Memristive Mixed-Signal Neuromorphic Systems: Energy-Efficient Learning at the Circuit-Level

Gangotree Chakma, Md Musabbir Adnan, Austin R. Wyer, Ryan Weiss, Catherinhe D. Schuman and Garrett S. Rose

IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), Vol. 8, No. 1, March 2018.

Citation Information (BibTex):

```
@ARTICLE{ChakmaJETCAS:2018,
  author="G. Chakma and M. M. Adnan and A. R. Wyer and
      R. Weiss and C. D. Schuman and G. S. Rose",
  journal="{IEEE} Journal on Emerging and Selected topics
      in Circuits and Systems{JETCAS}"
  title="Memristive Mixed-Signal Neuromorphic Systems:
      Energy-Efficient Learning at the Circuit-Level",
  year="2018",
  volume="8",
  number="1",
  pages="125-136",
  keywords="Biological neural networks;Memristors;
      Neuromorphics;Neurons;Resistance;Switches;
      Memristor;emerging technology;neuromorphic
      computing;simulator",
  doi="10.1109/JETCAS.2017.2777181",
  ISSN="2156-3357",
  month="March"
}
```

Memristive Mixed-Signal Neuromorphic Systems: Energy-Efficient Learning at the Circuit-Level

Gangotree Chakma, *Student Member, IEEE,* Md Musabbir Adnan, *Student Member, IEEE,*
Austin R. Wyer, *Student Member, IEEE,* Ryan Weiss, *Student Member, IEEE,*
Catherine D. Schuman, *Member, IEEE,* and Garrett S. Rose, *Member, IEEE*

***Abstract*—Neuromorphic computing is a non-von Neumann computer architecture for the post Moore's law era of computing. Since a main focus of the post Moore's law era is energy-efficient computing with fewer resources and less area, neuromorphic computing contributes effectively in this research. In this paper we present a memristive neuromorphic system for improved power and area efficiency. Our particular mixed-signal approach implements neural networks with spiking events in a synchronous way. Moreover, the use of nano-scale memristive devices saves both area and power in the system. We also provide device-level considerations that make the system more energy-efficient. The proposed system additionally includes synchronous digital long term plasticity (DLTP), an online learning methodology that helps the system train the neural networks during the operation phase and improves the efficiency in learning considering the power consumption and area overhead.**

***Index Terms*—Memristor, Simulator, Emerging Technology, Neuromorphic Computing.**

I. Introduction

The human brain is comprised of a complex interconnection of neurons that process and transmit data via electro-chemical signals. These neurons are interconnected at junctures known as synapses. The "strength" of the signal transmitted from one neuron to another is proportional to the strength of their interconnection, known as the *synaptic weight*. Each neuron performs the weighted summation of the signals it receives from its preceding neurons. When this summation exceeds a threshold, it transmits a *fire* signal to the succeeding neurons. When this condition occurs, the neuron is said to have fired.

G. Chakma, Md Musabbir Adnan, A.R. Wyer, R. Weiss, and G.S. Rose are with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, 37996 USA E-mail: {gchakma, awyer, rweiss, garose}@utk.edu.

C.D. Schuman is with Oak Ridge National Laboratory, Oak Ridge, TN.

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8750-16-1-0065 and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Reserach, under contract number DE-AC05-00OR22725. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government.

The striking feature of biological neural networks is their ability to adapt their architecture to produce the expected outputs when performing tasks such as image and speech recognition. This adaptation is performed by a process known as *learning* wherein the synaptic weights are updated, thereby affecting the information flow in the neural network.

Artificial Neural Networks (ANNs) are a network paradigm that mimic biological neural networks. They consist of a mathematical model that defines how neurons are interconnected, the strengths of their connections (synaptic weights), how weights are updated, and the behavior of neuron firing events. While ANNs have been shown to be effective in representative applications such as pattern, image and text recognition, they are still reliant on conventional von Neumann machines for implementation, which yield the expected results, but the throughput is incomparable to their biological counterparts. This is because the machines that run these ANN algorithms process information in a sequential manner unlike biological neural networks that are truly parallel.

This need for parallel processing motivates research on dedicated hardware for ANNs. A hardware circuit designed for neural networks is known as neuromorphic circuit. Numerous approaches to neuromorphic computing have been proposed, many of which use digital [1] or analog CMOS approaches [2]–[4]. While the digital implementations have precision, robustness, noise resilience and scalability, they are area intensive [2]. Their analog counterparts are quite efficient in terms of silicon area and processing speed. However, they rely on representing synaptic weights as volatile voltages on capacitors [4] or in resistors [5], which do not lend themselves to energy and area efficient learning. A review of several existing implementations of neural networks can be found in [6].

Lately, the semiconductor industry has begun to experience a significant slowdown in performance improvements gained from technology scaling. While this is due in part to the impending end of Moore's Law scaling, power consumption and architectural limitations have also become critical limiting factors for the level of performance achievable. The research proposed here aims to overcome this roadblock by (1) leveraging an emerging nano-scale device (i.e. the memristor [7]) and (2) Spiking Neural Network (SNN) architecture to realize neuromorphic computing [8].

We specifically propose a mixed-signal system where communication and control is digital while the core multiply-and-accumulate functionality is analog. The remainder of the paper

is as follows. Section II details the background for memristive devices (parameters and modeling) and the advantage of using the devices. The circuit specifications and design for the hardware implementation of the synapses and neurons with the algorithm of using in a particular neuromorphic design are described in section III. Section IV represents the learning topology in our specific neuromorphic architecture that utilizes genetic algorithm for offline training and digital long term plasticity for online training. Results illustrated in section V show the operation and benefits of the proposed system, with energy consumption for classification of three well-known datasets and the effect of online learning on the overall accuracy of learning process. Section VI concludes the paper.

II. Background

Memristors are two terminal nanoscale non-volatile devices first theorized by Leon O. Chua [7] in 1971. Memristors are resistors whose resistance can be modulated by the magnitude of voltage across the device and the time for which the voltage is applied. A memristor can attain multiple resistance levels between the two bounds known as their low resistance state (LRS) and high resistance state (HRS). The LRS and HRS of any memristor is dependent on the switching material, process conditions, noise and environmental conditions. The resistance of a memristor can be switched between HRS and LRS by applying an appropriate bias voltage. While the applied bias voltage is above a particular threshold level for at least some minimum amount of time, the resistance of the memristor switches from one state to another. Based on the switching from (HRS to LRS) and (LRS to HRS), threshold voltages and switching times could be different and can be defined as positive threshold voltage (V_{tp}), positive switching time (t_{swp}), negative threshold voltage (V_{tp}), and negative switching time (t_{swn}), respectively. Materials used to build memristors include TaO_x [9], TiO_2 [10], HfO_x [11], chalcogenides [12], [13], silicon [14], [15], organic materials [16], ferroelectric materials [17], [18], carbon nanotubes [19], etc. All of these memristors are differentiated by their LRS values, LRS to HRS ratios, threshold voltages, and switching times. For this design suitable ranges of LRS and HRS (Table I) have been considered based on the values found in the literature.

TABLE I
Switching Parameters for Metal-Oxide Memristors

Parameter \ Devices (mean)	TaO_x [9]	HfO_x [20]	TiO_x [10]	Parameter variance
HRS	$10k\Omega$	$300k\Omega$	$2M\Omega$	$\pm 20\%$
LRS	$2k\Omega$	$30k\Omega$	$500k\Omega$	$\pm 10\%$
V_{tp}	0.5V	0.7V	0.5V	$\pm 10\%$
V_{tn}	-0.5V	-1.0V	-0.5V	$\pm 10\%$
t_{swp}	105ps	10ns	10ns	$\pm 5\%$
t_{swn}	120ps	$1\mu s$	10ns	$\pm 5\%$

Here, the memristor model used for simulation is derived from a model previously developed in [21]. Our model specifically emphasizes the bipolar behavior considered in previous related works [20]. While performing a SET operation from HRS to LRS, the resistance change in the memristor is given by:

$$R_{new} = R_{initial} - \frac{\Delta r \times |V(t)| \times t_{pw}}{t_{swp} \times V_{tp}}. \tag{1}$$

The resistance change during the RESET operation is given by:

$$R_{new} = R_{initial} + \frac{\Delta r \times |V(t)| \times t_{pw}}{t_{swn} \times V_{tn}}, \tag{2}$$

where R is the resistance of the memristor, Δr is the absolute difference between the HRS and LRS values, $V(t)$ is the applied voltage across the memristor and t_{pw} is the time duration for an applied voltage pulse. Assuming the memristors have symmetric switching time and threshold voltage, the change in memristance (ΔR) in either direction is given by:

$$\begin{aligned} \Delta R &= R_{new} - R_{initial} \\ &= \frac{\Delta r \times |V(t)| \times t_{pw}}{t_{sw} \times V_{th}}, \end{aligned} \tag{3}$$

where $t_{sw} = t_{swp} = t_{swn}$ and $V_{th} = V_{tp} = V_{tn}$. An example current-voltage relationship of the memristor model used in this work is shown in Fig. 1,

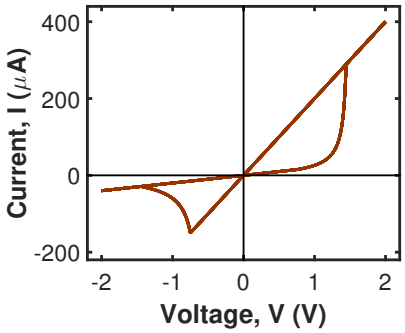

Fig. 1. I-V characteristics of memristor model.

Owing to their programmability and non-volatility, artificial synapses can be implemented using memristors to represent weight values and transmit analog weighted results to post-synaptic neurons. The neuron uses the analog output of the synapse to produce a digital firing event (or spike) that is synchronized with the system. Further, the system considered here leverages an unsupervised Long Term Plasticity (LTP) model for on-chip learning. Based on the temporal relationship of the pre- and the post-neuron fires, the synaptic weight is modified by a pre-neuron with an LTP control block and feedback from the post-neuron.

III. Memristive Dynamic Adaptive Neural Core

A. Neural Core System

The neuromorphic system considered (Fig. 2) consists of mxn memristive neuromorphic cores. Each core has several memristive synapses and one mixed-signal neuron (analog in, digital out) to implement a spiking neural network. To connect one neural core to another, a few issues need to be considered;

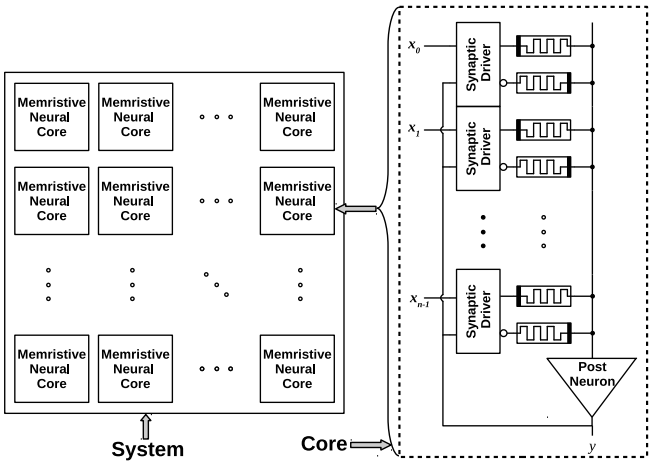

Fig. 2. A floorplan of memristive neuromorphic core system.

such as if we separate out the synapses and neurons from a core, their connecting wires would have long lengths and longer wire would lead to a larger capacitance and thus lower performance; different synapses driving different capacitances would result in different accumulation of charges though the synaptic weights would be same. So, considering these performance issues, we have developed the configuration shown in Fig 2(right) which represents the synapses and neuron included in each memristive neural core. This arrangement helps maintain similar capacitance at the synaptic outputs and corresponding neurons. The similar distance between synapse and inputs also results in negligible difference in charge accumulation.

B. Twin Memristor as a Synapse

A twin memristor configuration (shown in Fig. 3) is considered here for storing the synaptic weight value. Each synaptic weight is represented using a pair of memristors. In this design, voltage inputs across memristive weights yield a weighted sum in the form of a current which is similar to several other memristor-based neural network designs [22]–[25]. Here, the current flowing through the synaptic node is proportional to its weight, which is in turn dependent on the resistances of the two memristors.

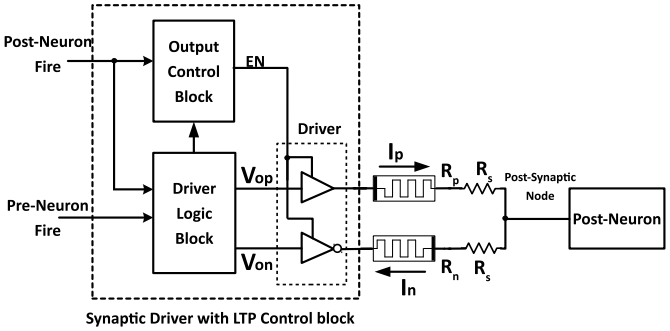

Fig. 3. Twin memristor synapse along with its control block providing the interlink between the pre- and post-neuron.

The twin memristor synapse is used here to implement both positive and negative synaptic weights. In the literature ideas have been explored [24] that represent negative components of the weights using a twin memristive crossbar. There has also been work with memristive crossbars [26]–[30] to mimic human brain. The design considered here does not specifically consider the crossbar array since the focus is to build a simple neuromorphic system core with synapses that can efficiently provide both positive and negative weight values. In the memristor pair used for each synapse, one memristor is used to drive positive current relative to positive weight component, while the other pulls current, relative to the negative component. Here we have assigned the post-synaptic node as a mid-rail (for this case a virtual ground) because the post-synaptic node is controlled by the op-amp/integrator input node as discussed in section III-D in detail. The effective current flowing into the post-neuron thus depends on the relative values of the resistances of the twin memristor pair. The weight of this synapse is proportional to the effective conductivity of the pair of memristors shown in equation 4 where $G_{eff,i}$ is the effective conductance of the i^{th} synapse and W_i is its synaptic weight.

$$G_{eff,i} \propto W_i \tag{4}$$

From this we see that the effective conductivity of any twin memristor has a linear relationship with the weight of the corresponding synapse. To model the synaptic weights through the memristors we utilized the following.

$$G_{eff,i} = W_i.G_{eff,1} \tag{5}$$

$$\begin{aligned} W_i.G_{eff,1} &= \frac{1}{R_{p,i}} - \frac{1}{R_{n,i}} \\ &= \frac{1}{R_{p,i}} - \frac{1}{LRS + HRS - R_{p,i}}; \\ where R_n &= LRS + HRS - R_p \end{aligned} \tag{6}$$

The effective conductance of the synapse is limited by HRS and LRS values of the memristors. Maximum conductance (G_{max}) is attained when R_p reaches LRS and R_n reaches HRS. On the other hand, we assume that the minimum conductance that represents a synaptic weight of "0" happens when both R_p and R_n are equal and at the mid level resistance between LRS and HRS which is $(HRS + LRS)/2$. For the initial condition, we assume that the synaptic change is approximately symmetric in both directions from the median of LRS and HRS such that ΔR_n=ΔR_p. Thus, the values of R_n and R_p would be initialized at an equal distance from the median of LRS and HRS which leads us to the assumption of R_n+R_p=LRS+HRS. After initialization, the subsequent changes via ΔR_n and ΔR_p update the values of R_n and R_p through online learning. Thus we can represent the resistance associated with each synaptic weight with

$$\begin{aligned} R_{p,i} = \frac{HRS + LRS}{2} + \frac{1}{W_i.G_{eff,1}} \\ + \frac{1}{2}.\sqrt{\Big[(HRS + LRS)^2 + \frac{1}{(W_i.G_{eff,1})^2}\Big]}. \end{aligned} \tag{7}$$

If the resistance values of both memristors in the pair are equal, their currents will cancel each other for any given input spike such that the effective weight is zero. Along similar lines, if R_p is lesser (greater) than R_n, the weight is positive (negative). The synapse uses a digital logic block to provide driving voltages to each memristor in the pair. The synapse here operates in two phases, namely *accumulation* and *learning*. The *accumulation* phase occurs when the pre-neuron fires. This fire event triggers the synaptic control block to drive a positive current through R_p and a negative current through R_n, while the post-synaptic node is held at mid-rail by the virtual ground of the post-neuron. The effective current flowing into the post-neuron either accumulates charge or discharges. The *learning* phase occurs when the post-neuron fires. If the pre-neuron fires just before the post-neuron, the synapse weight is increased (potentiation). Similarly, if the pre-neuron fires just after the post-neuron fires, the corresponding synaptic weight is decreased (depression). This is in accordance with the STDP rule, which is believed to be the cause for learning in biological neural networks.

C. Digital Long Term Plasticity (DLTP)

Most neural networks considered in the literature depend on learning or training algorithms such as supervised gradient descent learning or back-propagation. These learning topologies help the network to learn offline. For a network to learn online, Long Term Plasticity plays an important role in training the circuit with continuous updates of synaptic weights based on the timing of pre- and post-neuron fires. LTP is inspired by biological synapses where the synaptic weights are updated with the help of synaptic plasticity. Researchers have developed several circuits to mimic synaptic plasticity behavior [31]. Most of these prior works show that the time difference and the applied voltage tail create a difference in the magnitude of the voltage applied across the synapses which results in updating synaptic weights.

In our neuromorphic architecture we leverage the mixed-signal nature of the system for online learning. Instead of carefully crafting analog tails to provide variation in the voltage across the synapses, we utilize digital pre- and post-neuron firing signals and apply pulse modulation to implement a digital LTP (DLTP) technique. The process is a one cycle tracking process, meaning if there is any post-neuron fire present, the circuit considers the pre-neuron fires just before and after the cycle of the post-neuron fire. If the pre-neuron fire arrives before the post-neuron fire, the synapse weight is potentiated. Likewise, if it arrives a clock cycle later, the synapses weight gets depressed. Changing the weight requires the application of a voltage greater than the memristor threshold. Basically the online learning process implemented here is one clock cycle tracking version of Spike time Dependent Plasticity (STDP) [32]–[35]. A more thorough STDP learning implementation would need to track several clock cycles before and after the post-neuron fire leading to more circuitry and hence increased power and area. Our DLTP approach acts similarly but ensures lower area and power.

In Fig 3, the effective conductance of the twin memristor can be defined by the following equation:

$$G_{eff} = \frac{1}{R_p} - \frac{1}{R_n}. \tag{8}$$

When a synapse goes through DLTP, there is change in the resistances of the twin memristor ΔR which we assume to be the same for both potentiation and depression of synaptic weights. Considering potentiation, the new effective conductance can be defined by:

$$\begin{aligned}
G_{eff,pot} &= \frac{1}{R_p - \Delta R} - \frac{1}{R_n + \Delta R} \\
&= \frac{1}{R_p\Big(1 - \frac{\Delta R}{R_p}\Big)} - \frac{1}{R_n\Big(1 + \frac{\Delta R}{R_n}\Big)} \\
&= \frac{1}{R_p}\Big(1 - \frac{\Delta R}{R_p}\Big)^{-1} + \frac{1}{R_n}\Big(1 + \frac{\Delta R}{R_n}\Big)^{-1} \\
&= \frac{1}{R_p}\Big[1 + \frac{\Delta R}{R_p} + \Big(\frac{\Delta R}{R_p}\Big)^2 + ...\Big] - \frac{1}{R_n}\Big[1 \\
&- \frac{\Delta R}{R_n} + \Big(\frac{\Delta R}{R_n}\Big)^2 - ....\Big] \\
&= \frac{1}{R_p} - \frac{1}{R_n} + \Delta R\Big(\frac{1}{R_p^2} + \frac{1}{R_n^2}\Big) + \Delta R^2\Big(\frac{1}{R_p^3} \\
&- \frac{1}{R_n^3}\Big) + ... \\
&= G_{eff} + \Delta R\,(G_p^2 + G_n^2) + \Delta R^2\,(G_p^3 - G_n^3) + ...
\end{aligned} \tag{9}$$

Thus the change in the effective conductance can be described by:

$$\begin{aligned}
\Delta G_{pot} &= G_{eff,pot} - G_{eff} \\
&= \Delta R\,(G_p^2 + G_n^2) + \Delta R^2\,(G_p^3 - G_n^3) + ...,
\end{aligned} \tag{10}$$

and for positive weights ($R_p < R_n$) the change would be higher than that of the negative weights ($R_p > R_n$).

If we consider the reduction in weight, the new depressed effective conductance will be:

$$\begin{aligned}
G_{eff,dep} &= \frac{1}{R_p + \Delta R} - \frac{1}{R_n - \Delta R} \\
&= \frac{1}{R_p\Big(1 + \frac{\Delta R}{R_p}\Big)} - \frac{1}{R_n\Big(1 - \frac{\Delta R}{R_n}\Big)} \\
&= \frac{1}{R_p}\Big(1 + \frac{\Delta R}{R_p}\Big)^{-1} + \frac{1}{R_n}\Big(1 - \frac{\Delta R}{R_n}\Big)^{-1} \\
&= G_{eff} - \Delta R\,(G_p^2 + G_n^2) + \Delta R^2\,(G_p^3 - G_n^3) - ....
\end{aligned} \tag{11}$$

Thus, the synaptic weight change which is proportional to the effective conductance change would be:

$$\Delta G = -[\Delta R\,(G_p^2 + G_n^2) - \Delta R^2\,(G_p^3 - G_n^3) + ...], \tag{12}$$

and similarly we can say that the change would not be perfectly equal for both positive and negative weights. It is to be noted that the memristor device parameters and choice of clock frequency ensures ΔR to be smaller than both R_p and R_n. Hence the binomial series expansion is valid for both cases.

To implement DLTP at the circuit level, the output control block generates a signal that enables potentiation/depression by sensing the presence of firing spikes from the post-neuron caused by any synaptic input signal from the pre-neuron following

$$EN = \overline{\overline{F_{post} * F_{pre\_t}} * F_{pre\_b}}, \quad (13)$$

where F_{post} is the signal from post-neuron, F_{pre_t} is a delayed signal from pre-neuron and F_{pre_b} is the inversion of the pre-neuron signal. The EN signal is also asserted during the $accumulation$ phase so that V_{op} and V_{on} can drive positive and negative currents through R_p and R_n respectively.

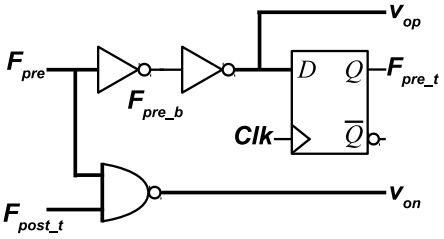

Fig. 4. Driver logic block.

The synapse driver logic block generates both the positive (V_{op}) and negative (V_{on}) driving voltages to the memristors. During accumulation, R_p and R_n are driven to positive rail and negative rail respectively. This is achieved by making $V_{op} = V_{on} = V_{DD}$. It should be noted that the signal V_{on} drives an inverter to supply negative voltage (V_{SS}) on R_n (Fig. 3). Additionally, the post-synaptic node is held at virtual ground(mid-rail) so that the voltages across the memristors stay below the switching threshold of the memristor. This operational block is also responsible for supplying correct driving voltage to the twin memristor during the learning phase. If the control block senses a potentiation, the driver logic block will operate in such a way that the voltage across the memristors R_p and R_n crosses the positive and negative threshold, respectively, and hence the synaptic weight will increase following equation 10. So, for potentiation, $V_{op} = V_{SS}$ and $V_{on} = V_{DD}$ while the post-synaptic node is held at V_{DD} by the feedback of neuron which will be described on section III-D in details. This results in rail-to-rail voltage drop across R_p and R_n. Since they are connected in opposite polarity, the value of R_p decreases while R_n increases making the G_{eff} rise according to equation 8. Similarly, the depression logic is also dependent on the proper voltage across the memristors R_p and R_n crossing the threshold in the opposite direction. However, the post-synaptic node is also responsible for controlling DLTP.

To analyze our implementation of DLTP, we have considered a small network of two synapses in Fig. 5 with weights of "1", two pre-neurons and a single post-neuron with a threshold of "2". Here the pre-neurons act as the sender of the synaptic signals to the corresponding synapses and the post-neuron receives that weighted signal and generates post-synaptic fires which will be input to the next layer of pre-neurons. The pre-neuron inputs are digital pulse-trains (shown in Fig. 6) with F_{pre1} and F_{pre2} and the post-neuron output is denoted by

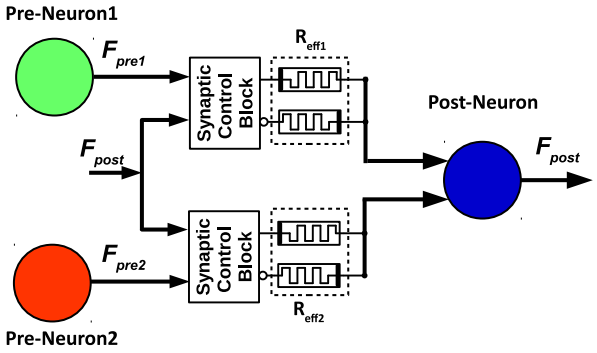

Fig. 5. Small network for DLTP with two synapses connected to a single neuron.

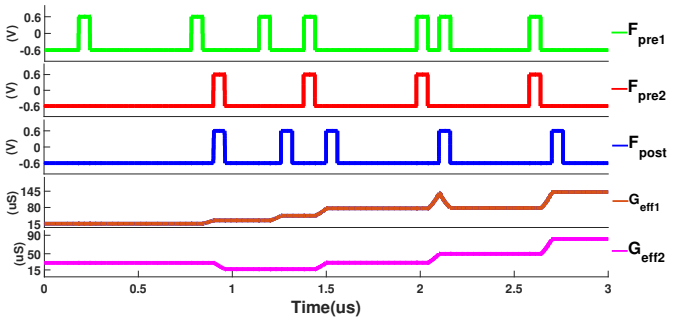

Fig. 6. Simulation result for DLTP with two synapses connected to a single neuron.

F_{post}. Since the DLTP circuit tracks the pre- and post- neuron spikes for a clock cycle before and after firing events, we can assume from the figure that both of the synapses will go through potentiation and depression in different clock cycles. In Fig. 6, G_{eff1}band G_{eff2} are the effective conductance of the two synapses and primarily these are initialized at an initial state based on the initial resistance of the memristive synapses. If we analyze the pre- and post- neuron spikes, we would see that the first post-neuron fire occurs after accumulating the charge of the first two F_{pre1} fires. So, the synapse R_{n1} is being potentiated and hence G_{eff1} is being increased. On the other hand, the first fire of F_{pre2} is arriving simultaneously with post-neuron fire and it is not responsible for post-neuron fire. So, the synapse R_{n2} is being depressed and hence the effective conductance, G_{eff2} is decreased. However the synaptic weight change will not be same for each stage because with online learning updates the weights and the next weight change will be based on the updated weights.

D. Analog CMOS Neurons

Leveraging mixed-signal design, our approach towards integrating the summing current is analog in nature while spike generation is digital. We consider two different implementations for analog neurons. One is a CMOS analog Integrate and Fire (IAF) neuron and the other is an Axon-Hillock neuron. For the IAF neuron, an integrate-and-fire circuit (Fig. 7-8) similar to that described by Wu *et al.* [36] is implemented where the neurons are designed to produce spikes based on

the incoming weighted input signals. The design allows the neurons to operate in two different phases, an integration phase and a firing phase. During the integration phase of the neuron,

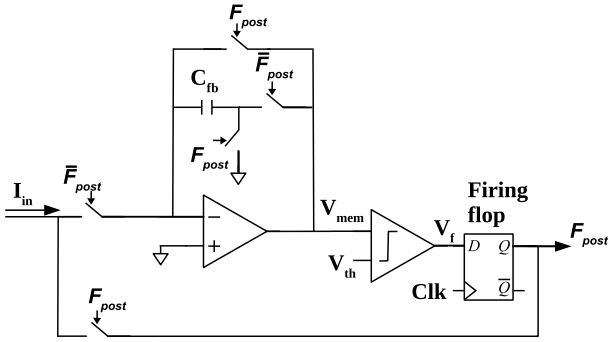

Fig. 7. Analog Integrate and Fire (IAF) Neuron.

the op amp acts as an integrator such that the capacitor, C_{fb} in Fig. 7, accumulates charge (from the current coming from the synapse) resulting in a change in membrane potential V_{mem}. A comparator circuit compares the membrane potential V_{mem} with the threshold voltage V_{th} and generates firing spikes when necessary. The firing flop then takes the comparator output and delivers a firing pulse (spike) synchronous with the digital inputs. Fig. 8 provides transistor-level detail for the integration and comparison techniques. Here we have used V_{ref} voltage as the mid rail voltage ("0V" in this case) to ensure the virtual ground on the "-" input of the op-amp during accumulation.

When a neuron is in the firing phase, the op amp acts as a buffer for resetting the charge potential. The feedback mechanism is also active during the firing phase to implement dynamic adaptation (i.e. DLTP) of the synaptic weights. Specifically, any synapse input that leads to a firing event is potentiated (weight increased) while that of any synapse input simultaneous with the output fire is depressed. Thus, the input node of the IAF neuron is responsible for partially driving the DLTP process described in section III-C. The feedback mechanism supplies a voltage potential on one side of the memristor pair so that the synaptic weights are updated based on the present voltage drop across the memristors. Moreover, this also helps in establishing a one cycle refractory period, meaning the neuron will be idle for one clock cycle after any firing event and will not accumulate charge even if there is any synaptic input present during the refractory period. The basic function of the IAF neuron is similar to the neuron in [36]. This implementation also ensures that the neuron can reset itself after generating a firing spike and prepare for the upcoming input fires being synchronous with the operating clock.

The axon hillock neuron first proposed by Carver Mead [37] integrates an input current on a capacitor and outputs a voltage spike upon crossing a threshold. The circuit shown in Fig. 9 implements a synchronized axon hillock neuron [38]. Like the integrate and fire model, the axon hillock has two phases of operation, accumulation and firing. During accumulation, the voltage V_{mem} is asymptotically driven to the voltage created by the resistive division of the synapse. When

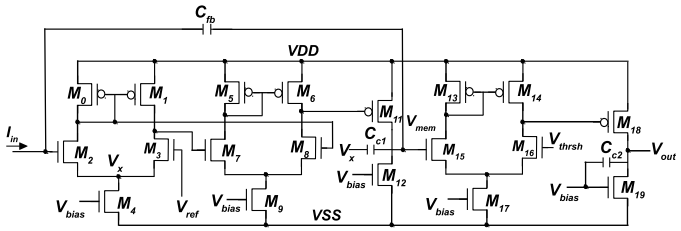

Fig. 8. Analog integration of charges and comparison with neuron threshold.

the accumulated voltage is relatively far from the voltage created by the synapse via the resistance devision of the memristors, the input charges or discharges approximately linearly towards the voltage created by the synapse. As the accumulated voltage approaches the voltage at the synapse, the change in accumulated voltage approaches zero. Because of this relationship between synapses and neuron, the threshold voltage V_{th} must be below the maximum voltage created by the voltage division at the synapse. If the accumulated voltage V_{mem} has reached a voltage higher than V_{th}, then in the clock cycle after a synapse input, the neuron will fire. The timing of this fire is facilitated by the delayed input fire signal F_{pre_t}. During the firing phase, the input current is blocked, and the voltage V_{mem} is reset to voltage V_{RST}. The common node is driven by M_{13} and M_{14} during firing for DLTP.

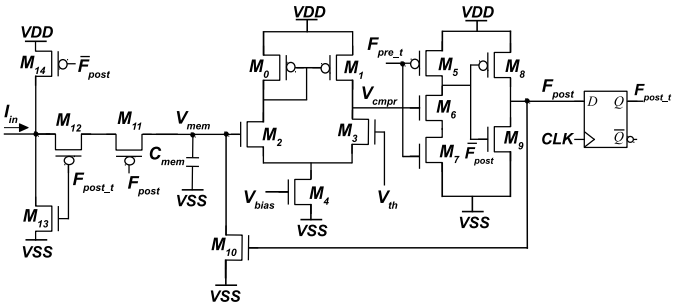

Fig. 9. Implementation of Axon-Hillock neuron with a comparator for variable threshold.

E. Mapping of Neural Networks on Proposed Mixed-Signal Architecture

Our goal behind designing the proposed mixed signal memristive architecture described in section III-B-III-D is to design a system specifically tailored for realizing artificial neural networks. Our motivation has been to help the research community in building a strong bridge while we are translating the neural networks to the circuits capable of simulating neural networks. We have modeled the behavior of the memristive synapses using C++ models to map the behavior of the memristive synapses correctly while generating neural networks. We have used several parameters related to the memristive material so that the high-level model can be adaptive to the changes made in the circuit level component. Moreover the

analog neurons have also been modeled at a high level using C++ to generate neural networks with components equivalent to the circuit level neurons. While the neurons operate in different phases in circuit-level implementation, the high-level model of the neuron also goes through the same operation mapping the charge accumulation and firings of the neurons. In addition, we have utilized the DLTP mechanism to train the networks while keeping the feature of online learning on and off. The following section IV would explain more about the procedure of network generation and initialization for online learning.

IV. Neuromorphic Learning Architecture

A. Evolutionary Optimization (EO) for Network Initialization

One of the challenges in using neural networks is finding a suitable network topology for a given problem. While some network topologies might excel at classification, they might be ill suited for control tasks. Thus, to produce suitable network topologies for a variety of applications, we use a genetic algorithm (Evolutionary Optimization or EO) for network initialization adapted from Schuman et al. [39]. This approach has proven successful in generating spiking networks for other neuromorphic systems for a range of applications including basic logic, control problems such as pole-balancing [40], and classification [39].

The genetic algorithm for generating an initial network for a specific task works as follows. For the task, the user specifies the number of input neurons and the number of output neurons required for that task, which in turn specify what information will be given to the network about the task (the inputs) and what the task needs to receive from the network (the outputs). The user also specifies a starting number of hidden neurons and synapses. A population of randomly initialized networks is then generated, where each network in the population has the same number and placement of input and output neurons, and the hidden neurons and synapses are randomly initialized, so that each network in the population is distinct. Connectivity in the network is also random and the system allows for both feed-back and feed-forward connections.

The user must also specify a fitness function for each task. The fitness function takes a network as input and returns a numerical value that corresponds to how well that network is performing the given task. This usually entails applying inputs to the network, simulating activity on the network, and obtaining outputs. We use the fitness scores for the population to perform a selection and reproduction process to produce the next generation of the population. We utilize tournament selection to preferentially select better performing networks to serve as parents. Given a set of parents, we probabilistically apply crossover and mutation operations to produce children networks. Crossover combines subnetworks from each parent to produce each child, and mutations make small scale structural changes (e.g., adding or deleting a neuron) or parameter changes to the network (e.g., updating a neuron's a threshold). The fitness evaluation, selection, and reproduction processes are repeated until a desired fitness level is reached. The algorithm is described in Figure 10. The highest performing network is then returned to the user to be deployed either in configured hardware or simulation. Once deployed, online learning is allowed to run so that the weight values of the synapses may continue to be refined.

```
1: procedure EVOLVE
2:     population = INITIALIZEPOPULATION
3:     MaxFitness = -1
4:     epoch = 0
5:     while MaxFitness < and epoch < MaxEpoch do
6:         fitnesses = []
7:         for net in population do
8:             fitnesses[net] = FITNESS(net)
9:             if fitnesses[net] > MaxFitness then
10:                MaxFitness = fitnesses[net]
11:                BestNet = net
12:            end if
13:        end for
14:        children = []
15:        while size(children) < size(population) do
16:            p1, p2 = SELECTPARENTS(population,
                   fitnesses)
17:            if RANDOMFLOAT < CrossoverRate then
18:                c1,c2 = CROSSOVER(p1,p2)
19:            else
20:                c1 = DUPLICATE(p1)
21:                c2 = DUPLICATE(p2)
22:            end if
23:            if RANDOMFLOAT < MutationRate then
24:                MUTATE(c1)
25:            end if
26:            if RANDOMFLOAT < MutationRate then
27:                MUTATE(c2)
28:            end if
29:            children.append(c1)
30:            children.append(c2)
31:        end while
32:        population = children
33:        epoch += 1
34:    end while
35:    return MaxFitness, BestNet
36: end procedure
```

Fig. 10. Method for evolutionary optimization for network initialization

One of the key advantages of utilizing a genetic algorithm (beyond the ability to optimize network topology) is the ability to work within the constraints of the memristive neuromorphic core, such as constraints on the weight values of synapses or constraints on the connectivity for the network. Rather than training using an ideal representation and creating a mapping procedure to map from ideal to reality, the genetic algorithm optimizes within the constraints of the neuromorphic system. The genetic algorithm can also operate with either simulation or with a chip-in-the-loop for evaluation. In this work, we utilize simulation only, but in the future, the genetic algorithm could optimize not only to the network model, but to a particular chip as well.

One of the features of our model is that the delay between

neurons is programmable by the genetic algorithm. This is accomplished by embedding the neurons of a network onto a 2-dimensional grid, and defining the delay to be the distance between the neurons. These delays can be adjusted by moving a neuron through a mutation, or by connecting to another subnetwork during the crossover operation.

B. EO-based Initialization for Online Learning

We incorporate DLTP in our simulator not only during online-testing, but also during network generation. When measuring the network's fitness, we allow DLTP to alter the synaptic weights. This allows the genetic algorithm to assess how DLTP affects the fitness of the network, and discard networks for which the fitness decreases. For the iris classification task that we have considered, fitness evaluation takes 22500 cycles.

To model the synaptic weights in our simulator we define a mapping from the resistance of the memristors to an abstract weight value. Abstract weights allow the simulator to accurately capture the behavior of the system without having to model it at the circuit level. An arbitrary range is defined for the abstract weight values. For the twin memristor model we consider this range to be symmetric, such that the most positive weight has the same magnitude as the most negative weight. While we allow the synapse to take any real value within the range during operation, we restrict the initial synaptic weights to integer values. In this way, we restrict the networks that can be generated by our genetic algorithm, but we allow online learning to adjust the synaptic weights to superior values.

We generate the mapping by taking the largest effective conductance of the synapse to represent the largest abstract weight, and then determine the effective conductance corresponding to an abstract weight of 1. This effective conductance for the weight of 1 can then be used normalize any effective conductance. To simulate DLTP the model maintains the resistance of both memristors in the synapse. When a potentiation or depression event occurs, the resistances are updated according to device parameters supplied by the user, and the effective conductance is recalculated. This conductance is then appropriately normalized to map to a synaptic weight. The use of synaptic weights is advantageous as they are more human readable.

It is important that DLTP be enabled during training. The effects of DLTP are dependent on the topology of the network, as the network's connections will determine which synapses are either potentiated or depressed. Therefore, DLTP is enabled during evolutionary optimization. In this way, networks where DLTP has a positive effect on network performance will have their fitness increased, while networks where DLTP has a negative effect on performance will have their fitness decreased. Without enabling DLTP during training it is possible that the solution networks generated have their performance adversely affected by DLTP. Because the networks are tested over many cycles during each epoch of training, the long term effects of DLTP on the network's performance are well captured by the fitness.

V. Experimental Results and Analysis

Since neuromorphic computing is an alternative for "computing beyond Moore's law," energy consumption is a particularly important metric to establish its benefit. The proposed mixed-signal memristive neuromorphic system consumes energy mainly in its core, through the synapses and neuron responsible for computation. Some prior works have mentioned the amount of energy consumption by their components. For instance, energy consumed by each synapse is $36.7pJ$ for learning in [26] where they also consider a resistance range of 70Ω to 670Ω for the memristors. In [41], the energy consumed per synapse is $11pJ$ to $0.1pJ$ and the working resistance is from 1kΩ to 1MΩ. For the proposed system, we consider three different types of memristive devices (see Table II), specifically TaO_x, TiO_2 and HfO_x based on information provided in existing literature.

TABLE II
Energy consumption on Synapses with Metal-Oxide Memristors

Synapse State \ Devices	TaO_x [9]	HfO_x [20]	TiO_x [10]
	Energy per spike		
Active	$8.074pJ$	$0.48pJ$	$0.17pJ$
Idle	$0.002pJ$	$0.002pJ$	$0.002pJ$
Potentiation	$10.76pJ$	$0.65pJ$	$0.26pJ$
Depression	$10.38pJ$	$0.58pJ$	$0.13pJ$

TABLE III
Characteristics of Data Set [42]

Data Set	No. of instances	No. of inputs	No. of Output Class
$Iris$	150	4	3
$Wisconsin\ Breast\ Cancer$	699	10	2
$Prima\ Indian\ Diabetes$	768	8	2

The energy consumed by the mixed-signal IAF neuron is $7.2pJ/spike$, $9.81pJ/spike$ and $12.5pJ/spike$ for idle, accumulation and firing phase, respectively. To determine the energy, we have sampled the current through the neuron during the accumulation time and the firing time separately and then took the average values of the currents for two different phases of operation. Since we are considering the average energy per spike, we multiplied the supplied voltage with the average currents to get the average power and then determined the average energy per spike using the integration and firing phase time duration. Here we have considered energy consumption for both the analog amplifiers and digital circuit components and hence the energy per spike is considered a bit larger relative to pure analog alternatives. Further, the digital output spikes generated can be routed through the overall system more efficiently and provide greater drive strength.

We have used the CMOS 65nm PTM model for circuit-level simulation in Cadence Spectre and have determined the energy estimates for three different classification tasks from the UCI Machine Learning Repository [42] that are commonly used in the literature: Iris, Wisconsin Breast Cancer and Prima

Diabetes. Iris dataset is a set of 150 instances where each instance includes four properties of an iris flower. The breast cancer dataset consists of 699 instances and each of these includes ten different features of cell nucleus. Lastly the diabetes dataset has 768 instances with each instance representing four fields per record. For these data sets, we encode the input values as integers between 0 and 10, inclusive, by scaling the raw attributes as needed. An example network for iris classification generated from the EO is shown in Fig. 11. This network has $four$ input neurons, six hidden neurons and one output neuron. The input neurons represent four different properties of the iris flower and the single output determines the result after classifying the input flower. Similarly, the networks generated for Breast Cancer and Diabetes dataset would have multiple input neurons with synapses to represent the input signals and a single output neuron. The number of fires for the output neuron indicates the selected output class. Characteristics of the datasets are summarized in Table III.

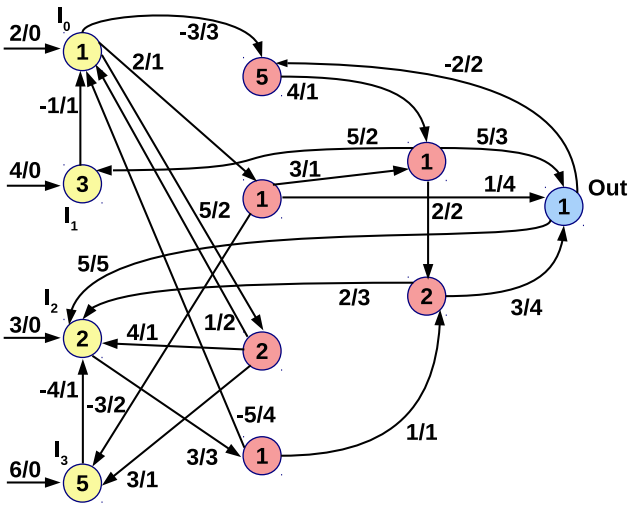

Fig. 11. One of the networks for the iris classification task, showing input neurons (yellow), output neuron (blue), and hidden neurons (red). The label for the neurons are the thresholds of the neuron and the label on the synapses are the weights followed by the delays.

Each network is defined by the number of neurons and synapses as well as the connectivity between various elements. To calculate the consumed energy for a particular application, activity factors for all the synapses and neurons have been collected from high-level network simulation. Energy per spike for different phases of operation for the synapses (active, idle, potentiation and depression) have been determined from Cadence Spectre simulations (see Table II). Then the total number of spikes corresponding to different phases of synapses are multiplied by their corresponding energy estimates to determine the total energy consumed for each application. The total energy for the three different classification applications per the three different memristive device specifications are given in Fig. 12.

Here, the iris classification task is simulated with 150 flower inputs and each classification is run for 300 cycles. From Fig. 12, we see that the energy consumed by each

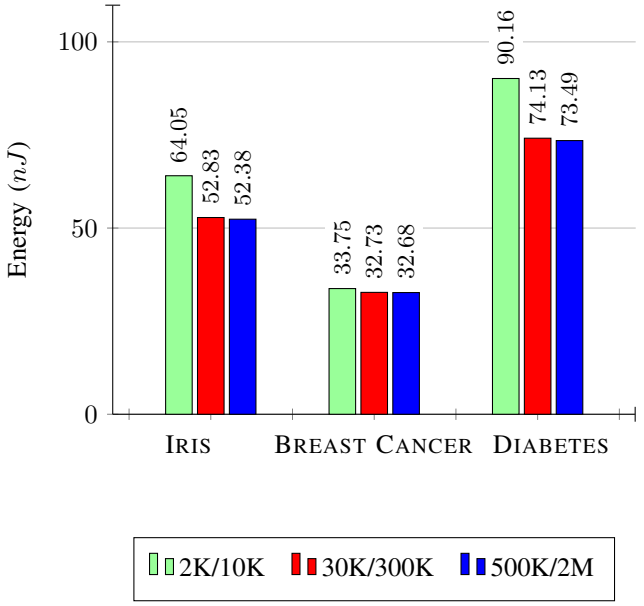

Fig. 12. Total energy per classification.

classification is lower for different memristive devices. We can assume that the higher the LRS/HRS, the lower the energy consumption will be. Similarly, we can see that same trend for the Breast Cancer and Diabetes datasets. All of these data are calculated using a clock frequency of $20MHz$ to be consistent with relative existing work in the literature [43]. However, the energy consumption per synapse per spike for a neuron in [36] has been determined considering $1MHz$ clock frequency. Moreover, the amplitude of the action potential ranges from "$-100mV$" – "$140mV$" whereas in this paper we are using digital programing pulses of $1.2V$ amplitude. Hence, the energy consumption is a bit higher in this approach because of the higher amplitude and higher clock frequency. It would be comparatively less if we simulate the neural networks with lower clock frequency. However, if we consider different metal-oxide memristors, the energy consumption can be altered effectively.

We have also analyzed the effectiveness of online learning (DLTP in this work) on classification tasks. We used our genetic algorithm to train networks with online learning to classify three different datasets. The accuracy of these networks was then tested both with online learning on and with online learning off. Each network was tested over the total dataset (Iris: 150 instances, Breast cancer: 699 instances and Diabetes: 768 instances) with the accuracy tracked as the simulation runs. The results for both trained with DLTP are shown in Fig. 13 (1st two columns) where we can see that for all three of the classification problems, the average accumulated accuracy is higher when the online learning (DLTP) is present than when it isn't, assuming the network was trained for online learning.

We have also considered the case for our networks trained without online learning and tested keeping the online learning off. The results for the stated case have been analyzed for all three databases with the results shown in Fig. 13 (3rd column).

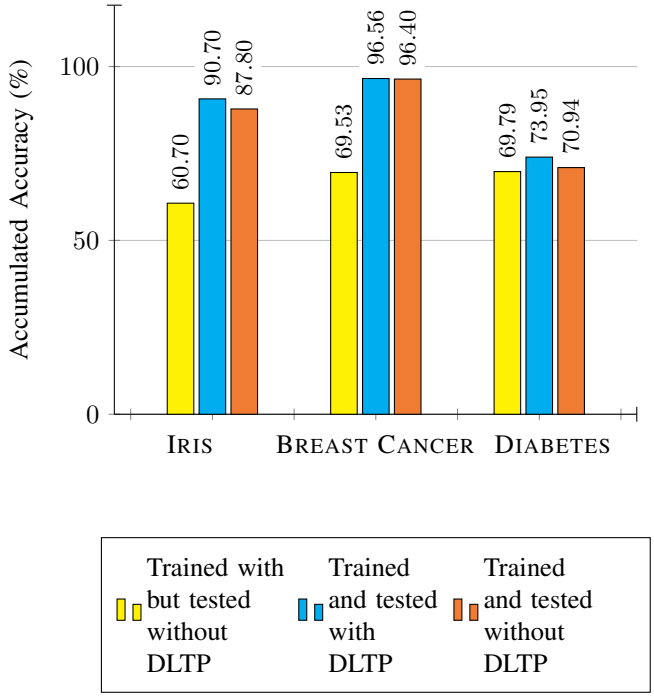

Fig. 13. Average accumulated accuracy for classification task for network trained with learning but tested with/without learning and trained/tested without learning

TABLE IV
AVERAGE NUMBER OF EPOCHS TO ACHIEVE ACCUMULATED ACCURACY

Data Set	Trained and tested without DLTP	Trained and tested with DLTP
Iris	194.2	267.2
Wisconsin Breast Cancer	37.7	108.6
Prima Indian Diabetes	299	299

Though it shows that there is only a small change in the accuracy result between the networks trained/tested with/without DLTP, the difference in average number of epochs to achieve the accuracies are different. To justify this we include Table IV which shows the average number of epochs while training and testing with DLTP is more than that achieved without DLTP. Because the EO goes through several iterations while training and testing with DLTP whereas the convergence to the highest accuracy in training/testing without DLTP does not consider better optimization steps. Hence, the DLTP process can be advantageous in achieving higher efficiency during training for classification tasks. Moreover, from Fig. 13, we can see that the average accumulated accuracy of the networks both trained with and without online learning is higher if we test keeping the online learning *on*. We have seen similar results in [44] where the maximum accuracy using an RRAM model is 85% where they are using STDP for Iris classification. In this paper, we are implementing DLTP (simple form of STDP).

For comparison with other approaches to DLTP, we have analyzed a similar technique of digital implementation of STDP [45]. This approach implements STDP with digital combinational logic which is similar but not equivalent to our approach. To implement DLTP we have used the driver logic and output control blocks shown in Fig. 3. Both of these blocks contain three NAND gates, two inverters and a Flip-flop whereas in [45], two OR gates, two AND gates and a shift register have been used to implement this simple form of STDP. The implementation in [45] considers Xilinx Spartan FPGA and leverages several LUTs and FF slices. Another approach presented in [46] also considers the implementation in Xilinx Spartan FPGA which refers to the use of block RAMs, embedded multipliers and more LUTs. Since both [45] and [46] are using LUTs and hence FPGA for the digital implementation of STDP, our approach to DLTP using CMOS 65nm PTM models is more efficient in area and power.

VI. CONCLUSION

In this paper, we have proposed a mixed-signal neuromorphic system with synchronous digital LTP approach (DLTP) for unsupervised online learning. We have implemented the design with analog nature for multiply and accumulate functionality whereas made the design more digital for communication between the cores. This mixed-signal approach led us to DLTP which is synchronous with the digital inputs. Our architecture leveraging nanoscale device with DLTP has also been designed to ensure low power and less area. We have used widely used datasets to analyze the effect of online learning in training neural network with the proposed system. It is shown that the digital approach of LTP ensures higher accuracy in classification tasks than that of without any long term plasticity (only offline training). Moreover, our digital LTP approach also ensures efficiency in power and area with mixed-signal circuit implementation. In future, the proposed memristive mixed-signal neuromorphic architecture would also be used to implement spatio-temporal applications where the DLTP method of online learning would be highly appreciated.

ACKNOWLEDGMENT

The authors would like to thank Dr. Mark Dean, Dr. James Plank, Sagarvarma Sayyaparaju, Sherif Amer and Nicholas Skuda from the University of Tennessee, Knoxville, Dr. Nathaniel Cady and Dr. Karsten Beckmann from SUNY-PI and Joseph Van Nostrand from AFRL for interesting and useful discussions on this topic.

REFERENCES

[1] J.-s. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha *et al.*, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*. IEEE, 2011, pp. 1–4.

[2] B. Linares-Barranco, E. Sanchez-Sinencio, A. Rodriguez-Vazquez, and J. L. Huertas, "A cmos analog adaptive bam with on-chip learning and weight refreshing," *IEEE Transactions on Neural networks*, vol. 4, no. 3, pp. 445–455, 1993.

[3] C. Schneider and H. Card, "Analog cmos synaptic learning circuits adapted from invertebrate biology," *IEEE transactions on circuits and systems*, vol. 38, no. 12, pp. 1430–1438, 1991.

[4] C. Schneider and H. Card, "Cmos implementation of analog hebbian synaptic learning circuits," in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. 1. IEEE, 1991, pp. 437–442.

[5] H. Graf, L. Jackel, R. Howard, B. Straughn, J. Denker, W. Hubbard, D. Tennant, D. Schwartz, and J. S. Denker, "Vlsi implementation of a neural network memory with several hundreds of neurons," in *AIP conference proceedings*, vol. 151, no. 1. AIP, 1986, pp. 182–187.

[6] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1, pp. 239–255, 2010.

[7] L. O. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, September 1971.

[8] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[9] J. J. Yang, M. Zhang, J. P. Strachan, F. Miao, M. D. Pickett, R. D. Kelley, G. Medeiros-Ribeiro, and R. S. Williams, "High switching endurance in taox memristive devices," *Applied Physics Letters*, vol. 97, no. 23, p. 232102, 2010.

[10] G. Medeiros-Ribeiro, F. Perner, R. Carter, H. Abdalla, M. D. Pickett, and R. S. Williams, "Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution," *Nanotechnology*, vol. 22, no. 9, p. 095702, 2011.

[11] H. Lee, Y. Chen, P. Chen, T. Wu, F. Chen, C. Wang, P. Tzeng, M.-J. Tsai, and C. Lien, "Low-power and nanosecond switching in robust hafnium oxide resistive memory with a thin ti cap," *IEEE Electron Device Letters*, vol. 31, no. 1, pp. 44–46, 2010.

[12] Y. Li, Y. Zhong, L. Xu, J. Zhang, X. Xu, H. Sun, and X. Miao, "Ultrafast synaptic events in a chalcogenide memristor," *Scientific Reports*, vol. 3, p. 1619, 2013.

[13] A. S. Oblea, A. Timilsina, D. Moore, and K. A. Campbell, "Silver chalcogenide based memristor devices," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–3.

[14] K. D. Cantley, A. Subramaniam, H. J. Stiegler, R. A. Chapman, and E. M. Vogel, "Neural learning circuits utilizing nano-crystalline silicon transistors and memristors," *IEEE transactions on neural networks and learning systems*, vol. 23, no. 4, pp. 565–573, 2012.

[15] A. Mehonic, S. Cueff, M. Wojdak, S. Hudziak, O. Jambois, C. Labbé, B. Garrido, R. Rizk, and A. J. Kenyon, "Resistive switching in silicon suboxide films," *Journal of Applied Physics*, vol. 111, no. 7, p. 074507, 2012.

[16] T. Berzina, S. Erokhina, P. Camorani, O. Konovalov, V. Erokhin, and M. Fontana, "Electrochemical control of the conductivity in an organic memristor: a time-resolved x-ray fluorescence study of ionic drift as a function of the applied voltage," *ACS Applied materials & interfaces*, vol. 1, no. 10, pp. 2115–2118, 2009.

[17] A. Chanthbouala, V. Garcia, R. O. Cherifi, K. Bouzehouane, S. Fusil, X. Moya, S. Xavier, H. Yamada, C. Deranlot, N. D. Mathur *et al.*, "A ferroelectric memristor," *Nature materials*, vol. 11, no. 10, pp. 860–864, 2012.

[18] A. Panchula, "Oscillating-field assisted spin torque switching of a magnetic tunnel junction memory element," May 29 2007, uS Patent 7,224,601.

[19] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[20] M. Uddin, M. B. Majumder, G. S. Rose, K. Beckmann, H. Manem, Z. Alamgir, and N. C. Cady, "Techniques for improved reliability in memristive crossbar puf circuits," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*. IEEE, 2016, pp. 212–217.

[21] S. Amer, S. Sayyaparaju, G. S. Rose, K. Beckmann, and N. C. Cady, "A practical hafnium-oxide memristor model suitable for circuit design and simulation," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*.

[22] G. S. Rose, H. Manem, J. Rajendran, R. Karri, and R. Pino, "Leveraging memristive systems in the construction of digital logic circuits," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 2033–2049, June 2012.

[23] C. E. Merkel and D. Kudithipudi, "A current-mode cmos/memristor hybrid implementation of an extreme learning machine," in *Great Lakes Symposium on VLSI 2014, GLSVLSI '14, Houston, TX, USA - May 21 - 23, 2014*, 2014, pp. 241–242. [Online]. Available: http://doi.acm.org/10.1145/2591513.2591572

[24] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 10, pp. 1864–1878, October 2014. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2013.2296777

[25] I. Kataeva, F. Merrikh-Bayat, E. Zamanidoost, and D. Strukov, "Efficient training algorithms for neural networks based on memristive crossbar circuits," in *2015 International Joint Conference on Neural Networks, IJCNN*, July 2015, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/IJCNN.2015.7280785

[26] M. Hu, Y. Chen, J. J. Yang, Y. Wang, and H. Li, "A memristor-based dynamic synapse for spiking neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[27] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nature communications*, vol. 4, 2013.

[28] S. H. Jo, K.-H. Kim, and W. Lu, "High-density crossbar arrays based on a si memristive system," *Nano letters*, vol. 9, no. 2, pp. 870–874, 2009.

[29] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2011.

[30] Q. Xia, W. Robinett, M. W. Cumbie, N. Banerjee, T. J. Cardinali, J. J. Yang, W. Wu, X. Li, W. M. Tong, D. B. Strukov *et al.*, "Memristor-cmos hybrid integrated circuits for reconfigurable logic," *Nano letters*, vol. 9, no. 10, pp. 3640–3645, 2009.

[31] X. Wu, V. Saxena, and K. Zhu, "A cmos spiking neuron for dense memristor-synapse connectivity for brain-inspired computing," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–6.

[32] G. Bi and M. Poo, "Synaptic modification by correlated activity: Hebb's postulate revisited." *Annual review of neuroscience*, vol. 24, no. 1, pp. 139–166, 2001.

[33] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "Stdp and stdp variations with memristors for spiking neuromorphic learning systems," *Frontiers in Neuroscience*, vol. 7, no. 2, February 2013.

[34] G. S. Snider, "Spike-timing-dependent learning in memristive nanodevices," in *IEEE International Symposium on Nanoscale Architectures, 2008 (NANOARCH)*, June 2008, pp. 85–92.

[35] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, March 2010.

[36] X. Wu, V. Saxena, and K. A. Campbell, "Energy-efficient STDP-based learning circuits with memristor synapses," in *Proceedings of SPIE*, vol. 9119, 2014, pp. 911 906–1–911 906–7. [Online]. Available: http://dx.doi.org/10.1117/12.2053359

[37] C. A. Mead, in *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[38] R. Weiss, G. Chakma, and G. S. Rose, "A synchronized axon hillock neuron for memristive neuromorphic systems," in *Proceedings of IEEE International Midwest Symposium on Circuits and Systems MWSCAS*, Boston, Massachusetts, August 2017.

[39] C. D. Schuman, J. S. Plank, A. Disney, and J. Reynolds, "An evolutionary optimization framework for neural networks and neuromorphic architectures," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 145–154.

[40] M. E. Dean, J. Chan, C. Daffron, A. Disney, J. Reynolds, G. Rose, J. S. Plank, J. D. Birdwell, and C. D. Schuman, "An application development platform for neuromorphic computing," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 1347–1354.

[41] K. A. Campbell, K. T. Drake, and E. H. B. Smith, "Pulse shape and timing dependence on the spike-timing dependent plasticity response of ion-conducting memristors as synapses," *Frontiers in Bioengineering and Biotechnology*, vol. 4, 2016.

[42] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[43] C. Liu, Q. Yang, B. Yan, J. Yang, X. Du, W. Zhu, H. Jiang, Q. Wu, M. Barnell, and H. Li, "A memristor crossbar based computing engine optimized for high speed and accuracy," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*. IEEE, 2016, pp. 110–115.

[44] A. Shukla, V. Kumar, and U. Ganguly, "A software-equivalent snn hardware using rram-array for asynchronous real-time learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 4657–4664.

[45] A. Cassidy, A. G. Andreou, and J. Georgiou, "A combinational digital logic approach to stdp," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, May 2011, pp. 673–676.

[46] B. Belhadj, J. Tomas, Y. Bornat, A. Daouzli, O. Malot, and S. Renaud, "Digital mapping of a realistic spike timing plasticity model for real-time neural simulations," in *Proceedings of the XXIV conference on design of circuits and integrated systems*, 2009, pp. 1–6.

Gangotree Chakma received her B.Sc. in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 2014. Presently, she is continuing her Ph.D. in Electrical Engineering at University of Tennessee, Knoxville, TN. She is an active member of IEEE. Her research interest includes neuromorphic computing, mixed-signal circuit design, emerging nano-devices and low-power VLSI designs.

Md Musabbir Adnan received his B.Sc. in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh in 2015. Starting August 2016, he is pursuing his PhD in Computer Engineering at University of Tennessee, Knoxville. He is a member of SENECA research group of University of Tennessee, Knoxville and his research interests include neuromorphic computing, VLSI design of nanoelectronic circuits and emerging devices.

Austin R. Wyer received his B.S. in Computer Science from the University of Tennessee, Knoxville, in 2016. Presently, he is pursuing a Ph.D. in Computer Science at the University of Tennessee, Knoxville. His research interest includes Machine Learning and Graph Theory.

Ryan Weiss received his B.S. in Electrical Engineering from the University of Tennessee, Knoxville, in 2016. Presently, he is pursuing a Ph.D. in Electrical Engineering at the University of Tennessee, Knoxville. His research interest includes analog circuit design, Neuromorphic computing, and emerging devices.

Katie Schuman is a Liane Russell Early Career Fellow in Computational Data Analytics at Oak Ridge National Laboratory. Katie received her B.S. in Computer Science and Mathematics from the University of Tennessee in 2010. She received her doctorate in Computer Science from the University of Tennessee in 2015, where she completed her dissertation on the use of evolutionary algorithms to train spiking neural networks for neuromorphic systems. She is continuing her study of models and algorithms for neuromorphic computing as part of her fellowship at ORNL. Katie also has a joint faculty appointment with the Department of Electrical Engineering and Computer Science at the University of Tennessee, where she, along with four other professors at UT, leads a neuromorphic research team.

Garrett S. Rose (S'98-M'06) received the B.S. degree in computer engineering from Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, in 2001 and the M.S. and Ph.D. degrees in electrical engineering from the University of Virginia, Charlottesville, in 2003 and 2006, respectively. His Ph.D. dissertation was on the topic of circuit design methodologies for molecular electronic circuits and computing architectures.

Presently, he is an Associate Professor in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville where his work is focused on research in the areas of nanoelectronic circuit design, neuromorphic computing and hardware security. Prior to that, from June 2011 to July 2014, he was with the Air Force Research Laboratory, Information Directorate, Rome, NY. From August 2006 to May 2011, he was an Assistant Professor in the Department of Electrical and Computer Engineering at the Polytechnic Institute of New York University, Brooklyn, NY. From May 2004 to August 2005 he was with the MITRE Corporation, McLean, VA, involved in the design and simulation of nanoscale circuits and systems. His research interests include low-power circuits, system-on-chip design, trusted hardware, and developing VLSI design methodologies for novel nanoelectronic technologies.

Dr. Rose is a member of the Association of Computing Machinery, IEEE Circuits and Systems Society and IEEE Computer Society. He serves and has served on Technical Program Committees for several IEEE conferences (including ISVLSI, GLSVLSI, NANOARCH) and workshops in the area of VLSI design. In 2010, he was a guest editor for a special issue of the ACM Journal of Emerging Technologies in Computing Systems that presented key papers from the IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH09). From April 2014 through March 2017 he was an associate editor for IEEE Transactions on Nanotechnology.