeps, and the constant $\eta \approx 3.0$ sets the curvature of the RESET transition as estimated by a least-squares fit to the training data.

In this scheme, the state evolution of a device is determined by the discrete sequence of voltages applied to it. While the model has no internal representation of time, the training data carry information about the resistive switching dynamics at the timescale used for its measurement. As such, the simulation timescale is assumed to be appropriately matched with the experimental timescale used to collect the training data. Furthermore, while the training data are collected using continuous voltage sweeps, it is also common to program memory cells by applying (square) voltage pulses. Triangle sweeps are used to capture more information each cycle, helping the base VAR process maintain causal consistency without generating unphysical correlations. However, this does not preclude simulation of differently shaped pulses, within the approximation that such pulses produce the same effect on the device as a ramp to the same voltage at the experimental sweep rate (7 kV s^{-1}). One caveat is that repeated application of pulses with the same voltage amplitude will not have a cumulative effect on resistance states, and intermediate states must instead be programmed through application of voltage amplitudes in the RESET range. Nevertheless, the model captures the resistance variations inherent to the material system, which we expect to be comparable between alternative programming methods.

C. Implementation and Benchmarks

Using easily evaluated polynomials and matrix multiplications throughout, Synaptogen is designed for high throughput and parallelization. We recently benchmarked an implementation in the Julia programming language, comparable with the present model in terms of speed, demonstrating the practicality of simulating large-scale physical neural networks with over 10^9 weights [11]. However, due to the growing interest in simulating networks at the circuit level, efficient stochastic device models implemented in an HDL are currently highly sought after [2], [3], [4].

To suit a circuit design ecosystem and to compare speeds with alternative models, we implemented Synaptogen in the Verilog-A HDL. Special programming requirements were imposed by the adaptation to a transient model description and by the weak support for dynamic structures in Verilog-A. Furthermore, due to the discontinuities at the threshold voltages, the simulation step size was limited locally at each device threshold to aid convergence. The order of the VAR process in the Verilog-A implementation was fixed to $p = 10$.

Simulation speeds were compared with a minimalistic nonstochastic linear ion drift model (LinearDrift) as a baseline [14], as well as the more complex physics-based Jülich Aachen Resistive Switching Tool (JART) v1b variability-aware model [2]. Read speeds are also compared with randomly initialized arrays of ohmic resistances, a linearly solvable problem which gives an upper bound for the speed of the simulation framework.

We benchmarked the read and write performance for both parallel operation of M independent cells and for $\sqrt{M} \times \sqrt{M}$ crossbar arrays with resistive leads (5 Ω between every circuit node). This distinction is important because lead resistance has a strong impact on the system, but is much slower to solve due to the strongly coupled equations [15], [16], [17]. For the purpose of comparing simulation speeds between the independent device and crossbar-connected cases, the same

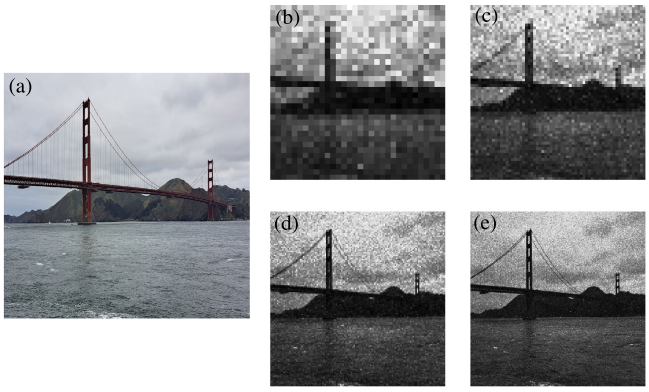

Fig. 9. For write tests, the image (a) was desaturated, resampled, and each pixel value was written as an intermediate resistance state of the individual devices in a crossbar array (0-Ω lead resistance). The results for Synaptogen-based arrays are shown, with dimensions (b) 32 × 32, (c) 64 × 64, (d) 128 × 128, and (e) 256 × 256.

applied voltage waveform was shared by rows and columns of the independent devices as though they were connected by WLs and BLs. This makes the problem equivalent to a crossbar array with zero lead resistance, but in practice is significantly faster than enforcing crossbar connectivity in the netlist.

Simulations were performed using the Cadence Spectre simulator with "moderate" settings for both the "accelerated parallel simulator" (APS) and error tolerance, running on eight (out of 18) cores of Intel Xeon Gold 6154 CPU. Square bipolar voltage pulses were applied to the WL terminals to simulate read/write operations, and the throughput in operations per second (OPS) was calculated as the number of devices involved in the read/write process divided by the total time taken for the transient analysis.

For weight programming benchmarks, differently sized arrays of devices were initialized in an LRS before writing grayscale image data (one pixel value per device) as intermediate resistance states (Fig. 9). The pixel values were linearly mapped to a suitable positive voltage range for partial RESET, and using a half-select voltage scheme [18], the voltages were sequentially applied to the corresponding cells for 1 μs. For situations where the entire array could not be written in a practical amount of time, the throughput was determined by writing a 16 × 16 subblock of devices. For readout benchmarks, 200-mV pulses were simultaneously applied to all WLs for 1 ns as current was measured at the grounded BL terminals. The results of the read and write benchmarks for all the device models are summarized in Fig. 10.

III. Results and Discussion

The described hierarchical modeling approach efficiently generates feature vectors that closely resemble the training data. This can be verified visually by comparing the time series behavior of the measurement data with the output of the generative model (Fig. 11) and also numerically by comparing the correlations between each feature (Table I). Feature distributions are very closely replicated, including the covariations in the C2C distributions between different

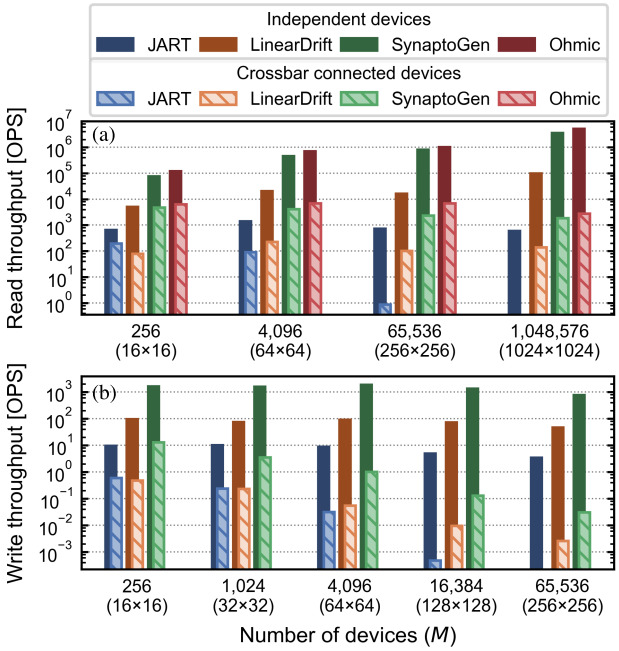

Fig. 10. Benchmarks of different Verilog-A models for (a) reading and (b) writing M independent devices and $\sqrt{M} \times \sqrt{M}$ resistive crossbars. For the largest arrays, the JART model did not terminate.

TABLE I
Correlations[1] of Measured/Generated Features

	$R_{\mathrm{H},n}$	$V_{\mathrm{S},n}$	$R_{\mathrm{L},n}$	$V_{\mathrm{R},n}$
$R_{\mathrm{H},n-1}$	0.13 / 0.13	(0.05 / 0.05)	0.05 / 0.05	(0.01 / 0.01)
$V_{\mathrm{S},n-1}$	(0.05 / 0.05)	0.17 / 0.17	(0.06 / 0.06)	(0.05 / 0.05)
$R_{\mathrm{L},n-1}$	0.04 / 0.04	(0.05 / 0.06)	0.49 / 0.49	0.13 / 0.14
$V_{\mathrm{R},n-1}$	(0.01 / 0.01)	(0.04 / 0.05)	0.13 / 0.13	0.29 / 0.32
$R_{\mathrm{H},n}$	1.00 / 1.00	(0.22 / 0.23)	0.05 / 0.05	(0.01 / 0.01)
$V_{\mathrm{S},n}$		1.00 / 1.00	(0.07 / 0.07)	(0.04 / 0.05)
$R_{\mathrm{L},n}$			1.00 / 1.00	0.18 / 0.18
$V_{\mathrm{R},n}$				1.00 / 1.00

[1] Pearson correlation coefficients (intra-cycle and with one cycle lag) were calculated for each device and averaged. All R_{H} values were logged before the calculation. Negative values are parenthesized.

devices, as well as the total marginal distributions over all the cycles and devices (Fig. 12).

For all the models and conditions, read operations were significantly faster than writes, and speeds were much higher for independent devices than for an equal number of crossbar connected devices. Synaptogen wrote at $\sim 10^3$ OPS for independent devices, but started at 13 OPS for 16 × 16 crossbars, degrading with crossbar size to only 0.3 OPS at 256 × 256. For readout, Synaptogen is competitive with simple ohmic resistive networks, reaching 60%–80% of their speed in most cases. The throughput of these read operations increased for larger numbers of devices, with 8×10^3 OPS for 256 devices and 4×10^6 OPS for 1 048 576 devices. Crossbar-connected readouts were slowed by 2–4 orders of magnitude relative to independent devices as the array size increased from 16 × 16 to 256 × 256.

Synaptogen was between 10× and 100× faster than LinearDrift for all the benchmarks, which is remarkable because LinearDrift is a very simple ordinary differential equation (ODE) formulation for which the simulator should

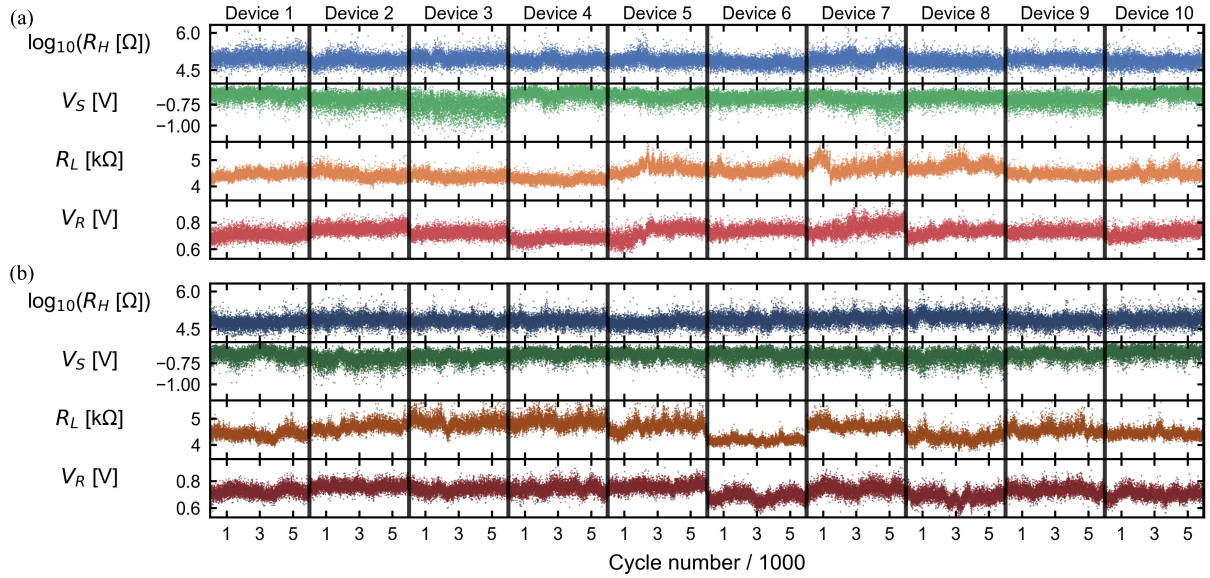

Fig. 11. Comparison between (a) measured and (b) generated feature vector time series across 6000 cycles for ten randomly selected devices. Visual inspection confirms that the model closely reproduces the variability between devices and the cycling cross-correlations.

be well-adapted. Furthermore, LinearDrift does not include C2C or D2D variability and cannot reproduce many important switching features of actual devices. Synaptogen even more significantly outperforms the JART v1b variability model, which is more closely comparable in terms of covered device behavior. Due to its complexity and implicit formulation, JART performance degrades faster than the other models as the array size grows; for JART array sizes 256 × 256 and above, not even a single write operation could be performed in a reasonable time frame. At 1024 × 1024, read operations were also impossible. For the conditions that could be simulated, operations on independent Synaptogen devices were always over 100× faster, with the speed of writes approximately 200×, and reads reaching 6000× those of JART. For the resistive crossbar simulations, Synaptogen was between 10× and 100× faster for 64 × 64 and smaller arrays, and between 100× and 10 000× for larger arrays.

While analog circuit simulations are indispensable for time-domain analysis of systems, they face intrinsic speed limitations due to the computation necessary at each time step to converge on solutions to large systems of nonlinear differential equations. Furthermore, circuit simulations lack the regular memory access pattern typical of ML, and their data-dependent control flow makes it challenging for them to take advantage of GPU acceleration. Even with dramatic speed increases over competing models, simulations in Cadence Spectre with Synaptogen-based synapses still have clear practical limitations for training and inference with fully connected neural network layers. Table II shows the time necessary to write pretrained weights and to perform a single-layer inference operation according to our benchmarks. Many operations can be completed in well under a second, while others (such as writing to large resistively coupled crossbars) can take a considerable amount of time (hours or days). This may limit the ability to simulate entire analog ML chips at once, but investigations are possible the level of individual cores, comprising, e.g., 256 × 256 devices [19], [20].

TABLE II
ESTIMATED TIME REQUIRED FOR NEURAL NETWORK OPERATIONS USING SYNAPTOGEN WEIGHTS IN THE CADENCE SPECTRE SIMULATOR

Layer size	Weight Initialization		Inference	
	Independent	Crossbar	Independent	Crossbar
16×16	160 ms	20 s	3.4 ms	54 ms
32×32	660 ms	4.9 min		
64×64	2.3 s	1.1 h	9.2 ms	1.0 s
128×128	13 s	1.5 d		
256×256	1.4 min	25 d	82 ms	28 s
1024×1024			300 ms	9.6 min

As modern ML networks commonly exceed millions of weights, the benchmarks reported here highlight the need to extend the device model's applicability to larger scales. Therefore, while the Verilog-A implementation provides compatibility with circuit design tools, we also implemented Synaptogen in the Julia programming language. The internal operation is the same for both the models, while the latter achieves orders of magnitude (>3 orders for read, >6 orders for write) higher speeds by avoiding transient calculations and through parallel execution on CPUs and GPUs [11]. This high-level model is well-suited for integration with analog compute-in-memory software frameworks to incorporate more realistic device behavior without compromising on speed. The existing frameworks commonly interface with well-known ML libraries such as PyTorch and achieve speeds within 5× of conventional floating-point training and inference [21], [22], [23], [24], [25], [26], [27], [28].

Finally, because Synaptogen focuses on directly reproducing measured quantities in a generalized way, the same concept

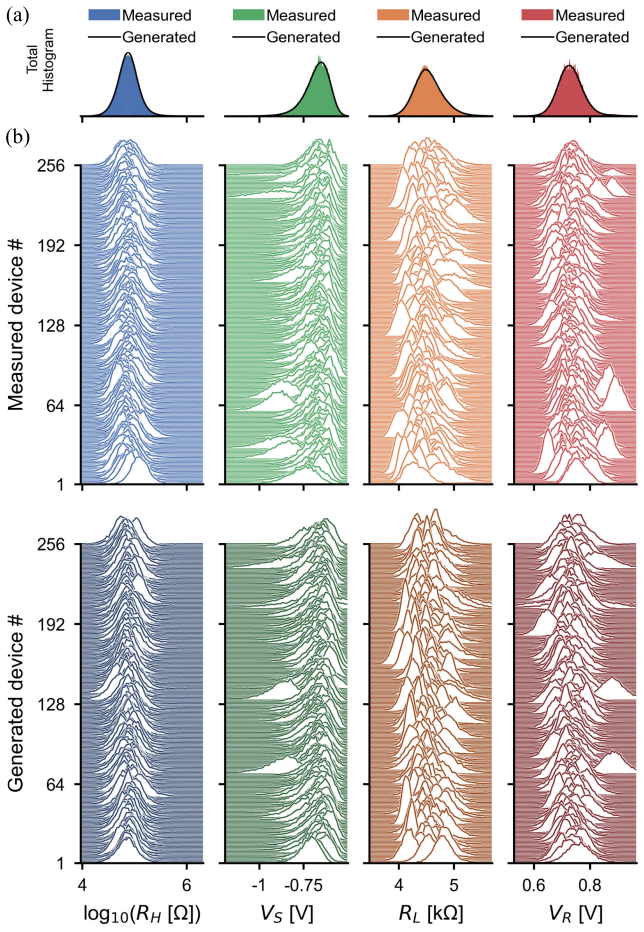

Fig. 12. Comparison between the measured and generated distributions for 6000 cycles of 256 devices. Total marginal feature distributions, including every cycle of every device, are compared in (a), and the ridgeline plots (stacked histograms) in (b) show the distributions conditioned on device number. The C2C distributions and their D2D covariances are very closely replicated by the generative model.

can also be adapted to different memory technologies based on other physical effects, such as ferroelectric, magnetic, correlated electron, and phase change memories [29], [30], [31], [32]. In this work, we selected four specific switching characteristics to encode the (I, V) cycling behavior. However, the generative framework allows for the incorporation of additional features derived from experimental measurements. Based on new statistical data, various functional forms, transition dynamics, temporal dependencies, and underlying stochastic processes can be integrated as required.

IV. Conclusion

In this work, we developed a generative compact model for resistive memories that seamlessly adapts to statistical measurements of real-world devices through an automated training procedure. We show that the model closely captures the nonlinearity, cross-correlations, and both the C2C and D2D variability of electrical data measured on integrated ReRAM devices. While an equivalent model can be used in a high-level programming domain for larger scale simulations, here we demonstrate its use in analog circuit simulation of 1T1R arrays. The implemented circuit-level model operates orders of magnitude faster for reading and writing compared with other compact models, and we demonstrate crossbar programming (256 × 256 devices) and readout (1024 × 1024 devices) at scales which exceed what was previously possible in the analog circuit simulation domain.

Code Availability

The Verilog-A compact model and its Julia counterpart are available on GitHub (https://github.com/thennen/synaptogen) and archived in Zenodo (https://zenodo.org/doi/10.5281/zenodo.10942560).

References

[1] C. Nail et al., "Understanding RRAM endurance, retention and window margin trade-off using experimental results and simulations," in *IEDM Tech. Dig.*, Dec. 2016, pp. 451–454, doi: 10.1109/IEDM.2016.7838346.

[2] C. Bengel et al., "Variability-aware modeling of filamentary oxide-based bipolar resistive switching cells using SPICE level compact models," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 12, pp. 4618–4630, Dec. 2020.

[3] V. Ntinas et al., "A simplified variability-aware VCM memristor model for efficient circuit simulation," in *Proc. 19th Int. Conf. Synth., Model., Anal. Simul. Methods Appl. Circuit Design (SMACD)*, Jul. 2023, pp. 1–4, doi: 10.1109/SMACD58065.2023.10192107.

[4] J. Reuben, M. Biglari, and D. Fey, "Incorporating variability of resistive RAM in circuit simulations using the Stanford–PKU model," *IEEE Trans. Nanotechnol.*, vol. 19, pp. 508–518, 2020.

[5] Z. Jiang et al., "A compact model for metal–oxide resistive random access memory with experiment verification," *IEEE Trans. Electron Devices*, vol. 63, no. 5, pp. 1884–1892, May 2016.

[6] F. M. Puglisi, L. Larcher, A. Padovani, and P. Pavan, "Bipolar resistive RAM based on HfO_2: Physics, compact modeling, and variability control," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 2, pp. 171–184, Jun. 2016.

[7] N. Gong et al., "Signal and noise extraction from analog memory elements for neuromorphic computing," *Nature Commun.*, vol. 9, no. 1, p. 2102, May 2018, doi: 10.1038/s41467-018-04485-1.

[8] V. Agrawal et al., "Subthreshold operation of SONOS analog memory to enable accurate low-power neural network inference," in *IEDM Tech. Dig.*, Dec. 2022, pp. 2171–2174.

[9] R. Naous, M. Al-Shedivat, and K. N. Salama, "Stochasticity modeling in memristors," *IEEE Trans. Nanotechnol.*, vol. 15, no. 1, pp. 15–28, Jan. 2016, doi: 10.1109/TNANO.2015.2493960.

[10] R. Picos, J. B. Roldan, M. M. A. Chawa, F. Jimenez-Molinos, and E. Garcia-Moreno, "A physically based circuit model to account for variability in memristors with resistive switching operation," in *Proc. Conf. Design Circuits Integr. Syst. (DCIS)*, Granada, Spain, Nov. 2016, pp. 1–6, doi: 10.1109/DCIS.2016.7845383.

[11] T. Hennen et al., "A high throughput generative vector autoregression model for stochastic synapses," *Frontiers Neurosci.*, vol. 16, Aug. 2022, Art. no. 941753, doi: 10.3389/fnins.2022.941753.

[12] A. Grossi et al., "Fundamental variability limits of filament-based RRAM," in *IEDM Tech. Dig.*, Dec. 2016, pp. 471–474, doi: 10.1109/IEDM.2016.7838348.

[13] J. D. Hamilton, *Time Series Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 1994.

[14] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, "TEAM: ThrEshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Jan. 2013.

[15] T. P. Xiao, B. Feinberg, J. N. Rohan, C. H. Bennett, S. Agarwal, and M. J. Marinella, "Analysis and mitigation of parasitic resistance effects for analog in-memory neural network acceleration," *Semiconductor Sci. Technol.*, vol. 36, no. 11, Nov. 2021, Art. no. 114004, doi: 10.1088/1361-6641/ac271a.

[16] A. Chen, "A highly efficient and scalable model for crossbar arrays with nonlinear selectors," in *IEDM Tech. Dig.*, Dec. 2018, pp. 3721–3724, doi: 10.1109/IEDM.2018.8614505.

[17] D. Joksas and A. Mehonic, "Badcrossbar: A Python tool for computing and plotting currents and voltages in passive crossbar arrays," *SoftwareX*, vol. 12, Jul. 2020, Art. no. 100617, doi: 10.1016/j.softx.2020.100617.

[18] A. Chen, "Analysis of partial bias schemes for the writing of crossbar memory arrays," *IEEE Trans. Electron Devices*, vol. 62, no. 9, pp. 2845–2849, Sep. 2015, doi: 10.1109/TED.2015.2448592.

[19] M. Le Gallo et al., "A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference," *Nature Electron.*, vol. 6, no. 9, pp. 680–693, Aug. 2023, doi: 10.1038/s41928-023-01010-1.

[20] W. Wan et al., "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, no. 7923, pp. 504–512, Aug. 2022.

[21] Z. Zhu et al., "MNSIM 2.0: A behavior-level modeling tool for processing-in-memory architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 11, pp. 4112–4125, Nov. 2023, doi: 10.1109/TCAD.2023.3251696.

[22] C. Lammie, W. Xiang, B. Linares-Barranco, and M. R. Azghadi, "MemTorch: An open-source simulation framework for memristive deep learning systems," *Neurocomputing*, vol. 485, pp. 124–133, May 2022, doi: 10.1016/j.neucom.2022.02.043.

[23] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, "DNN+NeuroSim V2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2306–2319, Nov. 2021, doi: 10.1109/TCAD.2020.3043731.

[24] M. J. Rasch et al., "A flexible and fast PyTorch toolkit for simulating training and inference on analog crossbar arrays," in *Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2021, pp. 1–4.

[25] S. Roy, S. Sridharan, S. Jain, and A. Raghunathan, "TxSim: Modeling training of deep neural networks on resistive crossbar systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 4, pp. 730–738, Apr. 2021.

[26] S. Jain, A. Sengupta, K. Roy, and A. Raghunathan, "RxNN: A framework for evaluating deep neural networks on resistive crossbars," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 2, pp. 326–338, Feb. 2021.

[27] T. P. Xiao, C. H. Bennett, B. Feinberg, M. J. Marinella, and S. Agarwal. *CrossSim: Accuracy Simulation of Analog In-Memory Computing*. Accessed: Jun. 6, 2024. [Online]. Available: https://github.com/sandialabs/cross-sim

[28] D. Kireev et al., "Metaplastic and energy-efficient biocompatible graphene artificial synaptic transistors for enhanced accuracy neuromorphic computing," *Nature Commun.*, vol. 13, no. 1, p. 4386, Jul. 2022, doi: 10.1038/s41467-022-32078-6.

[29] H. Ning et al., "An in-memory computing architecture based on a duplex two-dimensional material structure for in situ machine learning," *Nature Nanotechnol.*, vol. 18, no. 5, pp. 493–500, May 2023, doi: 10.1038/s41565-023-01343-0.

[30] H. Liu et al., "Dynamics of spin torque switching in all-perpendicular spin valve nanopillars," *J. Magn. Magn. Mater.*, vols. 358–359, pp. 233–258, May 2014, doi: 10.1016/j.jmmm.2014.01.061.

[31] Y. Zhou and S. Ramanathan, "Mott memory and neuromorphic devices," *Proc. IEEE*, vol. 103, no. 8, pp. 1289–1310, Aug. 2015, doi: 10.1109/JPROC.2015.2431914.

[32] M. L. Gallo and A. Sebastian, "An overview of phase-change memory device physics," *J. Phys. D, Appl. Phys.*, vol. 53, no. 21, p. 213002, Mar. 2020.