Development of a Short-Term to Long-Term Supervised Spiking Neural Network Processor

Tony James Bailey, *Member, IEEE*, Andrew J. Ford, *Member, IEEE*, Siddharth Barve, *Member, IEEE*, Jacob Wells, *Member, IEEE*, and Rashmi Jha, *Member, IEEE*

***Abstract*— We report a realization of a mixed-signal, supervised spiking neural network (SNN) architecture utilizing short-term plasticity in synaptic resistive random access memory (RRAM). First, the development of a phenomenological RRAM SPICE model is discussed based on the previously reported device data. Then, the design of the neuroprocessor's architectural components are described. To achieve learning using the synaptic RRAM devices, a novel method of backpropagation in hardware SNNs is presented using the proposed gated bidirectional amplifier circuit. A method to perform quantized weight transfer between the short-term memory (STM) and long-term memory (LTM) is also proposed, allowing transient associated memories to be stored and used repeatedly. The neuroprocessor is able to associate input digits with class labels, transfer learned associations to a long-term register array, then recall all digits when presented again. The low operational power of 13.7 mW makes this system ideal for future integration onto embedded systems with limited available energy. Finally, the neuroprocessor's tolerance to input noise and internal device failure was measured to be 14% and 15%, respectively. We believe that this work provides significant insight into the development of hardware SNNs in addition to providing a framework to achieve more complex STM to LTM interactions in the future.**

***Index Terms*— Heteroassociative memory, neuromorphic, resistive random access memory (RRAM), short-term memory (STM), spiking neural network (SNN), spike timing-dependent plasticity (STDP), VLSI.**

I. Introduction

DESPITE significant progress in the development of neuromorphic processors, there is still a distinct gap between the processing capabilities of a biological brain and neuromorphic processors. It is well recognized that artificial intelligence (AI) algorithms are memory-intensive and developing neuromorphic architectures based on conventional memory devices for accelerating AI faces memory wall challenges. This issue makes a compelling case for developing novel synaptic devices and brain-inspired architectures [1]. However, majority of neuromorphic approaches using emerging synaptic devices use nonvolatile states (NVS) in resistive random access memory (RRAM) and analogs to long-term potentiation (LTP) of synapses to train and perform pattern matching tasks [2].

One major limitation of NVS/LTP is that it fails to capture the time series dynamics of states which imposes limitations in performing more advanced cognitive tasks such as those performed in biological short-term memories (STMs): real-time signal comparisons, differentiations, and life-long learning. At this juncture, it is important to understand how some of the lesser understood dynamics of the brain, such as short-term plasticity (STP) and working memory (WM), can be incorporated in neuromorphic approaches to bridge this gap [3]. Interestingly, RRAM devices also manifest short-term states (STSs) [4] and some reports have previously explored STS in RRAM for filtering noise [5]. While an interesting start in this area, noise filtering is just one of the many attributes of STP in the brain. There is a plethora of research indicating how STP in the brain plays a critical role in signal comparison, multimodal time series signal fusion, and decision-making tasks [3]. In this work, we investigate an application of the STS in RRAM to emulate WM for computation in our learning algorithm. We explore the utility of the state decay of the STS for synaptic weight depression during the training process. Our approach works to combine both volatile and nonvolatile memory rather than focusing on NVS/LTP done in previous works. We believe that the use of STS in RRAM for training provides advantages in terms of: 1) continuous analog weights that depresses gradually as governed by a decay constant rather than abruptly and 2) elimination for intentional weight depressing pulses that can help reduce the programming power and programming circuit complexity.

One issue that has to be reconciled with a volatile memory is how to utilize it for useful computation once learning has taken place. The learned memories will be quickly lost due to the forgetful nature of these networks. For humans, information from our environment is constantly flooding into our senses and our WM decides what to focus our attention on and what is important enough to remember [6]. The role of an attentional unit is accomplished in our network with the inclusion of a supervisor, which enforces what information gets stored into the STM. To reinforce these memories into long-term memory (LTM), we propose a method of "remembering" by transferring the STM weights to a static LTM network. Inferencing and recall could then take place at a later date via this new LTM. To keep the digital memory storage overhead

Manuscript received March 8, 2020; revised June 15, 2020; accepted July 18, 2020. This work was supported by the National Science Foundation under awards ECCS 1556294, CNS 1556301, and SHF 1718428. *(Corresponding author: Rashmi Jha.)*

The authors are with the Electrical Engineering and Computer Science Department, University of Cincinnati, Cincinnati, OH 45220 USA (e-mail: baileytj@mail.uc.edu; fordaj@mail.uc.edu; barvesh@mail.uc.edu; wells2jd@mail.uc.edu; jhari@ucmail.uc.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2020.3013810

low in the LTM, we propose a thresholding scheme that quantizes the analog weights of the STM. This method has been previously utilized in deep neural networks to reduce the size of the weight matrices needed for classification [7], [8]. For numerical digit detection, we were able to reduce the STM weight complexity into a simple binary storage using D-flip flops (DFFs).

The development of our proposed mixed-signal neuromorphic architecture using STS in RRAM required the incorporation of both device and system level advances. With this system, we were able to achieve real-time learning for a supervised classification problem on a reduced data set using a spiking neural network (SNN) architecture. The functionality of this system was validated through its ability to learn and then differentiate digits based on their pixel representations. The decay-based nature of STM also allows for the mixed STM+LTM architecture to adapt to global changes in the input data while yielding significantly lower circuit complexity when compared to a purely LTM-based implementation.

The manuscript first discusses modeling methodology used to create an HSPICE compatible synaptic RRAM model using previously reported empirical RRAM data. Then, the design of the SNN architecture and all of its components are discussed. The performance of the SNN architecture is then discussed in terms of its ability to learn and its power consumption. Finally, the SNN is applied as a heteroassociative memory and its tolerance to both input pattern noise and internal device failures is measured.

This work presents the first complete VLSI realization of a mixed-signal, supervised SNN architecture utilizing STP in RRAM. We believe that our method of backpropagation in hardware SNNs using the proposed gated bidirectional amplifier (gBDA) circuit is a novel approach to implement learning and in-place weight updates (WUs). We also believe that our method of STM to LTM weight transfer and storage has not been shown previously. The proposed neuroprocessor architecture progresses the field of neuromorphic computing by presenting clear implementation details needed to design and integrate supervised learning methods into hardware SNNs. The remainder of this article is organized as: Section II—short-term synaptic modeling of RRAM, Section III—STOM to LTM memory architecture, Section IV—network performance, and Section V—heteroassociative memory.

II. Short-Term Synaptic RRAM Modeling

A. RRAM Synaptic Measurements

RRAM is a class of memristive devices that utilize the defect chemistry in thin-film, metal–insulator–metal structures to change their resistive state via programming signals [9]. We previously fabricated and characterized a candidate device, $SrTiO_3$ (STO)-based RRAM, that contains the following properties for STP: low conductance, unidirectional potentiation, up to 10× change in conductive state, and constant regularization via state decay [4]. The cross-sectional stack of the RRAM devices is shown in Fig. 1(a) and a fabricated crossbar array of these devices is shown in Fig. 1(b).

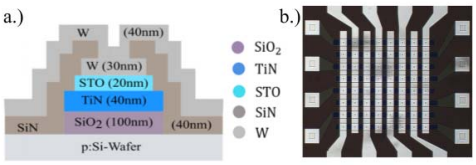


Fig. 1. (a) Fabricated $SrTiO_3$-based RRAM stack. The fabrication process of our modeled devices is described in [4]. (b) Top-down microscopy of an 8×8 array of 60 μm $\times$ 60 μm fabricated devices.

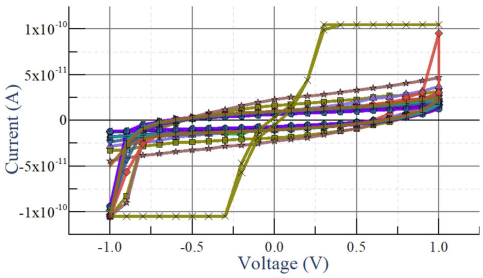


Fig. 2. Measured initial I–V sweep from -1 to $+1$ V of an 8×2 synaptic array. Note that while one device appeared to be nearly shorted in its initial state (due to fabrication variance), its sneak current contribution to other devices is small.

The measured initial conductance of an 8×2 array of the devices is shown in Fig. 2.

The manifestation of STS and STP in STO RRAM is likely due to the competing drift and diffusion mechanics of defects within the insulating film when stressed under a sufficiently high applied bias [10]. The backdiffusion of these defects creates a constant state decay within the device, allowing for short-term depression while a large enough drift current increases the device's state leading to short-term potentiation. These effects were previously captured by changing the spike rate of a spike train and measuring the resultant relative change in conductive state shown as the "experimental" trace in Fig. 3 [4]. A spike train creates an effective rms voltage via the relationship

$$V_{\text{rms}} = A \cdot \sqrt{D} \tag{1}$$

where A is the spike amplitude and D is the duty cycle of the spikes. By relating V_{rms} to the measured relative state change, we were able to curve fit an exponential relationship with $R^2 = 0.9$. The relationship between V_{rms} and relative change in conductance provides an easy way to model spike rate based STP in devices that has not been captured in other work.

B. Spike Timing Dependent Plasticity (STDP)

In neurobiological systems, information processing and "learning" occur due to the dynamic interactions in a network of neurons through connections called synapses. In the aggregate, these interactions lead to the emergence and formation of associative memories that relate actions or thoughts to some input stimulus. A popular, biologically driven, way to represent the relationship between neuron's firing and the corresponding synaptic modulation is called STDP. It states that synaptic weight change between a presynaptic neuron and postsynaptic

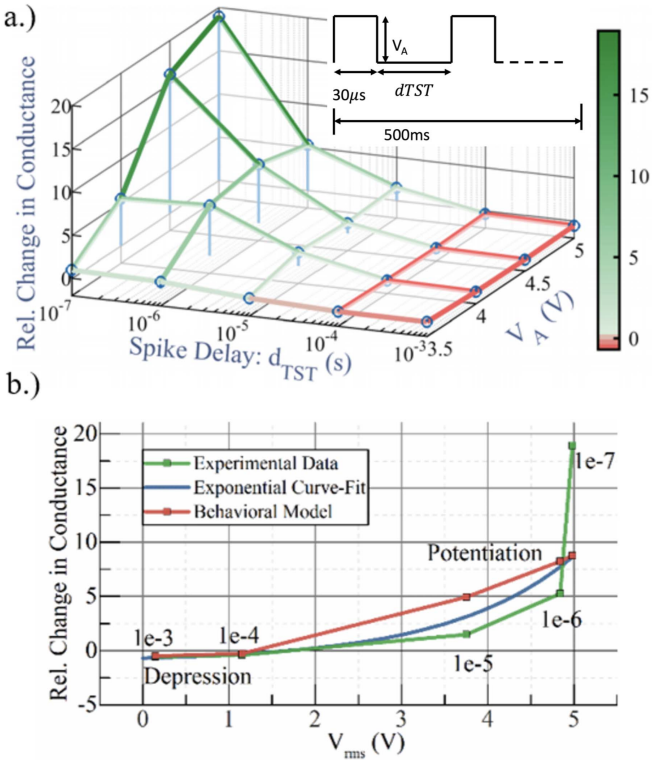


Fig. 3. (a) Mesh plot of the relative change in conductance averaged over three samples due to varying V_A and spike delay d_{TST}. Inset: illustration of the 500-ms pulse train with 30-μs pulses with an amplitude of V_A and delay of d_{TST}. (b) V_{rms} applied across the synapse and the resultant relative change in conductive state for the experimental, curve fit, and behavioral model data. Experimental data are an average of three trials and were extracted from Fig. 3(a) using a spike amplitude of 5 V. Labels show the spike delay (in seconds) between 30-μs-wide spikes (i.e., modulating their duty cycle). In both fabricated and modeled cases, potentiation and depression can be achieved by the devices, allowing for synaptic behavior to occur.

neuron is proportional to the nearness of the two spikes as well as the temporal order of their firing. If the preneuron fires first, it "predicts" the postneuron firing and will thus strengthen the synapse. Conversely, if the postneuron fires first, it is not being influenced by the preneuron and therefore weakens the synaptic weight [11]. This is represented in the following STDP model:

$$\Delta w_{ji} = \sum_{f=1}^{N}\sum_{n=1}^{N} W\left(t_i^n - t_j^f\right) \tag{2}$$

$$\begin{cases} W(x) = A_+ e^{-\dfrac{x}{\tau_+}}, & x > 0 \\ W(x) = -A_- e^{\dfrac{x}{\tau_-}}, & x < 0 \end{cases} \tag{3}$$

where Δw_j is the change in weight for neuron j for a particular time-step. t_i^n, t_j^f represent the time-stamps for the preneurons and postneurons' spikes, respectively, for N total spikes, indexing each by f and n [12], [13]. $W(x)$ represents the learning window with time constants τ_+, τ_-. A_+ and A_- are parameters of the learning window that can also depend on the current value of w_j [12]–[18]. In a synaptic RRAM device, the synaptic weight is bounded by some operational limits, i.e., $w^{\min} < w^j < w^{\max}$ yielding

$$A_+(w_{ji}) = (w^{\max} - w_{ji}) \cdot \eta_+ \tag{4}$$

$$A_-(w_{ji}) = (w_{ji} - w^{\min}) \cdot \eta_- \tag{5}$$

with η_+ and η_- as fit parameters.

The STDP model is useful in describing the behavior of physical devices because this relationship emerges due to simple voltage addition across the synapse. Larger spike overlap causes the net voltage applied to be larger, causing a larger change in synaptic weight. Conversely, temporally far apart spikes do not have significant constructive feedback and thus do not affect the synaptic weight strongly. Destructive interference does not occur due to only positive spikes being applied across the device from the perspective of the preneurons—a key difference between this architecture and traditional STDP.

In our synaptic RRAM devices, we experience a few differences from classical STDP. First, only positive voltages above a certain threshold have a significant impact on the state change. The TiN/STO/W stack gives rise to an asymmetric barrier which leads to a diode-like characteristic resulting in unidirectional potentiation. The net effect resembles spike coincidence detection since the temporal order does not matter. Mathematically, this modifies the learning window, $W(x)$, which can now be represented in a Gaussian form

$$W(x;\, \mu, \sigma) = A(w_{ji}) \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6}$$

with μ, σ as hyperparameters and $A(w_{ji})$ as the state-dependent parameter. Second, depression in our RRAM devices is due to a constant state decay as well as a hypothesized strain-induced decay when voltages are applied [4], [19]. In both cases, exponential decay terms are added to the weight of the synapse, changing the WU equation as follows:

$$\begin{cases} w_{ji}(t) = (w_{ji}(t-1) + \Delta w_{ji}) \cdot e^{-\frac{t_1}{T_1}}, & \text{bias applied} \\ w_{ji}(t) = (w_{ji}(t-1) + \Delta w_{ji}) \cdot e^{-\frac{t_2}{T_2}}, & \text{bias removed} \end{cases} \tag{7}$$

where t_1 is the time over which a bias is applied, T_1 is a fit parameter for the strain-induced decay, t_2 is the time elapsed since the bias has been removed, and T_2 is the continual state-decay time constant. This constant regularization via weight decay causes the device to experience STP. By combining the potentiation and depression mechanics, a net increase or decrease can be achieved based on the coincidence of forward propagating presynaptic spikes and backward propagating teaching signal-gated postsynaptic spikes across the synaptic RRAM device shown in Fig. 4. Empirically, this was determined by varying the spike rate and measuring the net relative change in conductance over a fixed 0.5-s time-frame as discussed in Section II-A.

C. Synaptic RRAM Device Modeling

There are several simulation methodologies, such as Euler's method, that could be used to solve the aforementioned dynamical systems of equations, but we chose to use the circuit-based linear equation solver SPICE in this report. In a circuit simulation environment, the potentiation and depression

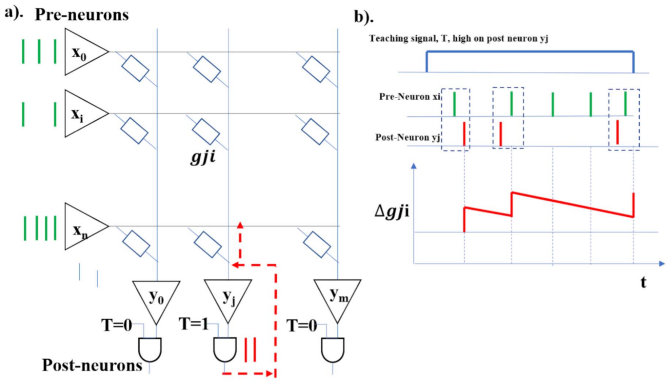


Fig. 4. (a) Behavioral schematic of the architecture during training of the network. Training is done in the STM implemented by the RRAM array. (b) Hebbian-like learning method is utilized to potentiate the synaptic weights. The potentiation of the synapses at the junction are due to the input neuron firing coincidentally with the teaching signal-gated output neuron. This will form an association between the firing of that output neuron and the firing of those input neurons. The synapse will also depress between potentiating pulses due to decay of STS.

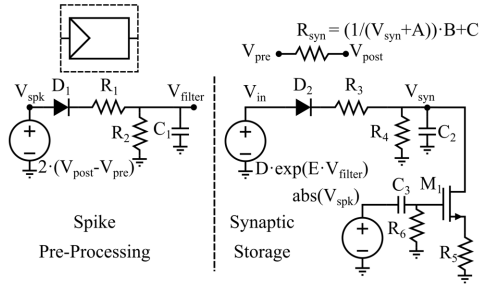


Fig. 5. Behavioral HSPICE model that mimics the STP dynamics of synaptic RRAM devices. The spike preprocessing module applies a low-pass filter to the positive contributions of the differential spike signals applied across *Rsyn*. The synaptic storage module mimics the state retention and synaptic potentiation/depression mechanics. Combined, the state change dynamics at the output resistance, *Rsyn*, matches the curve fit of the empirically measured synaptic RRAM devices' state change with an $R2$ value of 0.98.

mechanics are represented by components that are modeled by differential equations. Conveniently, since RLC circuits form a basis for linear system equation modeling, a phenomenological model can be developed that mimics the observed short-term potentiation and depression mechanics while simultaneously leveraging the SPICE simulator to perform the linear equation matrix solving. This method has the added benefit of allowing the ability to incorporate synthesizable static CMOS circuitry to interface with the RRAM array and test network level architectures. The full circuit used to represent the STP is shown in Fig. 5. The RC integrator consisting of R_3, R_4, and C_2 mimics the potentiation, depression, and storage of the resistive state of the fabricated devices. The time constants of this charging–discharging circuits represent the measured potentiation and depression rates from the empirical devices. The final "state" or weight of the device is proportional to the voltage, V_{syn} across C_2. The output of the model is a behavioral resistor with a resistance, R_{syn}, proportional to $1/V_{\text{syn}}$. The diodes, D_1 and D_2, ensure that only positive spikes cause potentiation to occur in the device. M_1 is used to represent the strain-induced decay by enhancing the decay rate. While

TABLE I
BEHAVIORAL SYNAPTIC RRAM: MODEL PARAMETERS

Parameter	Value	Parameter	Value
A	0.5	R_2	100kΩ
B	6426871	C_2	0.8nF
C	691000	R_3	50MΩ
D	0.14	R_4	10GΩ
E	0.9	R_5	20MΩ
C_1	1nF	C_3	0.1nF
R_1	10kΩ	R_6	50kΩ

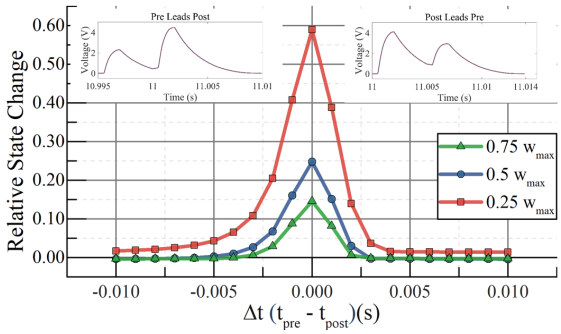


Fig. 6. Simulated unidirectional STDP curve for our $SrTiO_3$-based synaptic devices recorded at different starting state (w) between fully depressed ($w = 0$) and fully potentiated ($w = w_{\text{max}}$). Net potentiation is achieved when prespike and postspike are near each other and net depression is achieved when prespike and postspike are far enough apart. Inset waveforms show the net signal across the device when the prespike leads the post and when the postspike leads the pre.

a spike is being applied, additional charge will drain off of C_2 through the M_1 branch. C_3 and R_6 form a high-pass filter that only allows this effect to be active while the spike is applied. V_{in} has an exponential dependence to the differential voltage $(V_{\text{pre}} - V_{\text{post}})$ to match the observed relationship between V_{rms} and the empirical change of state in Fig. 5. Parameters B and C were picked to match the measured device conductance range, while parameter A was fit based on the average initial measured conductance ($R_{\text{syn}_{\text{max}}}$) using the conductance of the functioning initial devices from Fig. 2 measured at 0.5 V of the forward voltage sweep. Table I summarizes the values for the parameters used in the model.

The HSPICE model was then tested using the same test bench that acquired the potentiation and depression rates of the empirical devices described in Section II-A to produce the "behavioral model" curve in Fig. 5. The behavioral model fit to the curve fit with $R^2 = 0.98$, thus closely matching empirical data. The behavioral SPICE model was then used to generate Gaussian-like STDP curves by recording the relative change of state due to two neuron spikes at varying temporal distances shown in Fig. 6. The shape of these curves depends on the starting state of the synapse. If fully depressed, it is easy for the synapse to increase its state, but if it is fully potentiated, it is difficult to continue increasing its state and will likely result in a net depression. This relationship reflects the $A_{+/-}$ parameters of the learning window from (4) and (5).

III. STM TO LTM MEMORY ARCHITECTURE

SNNs are biologically inspired neuromorphic computing systems that can utilize the STDP learning rule to form

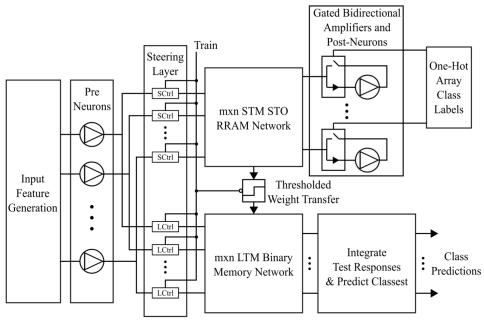


Fig. 7. Architectural level diagram of the SNN neuroprocessor with heterogeneously integrated short-term and long-term learning paradigms.

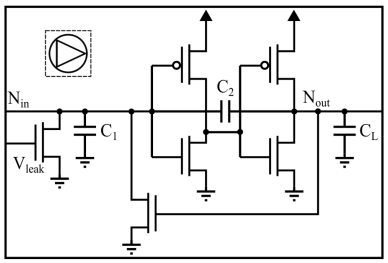


Fig. 8. Transistor-level diagram of Mead's leaky integrate and fire Axon-Hillock neuron [20].

correlations by performing in-memory computations that modify the conductance of synaptic devices [11]. SNNs using STDP are typically used for unsupervised learning, but our implementation presents a simple method to add supervision using gating via combinational logic, allowing the system to solve classification problems. We chose to implement a mixed analog–digital VLSI design strategy because neuron spiking events and control units are inherently digital systems, but other functionality such as resistance sensing and signal aggregation utilize analog design methodologies. This approach allows for real-time signal propagation through the RRAM devices, while also using static CMOS logic to handle the management of digital spikes. The system level schematic of our neuroprocessor architecture consists of an input feature layer, presynaptic neuron layer, steering layer, STM network, gBDA and postsynaptic neuron layer, weight threshold function, LTM network, and an inference layer as shown in the system level diagram in Fig. 7.

A. Input Feature Generator and Presynaptic Neuron Layers

The input feature generator is responsible for sensing external signals and converting them into currents that drive the presynaptic neurons. The neuron layer itself transduces these signals into spikes that are steered into the proper network during the training and testing phases. In our implementation of digit recognition, the input feature generator represents the pixels encoded as a current proportional to the intensity of each pixel.

Neuron circuits convert an input current into an output voltage spike if the aggregated charge exceeds a threshold. This relationship was realized using Mead's Axon-Hillock circuit [20] shown in Fig. 8. We added an additional NMOS

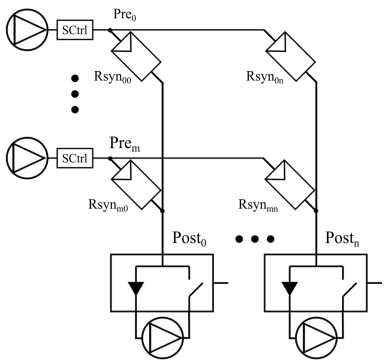


Fig. 9. STM connectivity between synaptic RRAM crossbar and the presynaptic neurons, signal steering layer, and gBDA circuits.

to add a parasitic resistance at the N_{in} node, causing it to leak when no input is applied.

B. STM Network

The STM network consists of three components: the fabricated synaptic RRAM array, a layer of gBDAs, and a layer of postsynaptic neurons. Our network implementation uses a single layer of postsynaptic neurons, similar to a perceptron. There was therefore one gBDA and postsynaptic neuron for each class. The synaptic RRAM array is fabricated in a crossbar configuration for a dense memory representation of a neural network's weight matrix. The STM crossbar array was connected to the signal steering and gBDA layers as shown in Fig. 9.

The gBDAs are one of the most critical and novel circuit designs in this architecture because they allow for the forward and backward propagation of spikes while also incorporating supervised learning during training. The component level diagram of the gBDA circuit shown in Fig. 10 consists of a forward and backward propagating circuit. As forward propagating spikes are applied across the RRAM, a current is established proportional to the state of each individual device. These currents are summed in each column of the network and fed into a current amplifier. This amplifier gains the $\approx$1-nA signals up to $\approx 1-\mu$A signals that can drive the postsynaptic neurons.

In biological neurons, external, inhibitory signals suppress the spiking of the neurons. In our supervised learning scheme, the labels act as inhibitory signals to the postsynaptic neurons, silencing all but the labeled neuron, thus allowing the input patterns to be encoded in different parts of the network. This is accomplished in the backward pass circuit using the Boolean function

$$F = \overline{N_{\text{out}} \cdot \text{Label.}} \tag{8}$$

The backpropagating teaching signal spikes are then shifted to the negative voltage range V_{ss} to 0 (where $V_{\text{ss}} = -V_{\text{dd}}$) so that the proper signal addition occurs across the RRAM device (i.e., so overlapping signals on the prenode and postnode constructively add two positive voltages together). This was accomplished using a level shifting circuit to shift the logic values to $0 = V_{\text{ss}}$ and $1 = 0$ V. For backpropagation,

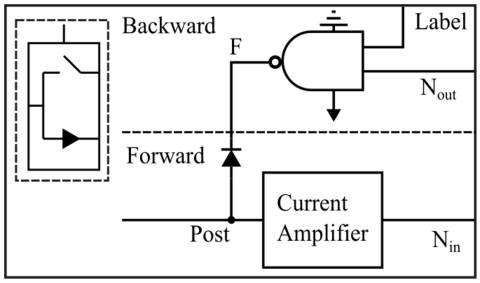


Fig. 10. Component-level diagram of the gBDA consisting of the forward and backward subcircuits (gBDA). NAND gates force a strong digital 1 and 0, making it ideal for resistive memory driving.

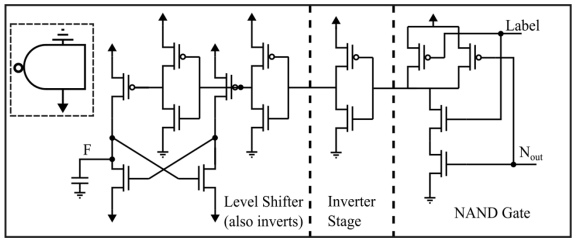


Fig. 11. Transistor-level diagram of the backward propagation circuit realized as a level-shifted NAND gate. The input swing is from $0-V_{dd}$, while the output swing at F is V_{ss} to V_{dd}. The positive half of this larger swing is then clipped by the series, reverse-biased diode in Fig. 9 which will only allow the negative swing to propagate.

negative spikes are therefore represented by logic 0 and no spikes are represented by logic 1, hence the use of a NAND gate to achieve the correct spike propagation logic. The transistor-level circuit to realize this logic is shown in Fig. 11. The output of the level shifter is then used to force a negative spike onto the postnode when a spike appears at its input. When no teaching signal spikes are backpropagating, the level-shifter should be left floating to allow the forward propagating spikes to pass through the current amplifier without contention at the postnode from the combinational circuitry. To prevent contention, a reverse-biased diode was added in a series configuration between the output of the level-shifter (F in Fig. 11) and the postnode. This creates an effective open-circuit when $F \geq 0$ V, but will force negative backpropagating spikes onto the postnode by forward-biasing the diode. Using the gBDA circuit block, we were able to successfully achieve forward and backward propagation as well as supervised learning using a simple 2-terminal synaptic RRAM.

C. Signal Steering Layer

The signal steering layer performs two functions. First, it routes presynaptic neuron spikes to either the STM or LTM network, helping to reduce the power consumption of active elements throughout the network. Second, during the weight transfer as described in Section III-D, they serve as control units to properly read and transfer weights from the STM to LTM network. One control unit is needed for each row of both the LTM and STM networks.

For the STM control unit (SCtrl), the former functionality is realized using transmission gates (T-gates) connected to the presynaptic neurons and the rows of the RRAM array. The

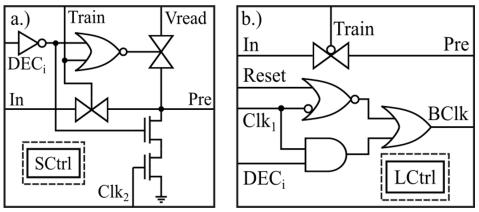


Fig. 12. (a) Gate-level diagram of the SCtrl unit that propagates training spikes through the STM and enables reconfiguration between training and resistive weight reading phases of operation. (b) Gate level diagram of the LCtrl unit that propagates inferencing spikes through the LTM network and controls the clock to shift in thresholded data once it has propagated to the register input.

T-gates are enabled when Train $= 1$, allowing signals to propagate through them during the training process. For the weight transfer operation, each row in the STM must be indexed one at a time and reconfigured to apply a small read voltage (V_{read}) across one row of RRAM devices. To accomplish this, a second T-gate is used that is connected to the V_{read} line and one row of the RRAM array. It is controlled by logic that turns on if its row is indexed and Train $= 0$. In this way, the ith row will be reconfigured to force V_{read} onto it when it is indexed by DEC_i. A pull-down network consisting of two NMOS was also added to each SCtrl unit to pull the prenodes back to 0 V after they were used. Otherwise, the prenodes would be left floating at V_{read} after each read operation and their resistance would contaminate future rows' measurements by acting as parallel resistances. The full gate-level diagram of the SCtrl units is shown in Fig. 12(a).

The LTM control unit (LCtrl) shown in Fig. 12(b) utilizes T-gates as well for signal routing, but it is enabled when Train $= 0$. This allows spikes to be routed to the LTM during testing and inferencing. For each row in the LTM network, the LCtrl unit also is needed to gate the register clocking so that data are only shifted in once the RRAM resistance has been measured and is available at the input to the register. The combinational logic shown in Fig. 12(b) will allow the register in the ith row to be clocked for a reset signal or if the row is indexed by DEC_i.

D. Thresholded STM to LTM Transfer

To enable the long-term storage of the associations that the STM network learned, a weight sensor and an indexing circuit were designed to reconfigure the system to allow it to read the state of each device and transfer it into long-term storage. In doing so, we quantize the resistive state of the RRAM devices in the STM because their continuous, analog resistance is converted into a discrete, digital value. In sparse patterns, a simple binary thresholding scheme can be sufficient to detect if a feature is present in the input data. We present a method to perform weight quantization on-chip for a binary representation. The resistance measurement output can also be easily connected to an ADC and wrote to larger registers with higher precision.

The sensing of the resistive weight was accomplished using transimpedance amplifiers (TIAs) connected to each column of the RRAM array. In each column, when V_{read} is applied

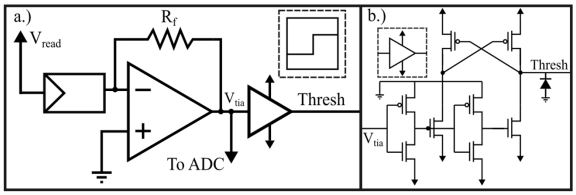


Fig. 13. (a) WTU realized using an analog transimpendance amplifier fed into a level-shifting buffer. A higher precision reading could also be taken using the "To ADC" tap-out point. (b) Transistor-level implementation of the level-shifting buffer circuit, shifting a $-V_{ss} - 0$ V swing into a $0 - V_{dd}$ swing.

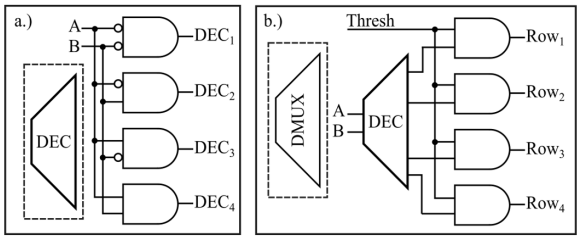


Fig. 14. (a) WTU realized using an analog transimpendance amplifier fed into a level-shifting buffer. A higher precision reading could also be taken using the "To ADC" tap-out point. (b) Transistor-level implementation of the level-shifting buffer circuit, shifting a $-V_{ss} - 0$ V swing into a $0 - V_{dd}$ swing.

to a particular row, a current is generated proportional to the resistive state of the indexed device and fed into the TIA. The amplifier then generates an output voltage that is proportional to the resistive state of the RRAM device. The feedback resistor (R_f) in the TIA was tuned to exceed the digital transition threshold when $R_{syn} = 0.5 \cdot R_{synmax}$. The TIA's output voltage, V_{tia}, will now be negative and must be level-shifted back into the range of $0 - V_{dd}$ to be wrote to a register. A level-shifting circuit, similar to the one used in backpropagation, is used to accomplish both the level shifting and the thresholding operations in a single circuit component. The first inverter will threshold its output to 0 V (logic 1) if the measured voltage is above the inverter's switching threshold (i.e., close to 0 V), else it will threshold its output to V_{ss} (logic 0). The level shifting occurs in the next stage to change the output swing to V_{ss} to V_{dd} as well as another logic inversion. A shunt-connected diode is attached to the Thresh output to clip the negative swing, yielding a 0 V $- V_{dd}$ output swing. The net result at Thresh will be a logic 0 if V_{tia} is less than the threshold and a logic 1 if V_{tia} is greater than the threshold. The full weight thresholding unit (WTU) for one column is shown in Fig. 13(a) and the transistor-level diagram of the level-shifting circuit is shown in Fig. 13(b). Interestingly, the weight thresholding operation can occur in parallel for all RRAM in a row since the columnwise current paths terminate at different WTUs.

To perform the indexing operation, an upcounter was used in conjunction with a decoder and demultiplexer (DMUX). The counter was enabled once Train $= 0$ and was then incremented from $0 - m$ where m is the number of rows. At each clock cycle, the counter output is fed into a decoder to select the appropriate row in the STM and LTM networks via the SCtrl and LCtrl units, respectively. The decoder was realized using an array of AND gates as shown in Fig. 14(a).

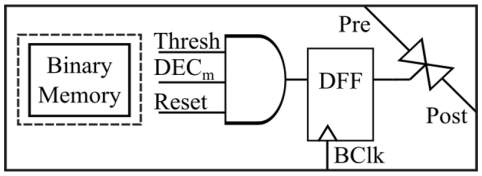


Fig. 15. Component-level diagram of the binary memory storage unit used for inferencing. Note that the DMUX's AND gate is included at each node with an extra input for synchronous reset of the registers.

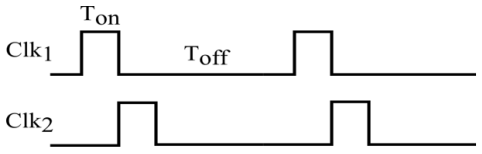


Fig. 16. Asymmetric, phase shifted clocking scheme used to transfer the quantized weights from the STM to LTM. The phase offset is equal to T_{on}.

The DMUX was used to route the thresholded information into the proper register row-wise in the LTM. Since the decoder and DMUX select bits are both controlled by the counter and a DMUX consists of a decoder and a series of and gates [as shown in Fig. 14(b)], the decoder can be reused to act as part of the realization of the DMUX. The AND gate of the DMUX is incorporated into each LTM binary memory element as shown in Fig. 15.

E. Weight Transfer Clocking

A phase-shifted, asymmetric clocking scheme was used to accommodate the propagation delay from when a row was configured to the time when it is settled at the output of the TIA. The clock sequence, shown in Fig. 16, consists of two clocks that have much longer "off" times than "on" (i.e., $T_{on} < T_{off}$) and are phase-shifted by T_{on}. Once Train is toggled to 0, Row_0 will get pulled up to V_{read} and the row's weights will immediately begin to propagate through the WTUs. After an initial duration of T_{off}, Clk_1 cycles its "on" portion and store the state of the Row_0's Thresh values into the LTM Row_0 registers. Then, the "on" portion of Clk_2 will increment the counter and pull down all rows except Row_1 to 0 V. Row_1 is then pulled up to V_{read} and its weights begin to propagate through the WTUs for T_{off}. The process then repeats as Clk_1 causes Row_1's registers to shift in their contents. This process will operate for total of 35 cycles, successfully transferring all weights into the LTM.

F. LTM Network

To store the binary state of the learned weights, several memory solutions could be used. Using a more conventional 2-state RRAM device, the state can be encoded as a high resistance state (HRS) or low resistance state (LRS) representing logic 0 and 1. Fabricationwise, this solution becomes expensive because another crossbar would have to be stacked on-top and properly isolated from the Synaptic RRAM array. Additionally, more peripheral circuitry, Set/Reset voltage lines (that are not necessarily V_{dd} or V_{ss}), and sequential control modules also would have to be implemented to administer

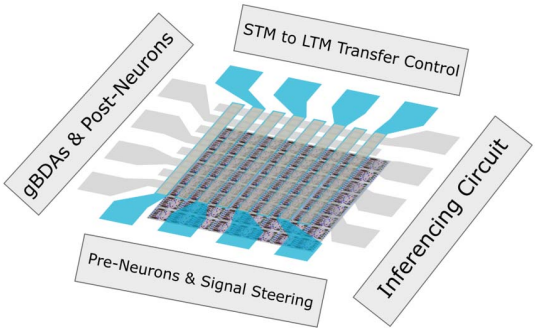


Fig. 17. Proposed layout for additive fabrication of $SrTiO_3$ based RRAM on top of the CMOS architecture. The DFF array would be less than or equal to the area of the RRAM array, so the footprint would be limited by the feature size of the added RRAM layer.

Set/Reset pulses to the devices. Flash could also be used as a more easily integrated CMOS compatible device to store the information, but would still require additional fabrication steps and sequential control modules to apply the set and clear operations.

A simple CMOS implementation was used instead that is compatible with the spiking neuron inputs. The binary memory element was realized by using a T-gate controlled by the D output of a DFF as shown in Fig. 15. One terminal of the T-gate is connected to the appropriate LCtrl unit in the signal steering layer and the other terminal is connected columnwise, forming a crossbar configuration. If $D = 0$, the T-gate would be disconnected, and if $D = 1$, the presynaptic neuron would be connected to the column line. Each column in the LTM network is connected to an external ADC or a simple RC integrator used here to integrate all incoming spike events. The predicted class for a given sample during inferencing is therefore the one-hot encoded column with the largest response.

G. Layout and Additive Manufacturing

One benefit of using the $SrTiO_3$-based RRAM as memory elements is that the fabrication process is entirely compatible with CMOS backend and frontend processes. The experimentally fabricated crossbars from Section II-B sit on top of a SiO_2 layer, which would be analogous to the top isolation oxide of a back-end-of-the-line process. All of the underlying CMOS components would be fabricated and routed together on-chip. To interface with the RRAM array, leads from the T-gates would be pulled out to via pads that are left exposed. The RRAM additive fabrication process would then begin with the deposition of the TiN bottom electrode, followed by the $SrTiO_3$ layer, and finally the W top electrode using a similar process as described in [4]. An efficient layout of the CMOS components would be to mirror the STM and LTM crossbars vertically, causing the steering layer to route the spikes up or down, thus minimizing the signal propagation lengths. The other peripheral circuitry would then be placed on the other three sides of the rectangular crossbar array, producing a compact circuit design. A block diagram of the proposed layout is shown in Fig. 17.

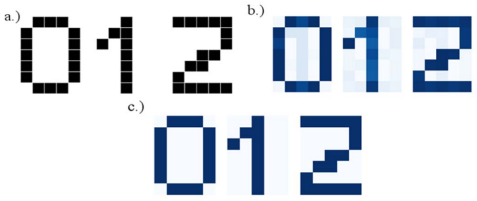


Fig. 18. (a) 7×5 pixel representation of the digits 0, 1, and 2 used to train the STM network. (b) STM encoded representations for each digit class learned during the training phase of the network's operation. Each pixel represents one synapse's conductive state normalized from 0 to 1. (c) LTM weights using the same methodology for each digit class after thresholding operation.

IV. Network Performance

Network-level simulations utilized Synopsys' HSPICE circuit simulator, however, due to computational limitations in HSPICE, the size of the circuit could not be scaled to use the full MNIST feature size and full data set. For all simulations, the ASU PTM 130-nm technology node [21]–[25] was used because it supports the 3.3-V rails required to apply a net potential difference of around 5 V across the synaptic RRAM devices for programming [4]. This platform was chosen for its simultaneous ability to implement realistic peripheral circuitry and its ability to model the dynamical RRAM devices using simple capacitive and resistive elements to approximate experimental device behavior using the aforementioned model in Section II-C. A behavioral model was also simulated in Python to demonstrate the network's performance on the full MNIST data set to accommodate for computational restrictions in HSPICE.

A. Learning

Training the network using the small digit representation shown in Fig. 18(a) demonstrates the ability for the network to associate input features with labels and "learn." To account for scalability, the subsequent power calculations are described in terms of a single circuit block and should scale approximately linearly with the number of input features and number of classes. Training took place by presenting each digit as a 35×1 array of pixels encoded as currents proportional to the value of the pixel intensity. Since there were 35 input features and three classes, the simulated STM and LTM networks both contained 35×3 memory arrays. While each digit was presented to the STM, a one-hot encoded label (i.e., "100" for class 0 and "010" for class 1 corresponding to the proper class enforced the supervision within the network. After training, the encoded states of the synaptic devices were extracted, normalized between 0 and 1, and remapped into heat maps shown in Fig. 18(b). Each representation of the digit is retained in the synapses corresponding to each of the three classes and the representation of each digit is isolated to only one neuron's synapses due to the applied labels. Interestingly, even though the RRAM devices are connected in a large passive array, the signal isolation via the supervised labels largely remove the effects of crosstalk between elements.

Fig. 18(c) shows a heat map of the corresponding thresholded values in the LTM after the thresholding operation

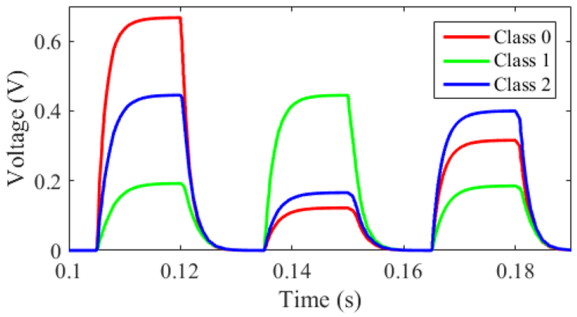


Fig. 19. Inferencing responses across the *RC* integrators in the LTM after training occurred. The largest response correctly corresponded to a 0, 1, and 2 as the samples are reapplied to the network.

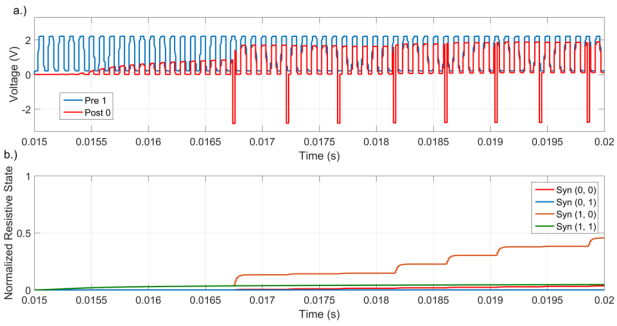


Fig. 20. Example of weight change occurring due to the backpropagation of spikes across the synapse at the prenode and postnode. (a) Spiking patterns for preneuron 1 and postneuron 0 for the first 5 ms of the "0" sample. (b) Normalized state of RRAM devices at locations (0, 0), (0, 1), (1, 0) and (1, 1).

occurred. Given the size and complexity of the given data, the binary quantization was all that was required to accurately store the pixel representation of the digits into the LTM. If this problem were extrapolated to more complex data sets, the learned STM weights could be quantized to a higher precision register for more nuanced inferencing. The architecture's predicted class responses during testing are shown in Fig. 19, demonstrating that this system can recall the learned digit representations with 100% accuracy with wide margins between the digits.

Fig. 20(a) shows the forward and backward propagating spikes during the application of the "0" sample. Note that for the "0," Pre0 is not firing, while Pre1 is firing [seen by the pixels in Fig. 16(a)]. Fig. 20(b) shows the normalized resistive state change for four devices in different situations. The locations represent the physical (x, y) location where x represents the pixel number and y represents the class number. The (0, 0) device is getting only postsynaptic spikes, (0, 1) is getting no spikes, (1, 0) is getting both presynaptic and postsynaptic spikes, and (1, 1) is only getting presynaptic spikes. Significant state change only occurs in the (1,0) device when there is direct coincidence of spikes. Some spikes (such as the second postspike) do not cause a change because the corresponding prespike is at 0 V. Devices (0, 0) and (1, 1) increase slightly due to only one set of spikes, but their state is significantly less than the reinforced (1, 0) device. The (0, 1) device's state had little to no change since no spikes were applied across it.

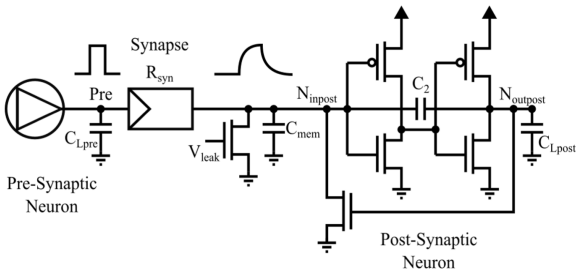


Fig. 21. Circuit diagram representing the MAC operation. One spiking event on the presynaptic neuron generates a square output voltage waveform, consuming dynamic power from the 0-1 transition. This spike is propagated and accumulated at the N_{inpost} node, which charges at a rate of $R_{\text{syn}} \cdot C_1$.

Note that the magnitude of presynaptic and postsynaptic spikes in Fig. 20(a) are asymmetric. The RRAM devices, especially at high initial resistance values, tended to potentiate even in the presence of smaller voltage spikes. This could become troublesome if the presynaptic voltage spikes caused a significant amount of unwanted potentiation without the reinforcing postsynaptic spikes. The supply voltage for the preneurons was therefore heuristically reduced to $0.76 \cdot V_{\text{dd}}$ to prevent significant unwanted potentiation from occurring.

B. Power Consumption

Power consumption was analyzed by examining the contributions from each architectural component during the period in which they were active. The total power consumption was determined by extracting the sum of the individual energy consumptions of each layer and dividing this by the total operation time. Total leakage power was extracted from HSPICE by operating the architecture with no stimulus and measuring the average power draw. These results are described below under bullets 1 to 4 and summarized in Table II.

1) Training: For the STM network during training, power consumption occurs whenever spiking events occur. The power consumption can be split into three separate operations: the multiple and accumulate (MAC) operation, the activation function (AF), and the WU. All other components only contribute to leakage power during training.

To compute the MAC operation, a simplistic case ignoring peripheral circuitry shown in Fig. 21 is used consisting of a Synapse with resistive state R_{syn} connected to a presynaptic and postsynaptic neuron. Since the active portion of the neurons are effectively a digital buffer, the power consumed per spiking event (i.e., a digital transition from 0-1-0) is equal to the digital dynamic power consumed during the 0-1 transition. This is represented using the following power equation:

$$P_{\text{dyn}_{\text{MAC}}} = V_{dd}^2 \cdot C_{L_{\text{pre}}} \cdot SR_{\text{pre}} \tag{9}$$

where $C_{L_{\text{pre}}}$ is the capacitive load at the prenode, SR_{pre} is the spike-rate of the preneuron. In an Elmore delay model with the preneuron as the source and $C_{L_{\text{pre}}}$ as the destination, C_1 in the postneuron does not share any path with $C_{L_{\text{pre}}}$, therefore the charging of $C_{L_{\text{pre}}}$ is independent of the resistive value, R_{syn}, of the RRAM device. The value of R_{syn} does

TABLE II
COMPONENTWISE POWER AND ENERGY CALCULATIONS

Circuit Block	Power (P_{block})	Instances	Samples	Duration	Energy (E_{block})
Pre-Neurons (MAC)	379μW	35	6	15ms	1.19mJ
SCtrl (MAC)	271μW	35	3	15ms	427μJ
Post-Neurons (AF)	100μW	3	3	15ms	13.5μJ
gBDA (WU)	830μW	3	3	15ms	112μJ
Threshold Control	58.8μW	1	1	35μs	2.06nJ
WTU	6.41mW	3	1	35μs	0.67μJ
Leakage	4.31mW	1	1	180ms	720μJ

however, impact the charge time constant of the N_{inpost} node and thus affects the spike rate of the postsynaptic neuron.

To compute the MAC power consumption using HSPICE, the average power was extracted by a single presynaptic neuron over the duration of one sample. To compute the worst case total energy contributed by the preneurons performing the MAC operation, this power was multiplied by the total number of preneurons (35), the duration of the applied sample (15 ms), and, assuming that all neurons fired for all samples, each neuron would fire for three samples. Since each spike also passes through the SCtrl unit during training, we also extracted the total power contribution due to a single SCtrl units while an input sample was applied. The scaled SCtrl energy contribution for the entire architecture was calculated similar to the presynaptic neurons.

Next, the AF operation occurs when the postsynaptic neuron decides whether or not to fire. Power is therefore only consumed when the postneuron actually fires. The power calculation for this operation is therefore represented by the same dynamic power consumption per spike using the SR of the postneuron, that is

$$P_{\text{dyn}_{\text{AF}}} = V_{dd}^2 \cdot C_{L_{\text{post}}} \cdot SR_{\text{post}} \tag{10}$$

where SR_{post} is the spike rate of the postsynaptic neuron.

To scale the extracted power from HSPICE, there are three postneurons that each spike for each of the three, 15-ms training samples (i.e., they spike for each sample, but the backpropagation is gated by the gBDA). Since the postneurons fire less than the preneurons (since it takes multiple preneuron spikes to trigger a postneuron spike), the power consumed by the postneurons will also be less. This can be seen in Table II where $P_{\text{dyn}_{\text{AF}}} = 0.26 \cdot P_{\text{dyn}_{\text{MAC}}}$.

For the WU operation, the postneuron's spike is backpropagated and applied across the postnode via a combinational logic gate. Since spikes are only propagated if the postneuron fires, $\text{SR}_{\text{WU}} = \text{SR}_{\text{post}}$. $C_{L_{\text{WU}}}$ will likely be different however due to the diffusion capacitance at the load nodes differing between the two circuits, yielding the following power equation:

$$P_{\text{dyn}_{wu}} = V_{dd}^2 \cdot C_{L_{wu}} \cdot \text{SR}_{\text{post}}. \tag{11}$$

To scale the extracted power from HSPICE, there are three gBDAs and each would allow backpropagating teaching signal spikes during one of the three applied samples. It will therefore draw power for one, 15-ms training sample. Since the NAND functionality is implemented in the first stage of the combinational teaching signal backprop circuit shown in Fig. 11, the internal state of this gate will not change unless the applied label voltage allows the gate to track the postneuron's response.

2) Weight Transfer: The weight transfer phase occurs once between training and inference. During this time, the TIAs are reconfigured to be quiescently biased and thus has a constant power draw. The power consumption for this operation was divided into the digital threshold control circuitry and the WTU unit consisting of the TIA. In HSPICE, the power was recorded for the aggregated threshold control circuit during the 35-μs weight transfer process. The WTU was recorded over the same time period, but the energy consumption was also multiplied by 3 since there are three WTUs, one for each class. In the context of the relatively slow spike-rate used for these particular synaptic RRAM devices, the total energy consumed of both circuit blocks was. 02% of the total energy contribution and was nearly negligible.

3) Inference: During the inference phase, the STM and weight transfer circuits are shutoff and only contribute leakage power. In the LTM, the T-gates in both the LCtrl unit and binary memory elements act as passive elements and do not contribute to dynamic power consumption (similar to the RRAM devices in the STM). We also used a simple, passive RC integration circuit to integrate the spike responses at each column. Therefore, the only active components drawing power at this point are the presynaptic neurons, which have a power consumption identical to the previously defined MAC operation in (9). To compensate for this in our energy calculation, we assumed the worst case instance where all 35 preneurons fired for all three, 15-ms long test samples (i.e., six samples total).

4) Total Power: Total power consumption (P_{total}) during the operation of the neuromorphic processor was determined by taking the summation of each circuit block's energy (E_{block}) consumed during the operation of the system from Table II divided by the total operation time (T_{total}), that is,

$$P_{\text{total}} = \frac{\Sigma E_{\text{block}}}{T_{\text{total}}}. \tag{12}$$

The total energy consumed by the system was 2.47 mJ over 180 ms, therefore the total power consumed by the system was 13.7 mW. Due to the relatively low-power nature of this architecture, we envision the use of such a system to be integrated onto an embedded, low-power application to potentially incorporate real-time neuromorphic processing on edge embedded devices.

C. RRAM Characteristic Optimization

One important design question to ask when trying to maximize the performance of the SNNs is to identify how the RRAM devices affects operational behavior. From a device designer's perspective, we would like to know what defines an ideal device so that constraints and goals can be placed when trying to optimize the physical design of RRAM.

From the power analysis of the architecture, two important observations can be made. First, based on the breakdown of the energy contributions in Table II, about 48% (1.19/2.47 mJ) of the total energy was consumed by just the preneurons. This illuminates the importance of focusing on low-power designs for the neuron components in SNNs. Since dynamic power has a quadratic dependence on the supply voltage, a significant source of power reduction could occur if the spike voltage can be reduced. In STO-based RRAM, the onset of resistive state change is likely due to the E-field (E) applied across the oxide layer [4], [10], [19]. Since $V = E \cdot t_{ox}$, where t_{ox} is the STO thickness, an equivalent E-field can be applied if t_{ox} and V are proportionally reduced to maintain the same field driven potentiation/depression behavior. Since most modern digital circuitry operates with V_{dd} at or below 1 V (i.e., 0.33% of the 3.3 V used here), t_{ox} would likely need to be reduced by about 2/3. For the 20-nm STO film used in [4], this would mean reducing the thickness to 6–7 nm, which is still a comparable thickness to those used with other insulating film RRAM devices.

Second, the max operational speed is directly proportional to the resistance ranges of the devices because the SR_{post} of the postsynaptic neuron is directly dependent on the resistance value of the synaptic RRAM devices connected to it. If the devices' resistance is too large, the time to charge the post-neuron could become much slower than the speed of the data being applied through the system. This would lead to many patterns that would be applied but would not generate any postsynaptic spikes, preventing real associations from forming. In our architecture, a current amplifier was added to the gBDA in Fig. 10 to increase the postsynaptic firing rate to match the speed of the incoming data. This circuit adds quiescent power draw however, so ideally it should be removed to minimize the system power.

From this observation, the design of the RRAM device properties therefore depends on the speed of the data to be processed. If a neuromorphic processor such as the one presented here are embedded locally on sensing devices, the STM network needs to react about as fast as the rate of incoming data. In general, too fast of a device response would lead to the network to overreact and overfit data, while too slow of a response would miss important information and underfit the data.

Once a timeframe is determined by the application, the RRAM device characteristics can ideally be tuned for that operational speed. The resistance range should be chosen to achieve the postsynaptic time constant comparable to the desired timeframe. In STO-based RRAM, this could be accomplished by n-type doping the STO film, which generates an excess of e-carriers, increasing the conducive range of the device [26]. Next, the potentiation and depression rate of the resistive state would have to be tuned to operate within the timeframe of the problem. In STO, one potential way could be to modulate the oxygen vacancy concentration via doping [26]. This would hopefully increase the speed of the drift/diffusion mechanics that are postulated to cause the state change within the STO film, but empirical testing is needed to validate these proposals.

TABLE III
PATTERN ROBUSTNESS

Pixels Flipped	3%	6%	11%	14%	18%
Accuracy	100%	100%	100%	100%	66.6%

V. HETEROASSOCIATIVE MEMORY

One application of the proposed SNN architecture would be as a heteroassociative memory that associates an input concept with an output concept via a correlation weight matrix [27]. These correlation matrices can recall previously stored memories while being robust to input noise and local device failure. This type of network differs from gradient descent-based backpropagation networks since no explicit cost function determines the WUs. Instead, the weight change is due to the association (or correlation) between input patterns and output classes. The performance metric of an associative memory is the rate of noise tolerance allowed before memories cannot be accurately recalled. We tested our neuroprocessor's ability to act as a heteroassociative memory by measuring its robustness in two contexts: recalling noisy patterns and tolerance to broken devices.

A. Robustness to Noisy Patterns

This test was conducted by first training the STM network on the same digit patterns as Fig. 18(a) in the same order and without noise. For testing, a percentage of each test pattern's bits were flipped, creating a noisy image. The test patterns were then applied to the LTM network and the accuracy was recorded when 3%, 6%, 11%, 14%, and 18% of the bits in each test sample were flipped.

Table III shows the resultant accuracy of the architecture's inferencing for each trial. Even after 14% of noise was introduced in each test sample, the system was still able to accurately recall all of the stored memories. There are only three samples for the three classes, so the accuracy must be a multiple of 33.3%. Even though the toy digit problem is relatively small, this is still significant since the corresponding available memory density is also small. Regardless of the memory density, each class can only be differentiable if its pixel representation contains at least one unique pixel. A pixel is considered unique for a class if it is not "1" for the other two classes. By inspection, the "0" image has seven unique pixels, the "1" only has 4, and the "2" has 6. Since the "1" class is the least differentiable, by randomly flipping more than about 4 bits, it seems reasonable to start having overlap between the stored images. Since the system failed with 6 bits flipped (i.e., 18%), it is tolerant to an even higher degree of noise.

B. Robustness to Broken Devices

This test was accomplished by replacing a percentage of the RRAM array with "broken" devices, represented as an unchanging resistor with $R = R_{syn_{max}}$. These devices effectively act as if their weight was stuck at "0." The train and test samples were unchanged and kept in the same order for this experiment to eliminate order bias, while 2%, 5%, 10%, 15%, and 20% of the RRAM devices were broken in each trial.

TABLE IV
ARRAY ROBUSTNESS (HSPICE)

Broken RRAM	2%	5%	10%	15%	20%
Accuracy	100%	100%	100%	100%	66.6%

Fig. 22. Normalized STM encoded representations for each digit class learned during the training phase with 10% of the devices randomly broken throughout the RRAM array. Note that system was still able to accurately recall each digit when the normal pattern was presented again during inference.

Table IV shows the results of this experiment and Fig. 22 shows the learned heat map when ten random devices were broken. This image clearly demonstrates the robustness inherent in the system. For detrimental effects to occur, the device has to be broken for a class where patterns exist, which is relatively sparse, and it has to break in a unique pixel location to tighten the inferred class responses. Tolerance to 15% of the system broken shows that this system is fairly robust to online device failure.

C. Full-Scale Network Simulation Results and Discussion

While HSPICE offered precise insight into device behavior, computational limitations rendered full-scale simulations of the MNIST data set impractical. Higher level behavioral simulations were conducted to obtain data on full images with all ten digits using a network with 784 input neurons, for each pixel, and ten output neurons, for each digit, producing a 784×10 weight matrix. Parameters include upper bound frequency F_H (Hz) and lower bound frequency F_L (Hz) for the spike trains generated from each pixel, spike train length l, weight decay percentage d, and potentiation rate p, with the following constants:

$$F_L = 1 \text{ Hz}, \quad F_H = 510 \text{ Hz}, \quad l = 1020, \quad W_{\text{damp}} = 0.2. \quad (13)$$

F_L and F_H were set low to reduce training time, and l was chosen to provide a high enough resolution for the network to distinguish between a pixel intensity of 254 versus 255. The weight change was then determined by the equation

$$w_{t+1} = xy(w_t + p) + (1 - xy)(1 - d)(w_t) \quad (14)$$

where x is the input spike train for the corresponding pixel, y is the teaching signal for the corresponding neuron, w_t is the current weight, and w_{t+1} is the next weight. This equation shows that weights get potentiated when the input spike and teaching signal coincide, consistent with Hebbian learning, captured by devices in (7) and illustrated in Fig. 4. The weights then decay at any other time due to the STP of the RRAM devices.

1) Network Accuracy: Peak test accuracy was recorded during 1 epoch of training on the MNIST data set and recorded in Table V. From Table V, optimal values selected for p and d were $1E-2$ and $5E-7$, respectively. Peak recorded accuracy was 80.4%—acceptable for a single-layer perceptron. The second to last row shows the effects of utilizing non-decaying RRAM in this architecture which would eventually force all devices to LRS. The last row shows behavior with non-decaying synaptic devices which reset to a high resistive state upon depression pulse, like current memristive device behavior [28].

TABLE V
HIGH LEVEL SIMULATION ACCURACY

Weight Decay d \ Weight Potentiation p	5E-01	1E-01	5E-02	1E-02	5E-03	1E-03	5E-04	1E-04	5E-05	1E-05
5E-03	68%	62%	67%	60%	62%	59%	55%	16%	14%	15%
1E-03	76%	78%	71%	73%	76%	76%	73%	52%	43%	16%
5E-04	73%	77%	75%	73%	74%	74%	76%	72%	57%	14%
1E-04	68%	74%	77%	72%	74%	78%	74%	74%	72%	66%
5E-05	70%	74%	77%	78%	76%	73%	75%	75%	73%	72%
1E-05	69%	71%	76%	80%	80%	73%	75%	72%	74%	28%
5E-06	68%	73%	77%	76%	77%	71%	74%	76%	59%	26%
1E-06	71%	75%	80%	76%	80%	72%	73%	68%	68%	22%
5E-07	71%	75%	79%	80%	78%	74%	75%	69%	60%	32%
1E-07	68%	73%	77%	77%	78%	73%	75%	66%	58%	24%
5E-08	67%	73%	77%	76%	79%	74%	72%	68%	58%	26%
1E-08	71%	75%	78%	78%	78%	74%	73%	69%	66%	24%
5E-09	72%	73%	76%	78%	78%	73%	76%	70%	65%	28%
1E-09	72%	71%	78%	77%	80%	72%	72%	65%	58%	30%
5E-10	68%	75%	75%	78%	80%	73%	74%	67%	60%	24%
1E-10	72%	73%	77%	78%	78%	74%	74%	66%	58%	18%
0E+00	62%	73%	73%	76%	76%	72%	75%	65%	54%	16%
0wRST	16%	16%	16%	16%	18%	15%	13%	15%	14%	16%

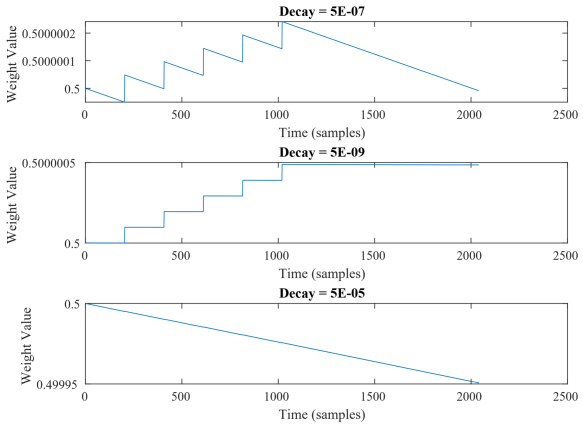
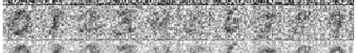


Fig. 23. Example of weight change dependence on coincidence of the teaching signal and input spike train spikes with an input frequency of 5 Hz, a potentiation constant of $1E-2$, and decay constant of (a) $5E-7$, (b) $5E-9$, and (c) $5E-5$.

2) Coincidence of Teaching Signal and Spike Trains: The weight changes shown in Fig. 23 demonstrate the potentiation and decay occurring as the input spike train coincides with the teaching signal given the optimal potentiation and varying decay parameters. The behavior shown in each graph in Fig. 23 from the higher level simulations is an accurate representation of those properties observed in Fig. 20(b) during circuit level simulations. Fig. 23(a) shows a balanced synaptic potentiation and decay for a winning neuron, Fig. 23(b) shows a dominant potentiation, and Fig. 23(c) shows a dominant decay.

3) Development of Synaptic Weight Matrices: The evolution of the feature maps as the network trains is important when considering how the network is learning, and can be seen

TABLE VI
EVOLUTIONS OF SYNAPTIC WEIGHT VALUES

Images Trained	Accuracy	Feature Map
0	9%	
250	41%	
500	59%	
750	53%	
1,000	61%	
5,000	68%	
10,000	66%	
20,000	70%	

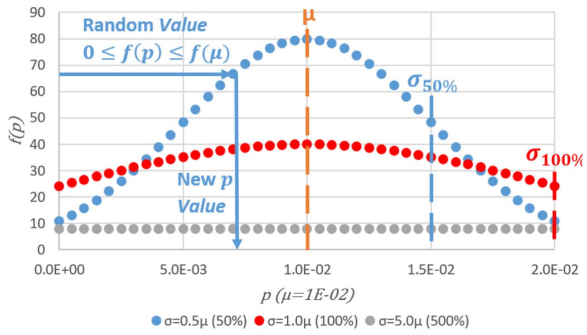


Fig. 24. Gaussian variation of the synaptic weight potentiation parameter where a standard deviation is 50%, 100%, and 500% of μ (the optimal p parameter).

in Table VI. With training on 40% of MNIST images, the most evident synapses used in the network potentiate significantly, while unused synapses decay substantially.

4) Device Variability: Components such as memristors could potentiate and decay variably. The network's tolerance to stochastic device behavior was simulated by varying p and d according to the Gaussian distribution function

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{15}$$

where μ is the previously determined optimal value for p or d, and σ is a percentage of μ. When an input image is propagated forward through the network, a random number $f(x)$ is chosen between 0 and $f(\mu)$; the peak value of the Gaussian distribution function, allowing the evaluation of the inverse Gaussian distribution function

$$x = \mu \pm \sigma \sqrt{2\ln\left(\frac{1}{f(x)\sigma\sqrt{2\pi}}\right)} \tag{16}$$

where x is the new value for p or d. Fig. 24 is a diagram showing the process, at 50%, 100%, and 500% variation. Results from device variability can be found in Table VII. Accuracy decreases as the parameter deviation increases. 5%–20% variance can be expected in most devices, whereas this network could readily handle 100% variance with a 3% reduction in accuracy.

5) Robustness: Another important aspect of an architecture is its ability to withstand broken or stuck memory bits. This could occur due to device failure or damage during fabrication or usage. To simulate, a portion of the weight matrix values were randomly set at 1 or 0 throughout training and inferencing, and observations are recorded in Table VII. Accuracy does not begin to substantially decrease until 20% of memory fails—which is unlikely under normal usage.

TABLE VII
ARRAY ROBUSTNESS (HSPICE)

Parameter Deviation	0%	50%	100%	500%	1000%	2500%	5000%
Accuracy	80%	79%	77%	63%	40%	28%	20%

TABLE VIII
ARRAY ROBUSTNESS (PYTHON)

Pixels Flipped	0.1%	1%	10%	25%
Accuracy	79%	74%	73%	65.6%

VI. DISCUSSION, CONCLUSION, AND FUTURE WORK

This work is an effective adapted implementation of a Hebbian and STDP-based learning algorithm illustrated in circuit level simulations in HSPICE and a behavioral model in Python. Behavioral simulations offer insight into behavior of the full-scale architecture. Accuracies upward of 80.4% were recorded with a single layer. A low total power consumption of 13.7 mW was measured. Simulations demonstrated robustness to over 10% device failure (Table IV) and resilience to 50% or more variability in RRAM devices (Table VII) without significant reduction in performance. Utilization of the inherent decay of STP devices promoted higher accuracy relative to conventional long-term plasticity devices (Table V). These findings offer potential for a wide range of applications for this network.

We reported a novel neuromorphic architecture that utilized a learning algorithm exploiting benefits from both STM and LTM. The empirically characterized STM was shown to demonstrate excellent potential for training by capitalizing on analog states and self-decay in RRAM devices while the trained weights were stored in LTM. A novel thresholding circuit was presented that can successfully translate analog weights into stable binary weights. The network showed promising training and inferencing capabilities with low power consumption.

The presented neuroprocessor architecture represents a framework that can be extended upon for more complex STM and LTM interactions to model more features of WM. One extension could add an ability to compare older stored LTM weights with newly associated STM weights to identify significant changes in STM's representation of classes. This would allow the system to update or modify its representation of classes if the environment changed over time. Future work using a more scalable device-level simulation platform to explore the limits of the proposed neuroprocessor architecture will be explored.

REFERENCES

[1] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 1–12, doi: 10.1145/3079856.3080246.

[2] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: Design considerations," *Frontiers Neurosci.*, vol. 10, p. 333, Jul. 2016, doi: 10.3389/fnins.2016.00333.

[3] G. Deco, E. T. Rolls, and R. Romo, "Synaptic dynamics and decision making," *Proc. Nat. Acad. Sci. USA*, vol. 107, no. 16, pp. 7545–7549, Apr. 2010, doi: 10.1073/pnas.1002333107.

[4] T. J. Bailey and R. Jha, "Understanding synaptic mechanisms in $SrTiO_3$ RRAM devices," *IEEE Trans. Electron Devices*, vol. 65, no. 8, pp. 3514–3520, Aug. 2018, doi: 10.1109/TED.2018.2847413.

[5] T. Werner *et al.*, "Experimental demonstration of short and long term synaptic plasticity using OxRAM multi k-bit arrays for reliable detection in highly noisy input data," in *IEDM Tech. Dig.*, Dec. 2016, p. 16, doi: 10.1109/IEDM.2016.7838433.

[6] T. Klingberg, "The Mental Workbench," in *The Overflowing Brain: Information Overload and the Limits of Working Memory*. New York, NY, USA: Oxford Univ. Press, 2009 pp. 33–44.

[7] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing Deep Convolutional Networks using Vector Quantization," 2014.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *CoRR*, vol. abs/1510.00149, 2016.

[9] H.-S.-P. Wong *et al.*, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012, doi: 10.1109/JPROC.2012.2190369.

[10] R. Waser, T. Baiatu, and K.-H. Härdtl, "Degradation of dielectric ceramics," *Mater. Sci. Eng. A*, vol. 109, pp. 171–182, Oct. 1989, doi: 10.106/0921-5093(89)90583-2.

[11] H. Markram, "A history of spike-timing-dependent plasticity," *Frontiers Synaptic Neurosci.*, vol. 3, pp. 1–24, Jun. 2011, doi: 10.3389/fnsyn.2011.00004.

[12] R. Kempter, W. Gerstner, and J. L. van Hemmen, "Hebbian learning and spiking neurons," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 59, no. 4, pp. 4498–4514, Apr. 1999, doi: 10.1103/PhysRevE.59.4498.

[13] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neurosci.*, vol. 3, no. 9, pp. 919–926, Sep. 2000, doi: 10.1038/78829.

[14] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, no. 6595, pp. 76–78, Sep. 1996, doi: 10.1038/383076a0.

[15] P. D. Roberts and C. C. Bell, "," *J. Comput. Neurosci.*, vol. 9, no. 1, pp. 67–83, 2000, doi: 10.1023/A:1008938428112.

[16] M. C. W. van Rossum, G. Q. Bi, and G. G. Turrigiano, "Stable Hebbian learning from spike timing-dependent plasticity," *J. Neurosci.*, vol. 20, no. 23, pp. 8812–8821, Dec. 2000, doi: 10.1523/JNEUROSCI.20-23-08812.2000.

[17] J. E. Rubin, R. C. Gerkin, G.-Q. Bi, and C. C. Chow, "Calcium time course as a signal for spike-timing-dependent plasticity," *J. Neurophysiology*, vol. 93, no. 5, pp. 2600–2613, May 2005, doi: 10.1152/jn.00803.2004.

[18] R. Gütig, R. Aharonov, S. Rotter, and H. Sompolinsky, "Learning input correlations through nonlinear temporally asymmetric Hebbian plasticity," *J. Neurosci.*, vol. 23, no. 9, pp. 3697–3714, May 2003.

[19] Y. B. Nian, J. Strozier, N. J. Wu, X. Chen, and A. Ignatiev, "Evidence for an oxygen diffusion model for the electric pulse induced resistance change effect in transition-metal oxides," *Phys. Rev. Lett.*, vol. 98, no. 14, Apr. 2007, Art. no. 146403, doi: 10.1103/PhysRevLett.98.146403.

[20] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA, USA: Addison-Wesley, 1989.

[21] *ASU PTM 130nm SPICE Model*. Accessed: Apr. 29, 2018. [Online]. Available: http://ptm.asu.edu

[22] S. Sinha, G. Yeric, V. Chandra, B. Cline, and Y. Cao, "Exploring sub-20nm FinFET design with predictive technology models," in *Proc. 49th Annu. Des. Autom. Conf.*, 2012, pp. 283–288, doi: 10.1145/2228360.2228414.

[23] A. Balijepalli, S. Sinha, and Y. Cao, "Compact modeling of carbon nanotube transistor for early stage process-design exploration," in *Proc. Int. Symp. Low power Electron. Des.*, 2007, pp. 2–7, doi: 10.1145/1283780.1283783.

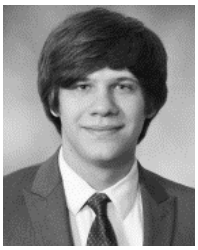
[24] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Trans. Electron Devices*, vol. 53, no. 11, pp. 2816–2823, Nov. 2006, doi: 10.1109/TED.2006.884077.

[25] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2019, pp. 201–204.

[26] M. J. Akhtar, Z.-U.-N. Akhtar, R. A. Jackson, and C. R. A. Catlow, "Computer simulation studies of strontium titanate," *J. Am. Ceram. Soc.*, vol. 78, no. 2, pp. 421–428, 1995, doi: 10.1111/j/1151-2916.1995.tb08818.x.

[27] S.-I. Amari, "Neural theory of association and concept-formation," *Biol. Cybern.*, vol. 26, no. 3, pp. 175–185, 1977, doi: 10.1007/BF00365229.

[28] D. Querlioz, P. Dollfus, O. Bichler, and C. Gamrat, "Learning with memristive devices: How should we model their behavior?" in *Proc. IEEE/ACM Int. Symp. Nanosc. Architectures*, Jun. 2011, pp. 150–156, doi: 10.1109/NANOARCH.2011.5941497.



Tony James Bailey (Member, IEEE) received the B.S. degree in electrical engineering from the University of Cincinnati, Cincinnati, OH, USA, in 2018, where he is currently working toward the M.S. degree in electrical engineering.

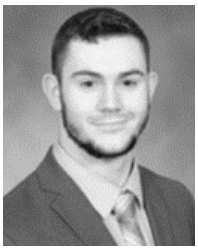
He is also a Graduate Research Assistant at the University of Cincinnati, where he focuses his research on neuromorphic computing and emerging memory devices.

Andrew J. Ford (Member, IEEE) received the B.S. degree in electrical engineering from the University of Cincinnati, Cincinnati, OH, USA, in 2019, where he is currently working toward the M.S. degree in electrical engineering.

He works as a Graduate Research Assistant at the University of Cincinnati, where he researches neuromorphic computing, emerging memory devices, and applications of electroencephalography data.

Siddharth Barve (Member, IEEE) is working toward the B.S. degree in electrical engineering and the M.S. degree in electrical engineering at the University of Cincinnati, Cincinnati, OH, USA.

He works as a Undergraduate Research Assistant at the University of Cincinnati, where he focuses his research on neuromorphic computing and emerging memory devices.



Jacob Wells (Member, IEEE) received the B.S. degree in electrical engineering from the University of Cincinnati, Cincinnati, OH, USA, in 2018. He worked on this project during senior years of his B.S. degree at the University of Cincinnati.

Rashmi Jha (Member, IEEE) received the B.Tech. degree in electrical engineering from IIT Kharagpur, Kharagpur, India, in 2000, and the M.S. and Ph.D. degrees in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 2003 and 2006, respectively.