Memristor Crossbar Architecture for Synchronous Neural Networks
Janusz A. Starzyk, Senior Member, IEEE, and Basawaraj

Abstract—This paper focuses on suitable architecture and neural network training using crossbar connections of memristive elements. We developed a novel memristor training scheme that preserves high density of synaptic connections in the crossbar organization. We designed supporting circuits and performed time domain analog simulation of the architecture, to demonstrate that it properly adjusts memristor values during neural network training. A single sensing and winner-takes-all circuit is used to adjust strength of synaptic connections implemented by all memristors. We present results of HSPICE simulation of the developed architecture, generated control signals and resulting changes of memristor values. We used crosstalk test and Monte Carlo analysis to demonstrate robustness of the proposed architecture. Tests performed on MNIST character recognition benchmark confirmed functionality of the proposed circuit and training scheme in a practical and demanding application. The proposed approach improves available in the literature architecture and training methods for memristive neural networks.

Index Terms—Dense analog memories, memristive cross-bar architecture, neural network self-organization.

I. INTRODUCTION

MEMRISTOR, an element relating charge and magnetic flux, was first hypothesized by Chua 40 years ago [1], [2]. These devices change their resistance based upon the input current or voltage, i.e., the resistance decreases if a positive signal is applied across its terminals and increases if a negative signal is applied. Though resistance switching was observed in a variety of oxides [3], it was only in 2008 that the switching in Titanium dioxide (TiO$_2$) was recognized as related to the memristor characteristics [4]. Chua argues [5] that all resistive two terminal devices capable of switching between different resistance values are memristors, and that the distinguishing feature of a memristor is pinched hysteresis loop.

Since the observation of memristor characteristics in [4] there has been considerable interest in memristors, memristive devices, and their potential applications. Recent work [6] shows that it is possible to fabricate memristors with high nonlinearity in the low resistance regions thus enabling fabrication of large crossbar arrays. A major gain in using memristors, which results from their small area and low power requirements, is their ability to produce high density devices embedded in the integrated circuit. Hence, in addition to potential applications in analog memories, memristors could be very useful in building practical neural networks with large number of neurons and synaptic connections. For example a neuromorphic network using memristive crossbar circuit was fabricated for pattern classification and is presented in [7], while [8] presents design and simulation of a scalable neural chip with about 73 728 memristors.

Memristors have been applied to a variety of areas, such as building associative memory [9], design of visual cortex [10], signal multiplication [11], position detection [12], logic blocks [13], [14], and synapses of neural networks [15]–[18]. Due to the absence of commercial memristive devices researchers tend to either model the devices based on [4] or resistive memories such as those in [5]. In addition researchers can also use SPICE macro models, such as those proposed in [19]–[21]. An emulator for TiO$_2$ memristor based on available solid state components is presented in [22] and an application of such model to design bridge synaptic circuit is presented in [23].

Many problems in graph theory, networking, neural networks, image processing, and related fields require massively parallel computations, and memristors can potentially be used to solve such problems. For instance, [24] proposes a memristor based approach to solving massively parallel maze problem, and shows that the algorithm is able to find multiple solutions. Memory organization and storage is another field where memristors can play a very important role due to their small size and low power requirement. Reference [25] proposes a method to store images using a crossbar array structure that can be easily scaled to implement large size memories. However, without development of memristor based image processing techniques such storage mechanism would not be very interesting.

A major evaluation criterion of memristive architecture in neural network applications is the ability to perform on-line training and the ability to scale the architecture to a large system. The architectures shown in [15]–[18] have potential for such scaling. References [17], [18] use memristive weighting circuits that can be used as synapses. The weighting circuits are in the form of a bridge network and use four and five memristors, respectively. The advantage of the bridge network is that they can be used to generate both positive and negative weights. The memristor bridge synapse of [17] has been used in multilayer neural network circuit for chip-in-the-loop learning [26].

But the drawback of both of these architectures is that they require weight programming (to set the weights) and weight processing (through a specialized circuit) stages. This limits their neural network applications, since off-line training needed in these architectures precludes on-line self-organization and learning often required in neural networks.
The approaches proposed in [15], [16] are better suited for networks using on-line training. Both these works use the leaky integrate-and-fire neuron models proposed by Mead [27]. The synapses used consist of a transistor switch in addition to memristor. The architecture proposed in [15] can be easily used in a memristor crossbar network since the transistor switch in the synapse could be shared between various synapses that have the same presynaptic neuron.

The architecture presented in [16] provides better control but the synapse was modified making it less useful for IC crossbar implementation. The most limiting factor that significantly reduces memristor density is that each synapse requires a feedback transistor. Analysis of the architecture proposed in [16] shows that the synapse is capable of bidirectional operation but detailed analysis of such operation has not been shown. Also, details of the memristor synapse training are unclear. Though highly dense approaches to memristor based crossbar architecture with memristor synapses are proposed in [28], [29], they lack some details. In [28] details of training algorithms are not provided, while [29] lacks in details of the circuitry needed to implement homeostasis and lateral inhibition of the output neurons.

Our aim was to provide details of the required circuitry and overcome the difficulties through network organization, training regime, and proper timing to make memristive on-line learning feasible in high density crossbar architectures. Specifically, we propose a neural network organization and training scheme, such that high density architectures with interconnection weights that can be easily implemented and controlled during the network operation by applying proper control signals to postsynaptic neurons feedback lines. Proper timing of the feedback signals and output signals is used to obtain desired weight adjustments. In [30] we proposed and compared two memristor based neural network learning schemes suitable for crossbar architectures. In this work we focused on circuit implementation of such learning schemes including the design and test of various circuits, such as sensing circuit, biasing circuit and overall crossbar architecture. We described operations of these circuits and memristor weight adjustment schemas. We also provide details of the memristor weight changes that would result under various conditions.

This paper is developed using popular memristor model briefly discussed in Section II. Section III contains an overview of memristor based neural network. Basic implementation of neuron synaptic connections and the weight adjustment scheme are discussed. In Section IV we present design and simulation of neural network circuits. This includes neuron and sensing circuit organization, as well as generated control signals and simulation waveforms. Tests on MNIST character recognition benchmark are provided in Section V. Section VI presents the conclusion.

II. BACKGROUND

A. Memristor Characteristics

Memristors are elements that can change their resistance based on the integral of the voltage applied. Fig. 1(a) shows the memristor’s physical model from [4]. It consists of two thin layers, an insulating titanium dioxide (TiO$_2$) and an oxygen poor TiO$_{2-x}$ layer, sandwiched between platinum contacts. The insulating layer is considered the undoped region and has high resistance while the oxygen poor region is considered the doped region and, due to the oxygen vacancies acting as positive dopants, it behaves as a semiconductor with low resistance. Here D and W are device and doped region thicknesses respectively.

The effective resistance is the sum of the resistances of the doped and undoped regions as

$$ M(x) = R_{ON}x + R_{OFF}(1 - x) $$

where

$$ x = \frac{W}{D} \in (0, 1) $$

and is known as the state variable. The memristor current and voltage are related through Ohm’s law, with memristance $M(x)$ representing current resistance value. $R_{ON}$ and $R_{OFF}$ are per unit length resistances of doped and undoped regions and for the device from [4] these values are 116 $\Omega$ and 15.8 k$\Omega$, respectively.

Polarization of the applied voltage is important in memristors. When a positive voltage is applied across memristor in a forward bias (positive at doped region and negative at undoped region) the memristor increases its conductivity, and if it is applied in a reverse bias the conductivity goes down. In this paper, memristors with a positive bias are referred to as p-type memristors and those with reverse bias are referred to as n-type. Fig. 1(b) shows the widely used memristor symbol along with the symbols for the p-type and n-type memristors that are used in this paper for discussion of the proposed neural network architecture. Also shown are the polarities of memristor voltages when the memristor is forward biased.

Experiments and analysis performed subsequent to publication of [4] has furthered the understanding of the memristor characteristics. For example, [31] predicted that the speed of switching ON was faster than switching OFF; applying ON-switching for sufficiently long time would turn the device OFF i.e., the switching polarity can be reversed. These devices have ohmic $i$-$v$ characteristics near the ON states and are non-linear towards the OFF states. Moreover, in [32] it was shown that increasing the applied current caused an exponential decrease in the required switching energy. In this work, we use the memristor SPICE macro model, proposed by Biolek et al., [19] based on [4]. In this model the memristor is implemented as a series combination of resistance $R_{OFF}$ and a controlled voltage source. The model from [4] was used due to its simplicity and wide use in literature. Using a more detailed model, would not considerably affect the algorithm or the results presented here. The various memristor parameters for the macro
model used are specified in Table I, were $\mu_e$ is the migration coefficient (i.e., dopant mobility) used to compute internal current, $x$ is the internal voltage in the memristor model, stp is the unit step function and $i$ is the memristor current.

It is necessary to know the resulting changes in conductance as a function of applied signal. Fig. 2(a) shows the memristor conductance as a function of the flux linkage. We see that the rate of change of conductance changes as the flux linkage increases. Fig. 2(b) plots the rate of conductance change as a function of the initial conductance value when a pulse waveform of $V$ and ON time of 4.5 ms is applied across the memristor (upper curve) and when a similar pulse of $V$ is applied (lower curve). The memristor is most sensitive to the applied voltage around conductance value of 6 mS.

### III. Memristor Based Neural Network

Research has shown potential applications of memristors in a variety of applications in which dense analog memories can be used, especially those related to neural networks. In large neural networks it is necessary that the building blocks, neurons, synapses, etc., are both small and operate at low power levels. Memristors have all these qualities. To advance the goal of building large scale neural systems we propose a new learning algorithm and supporting architecture for memristor based neural networks. The proposed architecture uses simple threshold based neurons, memristor based synapses and a common sensing network. Each synapse consists of only a single memristor. This feature of the synapse is advantageous and enables extension of the proposed architecture to a large scale system in which synapses are arranged in a grid structure and are trained on-line during the network operation.

#### A. Interconnection Links in a Neural Network

This section explains how we can simultaneously use and program neural network interconnection links for on-line learning. Thus we can use memristors fabricated using nanowire crossbar technology to construct and program high density neural network circuits.

In this work we consider a simple single layer feed-forward neural network with $n$ inputs and $m$ outputs. We also use a winner-takes-all competition [30] between output neurons to choose which weights will be adjusted. According to Hebbian learning, in a simple feedforward neural network, excitatory links increase their weights when both presynaptic and postsynaptic neurons fire. We adopt this rule to the memristor circuit with different schemas of weight adjustment for excitatory and inhibitory neurons.

For presynaptic excitatory neuron, the weight that connects this neuron to a postsynaptic excitatory neuron changes according to activation of both neurons as follows:

$$\Delta w_i = (V_i + V_N - 1) \Delta_i(G)$$

where $V_i$ is the output signal value of presynaptic neuron $x_i$, $V_N$ is the output signal value of postsynaptic neuron $N$, and $\Delta_i(G)$ is a small weight increment that corresponds to change in memristor conductance $G$. $V_i$ and $V_N$ have voltages in the range $[0, 1]$. Output neuron activation is a nonlinear function of its input activation,

$$V_N = f(y)$$

where $y = \sum w_i V_i$ and $\sum_i w_i = 1$ (see Fig. 3). Typically f(y) is a sigmoidal function.

As a result, excitatory neurons increase their output weight connection to another excitatory neuron when both neurons fire. Neurons reduce their output weights when neither presynaptic nor postsynaptic neurons fire. In this work we assume for convenience that for activated excitatory neuron, neuron’s output voltage is $V_{out} = 1 [V]$, while when the neuron is not active its output voltage is $V_{out} = 0 [V]$. In addition, we represent the inhibitory neuron as inverted excitatory neuron, thus if inhibitory
neuron fires its output $V_{\text{out}} = 0$ [V] and when it does not fire $V_{\text{out}} = 1$ [V].

If the presynaptic neuron is inhibitory and postsynaptic neuron is excitatory then when presynaptic neuron does not fire and postsynaptic fires the weight is increased and it is decreased if presynaptic neuron fires while postsynaptic neuron does not fire. In both cases $\Delta W_{ji}$ is adjusted to keep all normalized interconnection weights within $[0, 1]$ interval.

If the presynaptic neuron is excitatory and postsynaptic neuron is inhibitory the weight is decreased when presynaptic neuron does not fire and postsynaptic neuron fires, while it is increased when presynaptic fires and postsynaptic does not. Finally, if both the presynaptic and postsynaptic neurons are inhibitory the weight increases when none of them fires and the weight decreases when both of them fire.

These weight adjustments can be accomplished with memristors as follows. Fig. 4 shows a simplified schema of the input to a neuron $N$ in a crossbar architecture with a number of memristor elements connected to vertical bars crossing this neuron input (horizontal) line. To simultaneously use the memristive elements as weights of the neuron and to adjust these weights, depending on the presynaptic and postsynaptic neurons activity, we will consider sensing and active phases of neurons operation. These phases are responsible for triggering response to stimuli and training of the neural network, respectively.

While the first phase is easy to implement, the second phase is not, if one wants to train the neural network on line with all interconnection weights adjusting gradually according to Hebbian learning. Yet this is a desired approach, since the alternative—off line learning, require first training of a neural network off line, then programming its memristive connections one by one. Memristor programming in such scheme requires selecting the corresponding crossing between a vertical line that represents the output of presynaptic neuron and horizontal line that represents the input of postsynaptic neuron and sequential setting of all the memristor values one by one. Memristor value is set by adjusting the memristor programming time that regulates total flux linkage. Detailed implementation of our proposed on-line training scheme will be presented in Section IV.

For simplicity let us consider the case when both presynaptic and postsynaptic neurons are excitatory. When the neuron $N$’s input $V$ in Fig. 3 is charged to the voltage $V_y$ above a threshold value $V_{\text{th}}$, the neuron fires and subsequently resets $V_y$ to the resting potential $V_r$.

Thus when an input neuron $x_i$ fires it sets its output voltage to $V_{\text{out}}$ for a short period of firing time $T_f$. If the input potential $V_y$ of the output neuron $N$ has zero value, this produces a flux and increases the conductance of a p-type memristor. If, as a result of combined firing activities of its input neurons the output neuron $N$ fires, it reduces its input potential $V_y$ to zero. Thus with $1$ [V] on the active input $V_i$, the corresponding memristor increases its conductance value. However, if $N$ does not fire, then it changes its input potential $V_y$ to $1$ [V] for a short period of firing time $T_f$. This produces a negative flux across memristor connected to the inactive input with $V_i = 0$ [V] reducing its conductance (weight).

For this scheme to work properly, the positive flux across memristor with the input $V_i = 1$ [V] of the firing postsynaptic neuron must be on a similar level as the negative flux across memristor with the input $V_i = 0$ [V] of the inactive postsynaptic neuron. Neuron’s activity will be regulated by slowly discharging its load capacitance.

The combined input voltage $V_y$ is used to charge a large input capacitor. This voltage can be obtained as follows. If we assume steady state with capacitor current $i_C = 0$ in Fig. 4, then

$$\sum (V_i - V_y)G_i = 0$$

(6)

Subsequently $V_y$ can be obtained from

$$\sum V_iG_i - V_y \sum G_i = 0$$

(7)

$$V_y = \frac{\sum V_iG_i}{\sum G_i} = \sum w_iV_i$$

(8)

where

$$w_i = \frac{G_i}{\sum G_i}$$

(9)

where $G_i$ is the conductance of ith memristor, and $V_y$ can be thought of as a combined input to the neuron.

Thus the input signal is a weighted sum of the excitations and the sum of weights is equal to 1. As a result, numerical values of memristor conductances $G_i$ correspond to neural network weights between presynaptic and postsynaptic neurons. For activation of postsynaptic neuron, $V_y$ must be larger than $V_{\text{th}}$ and for that high input values $V_i$ must correspond to high weights $w_i$.

B. Neuron’s Basic Operation

Neuron has three phases of operation:

a) Sensing
b) Active
c) Resting

Sensing phase is when neuron waits for its input activities and does not fire. During this phase any increase of any input signal $V_y$ above threshold causes the neuron to change to the active phase. During active phase a neuron either fires or not for a specific amount of time $T_f < T_{\text{max}}$ (where $T_{\text{max}}$ is the neural networks input sampling rate) that depends on its activation level (represented here by the value of its input signal $V_y$). After the active phase, the neuron is in the resting phase for the reminder of time $T_{\text{max}} - T_f$. During resting phase all inputs and outputs go to zero and all capacitances are discharged. When a neuron is in the active phase its output activation level is used to change the input interconnection weights. The functioning of
the network is synchronized by proper selection of RC components and controlling the rate at which input signals are applied to the neural network.

A simple schema was selected here for implementing weights adjustment in memristor crossbar network. The motivating argument for such weight adjustment schema is to be able to adjust interconnection weights during the network operation. This precludes addressing a single neuron and adjusting its interconnection weights individually. Thus only the aggregated inputs from all presynaptic neurons $V_x$ and resulting postsynaptic neuron’s activation together with the individual input signal values $V_i$ are needed to perform weight adjustments. The remaining parts of this section describe weight adjustments in postsynaptic neurons.

**Postsynaptic excitatory neuron**

Synaptic weight from input $V_i$ to the postsynaptic excitatory neuron will be adjusted as follows. Once an input activity is detected, the neuron samples its input value and if the sampled input signal $V_x$ is greater than a prespecified threshold it fires setting its output $V_{out}$ to a high value otherwise the output is low. Neuron fires for a specific amount of time $T_f < T_{max}$ and during this time its weight adjustment takes place. This is accomplished by providing a feedback input signal from the output to neural input, forcing a new value $V_f = V_{out}$.

The weight adjustments are a result of the memristor voltage polarization. In case of a p-type memristor, when its voltage is positive its conductance increases which corresponds to the weight increase in the neural network. When its voltage is negative the conductance decreases and the weight adjustment is obtained from:

$$\Delta w_i = (V_i - V_f) \Delta_i(G)$$  \hspace{1cm}(10)$$

where $\Delta_i(G)$ is the increment of $\Delta w_i$ per unit voltage applied across the p-type memristor. To obtain similar simple weight adjustments for the input of excitatory neuron driven by an inhibitory neuron, we must use n-type memristor. When the voltage across such memristor is positive its conductance goes down and it goes up when the voltage is negative. As illustrated in Fig. 2b $\Delta_i(G)$ is not constant but depends on the present value of conductance and the amount of flux received. Thus in addition to the voltages applied the rate of learning, i.e., speed at which a memristor is trained in the desired direction, depends on the current conductance value.

To obtain a desired weight adjustment we must connect n-type memristor with the plus sign facing the neuron input $V_i$ and minus sign on the side of the postsynaptic neuron voltage $V_y$. As a result the weight adjustment is obtained from:

$$\Delta w_i = -(V_i - V_y) \Delta_i(G)$$  \hspace{1cm}(11)$$

**Postsynaptic inhibitory neuron**

If the postsynaptic neuron is inhibitory, during its activation its output voltage is zero. We can use a similar schema to provide weight adjustments for such neuron. First, let us consider the case when input signals to such neuron are from excitatory neuron with its output voltage $V_x$. If the output of activated inhibitory neuron is fed back to force new values for $V_y$, the weight adjustment is obtained from (10). This requires a p-type memristor between the output of presynaptic neuron and the input to postsynaptic neuron. Finally, when both presynaptic and postsynaptic neurons are inhibitory weight adjustment are obtained from (11).

Thus, if presynaptic neurons are excitatory we use p-type memristors, but if they are inhibitory we use n-type memristors. In addition, active inhibitory neurons provide output signal equal to 0, while inactive neurons provide output signal equal to 1. Finally, active excitatory neurons provide output signal equal to 1, while inactive neurons provide signal equal to 0. These results are collected in Table II that shows types of memristors and output voltages in all 4 combinations: where memristor + terminal is on the site of presynaptic neuron and its voltage is $V_i$, $V_y$. Such unified properties simplify design of memristive neural networks in which inhibitory and excitatory neurons can be easily combined without need negative memristor weights.

### IV. DESIGN OF NEURAL CIRCUITS

Let us assume for simplicity that both presynaptic and postsynaptic neurons are excitatory. To perform desired operations for weight adjustment, a sensing module is needed that senses activation of the neuron’s inputs. The input sensing circuit needs to respond to a small deviation from its input equal to zero, changing its output from logic one to zero. In a simplified scenario, an inverter with a low logical threshold can be used for sensing circuit. This is used only to explain the main operating principles, while real sensing circuit used in this work is described in Section IV-B.

The input sensing circuit after sensing input activity should, with some delay, disconnect the neuron’s inputs from presynaptic neurons and connect the output feedback to overwrite the signal $V_y$ value as illustrated in Fig. 5.

If the neuron is activated above threshold, it will fire providing zero feedback voltage $V_f = 0 \ [V]$, thus $V_y = 0 \ [V]$, otherwise $V_y = 1 \ [V]$, as required in Table II. Since the feedback voltage $V_f$ is provided with small delay the input memristors will be subject to $V_i$-$V_y$ voltage as specified in Table II most of the time when the neuron is activated, resulting in a desired weight adjustment.
Duration of the feedback activation period is timed in accordance with duration of the neuron’s activation, and is a function of the signal $V_y$ value in the moment when the sensing circuit disconnected the neuron’s input from presynaptic neurons. This initial value of the signal $V_y$ is stored on the input capacitance and will be slowly discharged during the neurons active period. Once it is discharged below the logical threshold of the inverter, the output signal will go low, switch $S$ will return to its original position, and the feedback will be disconnected.

An important function is to synchronize the input sensing circuit with neuron’s activation period. This can be accomplished by providing similar time constants for the dynamic input storage in the input sensing circuit and the neurons input, such that they retain their activation long enough to program the input memristors.

An example implementation of an excitatory neuron with the input sensing circuit and feedback connection is shown in Fig. 6. The postsynaptic neuron input $V_y$ is obtained from (8) as

$$V_y = \sum \frac{V_{i,x} G_i}{\sum G_i}$$

where the input signals $V_{i,x}$ are outputs of the ith presynaptic neurons (neurons from the previous layer).

Since it may be difficult to adjust leakage time for the sensing circuit, we design it to leak out just after the activities of the presynaptic neurons were finished. This will prevent additional sampling of the input signal and may cause short time period of undesirable feedback and memristor weight adjustment. This is described in Section IV-A.

Table III shows adjustments of memristor values for outputs of excitatory neurons. Similar results can be obtained for outputs from inhibitory neurons, with the exception that the signs of weights adjustments will be opposite to those shown in Table III.

In Table III $k, k_1,$ and $k_2$ are voltages in feedforward and feedback modes within the range $[0, 1]$, and represent the non-ideal case. That is, due to either voltage drops or multi-path effects the feedforward and feedback voltages are different from their ideal values of 0 or 1. In Table III $V_{i,x}$ and $V_{i,y}$ are the output voltages of the pre-synaptic and post-synaptic neurons.

Since a small change in the memristor weight $\omega \Delta w_i$ (in rows 5, 6, and 9) does not depend on the neuron’s output value, it may cause undesirable effect (lowering the weights of inactive inputs even though the output neuron fired or increasing the weight of the active inputs even though the output neuron did not fire). The amount of change depends on the value of parameter $\alpha < 1$ and the original neuron’s input activation value.

Fig. 7(a) shows the feedback connection and neuron structure for an excitatory neuron. An inhibitory neuron [Fig. 7(b)] has one less inverter on the output and drives n-type memristors. According to weight adjustment scheme presented in Section III, inhibitory and excitatory neurons are differentiated by their internal working and type of memristors that they drive. Fig. 7 shows typical connection structure for excitatory and inhibitory neurons. Inputs of these neurons can be freely connected to either inhibitory or excitatory neurons without restrictions.

The overall crossbar architecture connecting two layers of neurons is shown in Fig. 8. The input and output layers are connected by memristor devices acting as synapses. With crossbar architecture, the output neuron directly receives the sum of voltages of its input neurons (8). This architecture corresponds to a single layer feed-forward neural network. In this figure, the sensing network responds to any activation of the output of presynaptic neurons and is used both as a global feedback control and input disconnect signal to postsynaptic neurons.
A. Sensing and Training

Sensing neurons activities are performed by a dedicated circuit that detects any input activities on presynaptic neurons to the neural network. Only a single such circuit is needed to control neural network operations. Its role is to start and end activation of the neural network output neurons, provide the control to feedback signals for adjusting memristor values, and generate self-timing control signal regulating the flow of information throughout the network. Since the neural network is event driven this circuit regulates firing frequency in response to the neurons activities. An example implementation of this sensing network is shown in Fig. 9.

The sensing network must sense activation of the presynaptic neurons, even if only a small percentage of them are active. This may be accomplished by lowering the logical threshold value in the inverter \( I_{\text{in}} \) in Fig. 9. On the other hand, since the sensing circuit controls neuron activation time as well as feedback duration time, its internal time constant must be large and independent from the level of the sensing signal \( V_{\text{in}} \). Because neuron’s activation time is on the order of 5 milliseconds, the time constant

\[
\tau_{\text{act}} = R_{\text{act}}C_{\text{act}}
\]  

must be properly adjusted to match neurons’ activation time.

Since sensing circuit senses activation of presynaptic neurons and active inhibitory neurons have output equal to zero, we must use inverters before feeding inhibitory outputs to sensing circuits. Thus both excitatory and inhibitory neurons must use two inverters, however internal wiring from these neurons to the crossbar architecture is different as shown in Fig. 8. Notice that sensing circuit uses a single vertical line to sense activities in presynaptic neurons using constant and equal resistance values.

After sensing the input activation of presynaptic neurons, the sensing circuit disconnects the postsynaptic neurons’ inputs from presynaptic neurons and switches on the inverted output signal \( V_y \) to overwrite the input signals (see Fig. 5). Switching of the neurons’ input and sending the feedback signal must be done long enough to provide conditions for neural network training and yet short enough not to block the subsequent neuron firings. Due to the disconnection of postsynaptic neurons’ inputs from presynaptic neurons, only the postsynaptic neurons that were activated above a threshold \( V_{\text{th}} \) while others do not. Hence no explicit inhibition signal is needed to prevent firing of other postsynaptic neurons.

Simulations for the training algorithm were performed using the memristor parameters from Table I and HSPICE (BSIM4 v4.3, Level 54) transistor parameters for IBM 90 nm processes provided by MOSIS [33]. Fig. 10 shows simulation results for a sequence of neural network original input activation signals \( V_y \) (original input in Fig. 10). Original input activation signal values changed from 0.60 [V] during the first 4.5 ms, to zero during the resting time of 2.25 ms (from 4.5 ms to 6.75 ms), to 0.40 [V] in subsequent input activation of 4.5 ms (from 6.75 ms to 11.25 ms) and to zero for the remaining 2.25 ms. In Fig. 10 the original input signal \( V_y \) activation values are shown in a dashed line, and \( V_y \) signal modified by feedback is shown in a solid line. Dotted line shows the sensing control signal \( S \).

If \( V_y > V_{\text{th}} \) (we set \( V_{\text{th}} = 0.5 [\text{V}] \)) the neuron fires and its inverted feedback signal \( V_y \) is low. What we see in Fig. 10(a) towards the end of memristors programming period of 4.5 ms is a slight increase in \( V_y \) value due to loss of charge on the input capacitance \( C_{\text{large}} \) in Fig. 5. The rapid increase in \( V_y \) results from
the drop in the input signal value, $V_{\text{in}}$, of inverter $I_{\text{in}}$ in Fig. 6 below its $V_{\text{IL}}$ threshold. This happens towards the end of the memristors programming period at 4.45 ms. During this short time period (when the feedback signal is high) we change the memristor’s value in the direction that is opposite to a desired one. So the circuit time constants must be carefully designed to minimize this effect.

In Fig. 9 when the sensing signal $S$ is high the NMOS transistor is turned ON, the sensing of the input activations is enabled and neurons can sample their input signals $V_x$, (see Fig. 6). The input activations charge capacitor $C_{a0}$ thus increasing $V_{a0}$, the input voltage of inverter $I_{\text{in}}$. When $V_{a0}$ crosses threshold of logical “1” ($V_{\text{IH}}$) the inverter $I_{\text{in}}$ drives the sensing signal $S$ to low and thus turning OFF the NMOS transistor and disabling input sensing. With the inputs disconnected capacitor $C_{a0}$ starts discharging and when voltage $V_{a0}$ falls below $V_{\text{IL}}$ of inverter $I_{\text{in}}$ the sensing signal $S$ goes high and the input sensing is again enabled. These changes can be observed in Fig. 10(a). The input signals used have a rise and fall time of 10 $\mu$s. At 0 ms the input signal is enabled and reaches its peaks at 10 $\mu$s. Around 15 $\mu$s the input activation is detected and the sensing signal $S$ falls below $V_{\text{IL}}$ of the NMOS transistor (below 0.5 V) turning it off. This in turn enables the feedback signal, $V_y$, which goes low for the first input signal equal to 0.60 [V] and the feedback signal $V_y$ is high at about 6.77 ms for the input signal 0.40 [V] which indicates correct operation.

We can also see that after firing period neuron rests from 4.5 ms to 6.75 ms and from 11.25 ms to 13.5 ms. The input signal starts falling at 4.5 ms and reaches its resting potential of 0 V at 4.51 ms. The RC time constant of the sensing circuit consisting of $C_{a0}$ and $R_{a0}$ was designed to have a time period of 4.6 ms as and as the capacitor discharges the voltage $V_{a0}$ falls below $V_{\text{IL}}$ of inverter $I_{\text{in}}$ and the sensing signal $S$ is driven high at about 4.6 ms consequently opening the transmission gate and disconnecting the feedback signal. The RC time constant for the sensed input signal $V_{\text{y}}$ is designed such that this signal will discharge in less than 4.6 ms thus forcing the feedback high at about 4.45 ms and remains high till sensing signal $S$ is driven high around 4.6 ms. This is shown in Fig. 10(b).

Thus for a brief period of time at the beginning and the end of each training cycle, feedback signal $V_y$ may go to an undesirable value. In our simulations it was observed that the duration of this undesirable weight change pulse was in the range of 0.12 ms to 0.15 ms, i.e., the duration of undesirable weight change pulse was about 30 times shorter than the duration of the desired weight change pulse (4.5 ms). Since the duration of time when the training voltage (and the resulting flux) across memristors is correct greatly exceeds the time when the flux direction is opposite to a desired one, the overall conductance change will be correct and perturbation that results from undesired flux is small. Our simulation demonstrated that memristor weights change in the correct direction following the neural network training.

Fig. 11 shows simulation results for the sequence of neuron’s input activations on the level described by the following sequence:

$$V_y(t) = [0.9, 0.2, 0, 0.09, 0.8, 0.7, 0.3, 0, 0.9, 0.4, 0.1] [V]$$

$$[14]$$

In this simulation the firing time $T_f$ was set to 4.5 ms and resting time was 2.25 ms. In Fig. 11(a) the original input is shown using a dashed line and the neuron’s output signal using the solid line. Likewise, in Fig. 11(b) the original input is shown using a dashed line and the feedback signal using a solid line.

### B. Biasing of the Sensing Network

The sensing circuit is critical for correct implementation and testing described in the neural training algorithm. In the sensing circuit shown in Fig. 9, outputs from all presynaptic neurons were connected to the input of the sensing circuit through 1 kΩ resistors that act as voltage summers. The input voltage $V_{\text{in}}$ of the sensing circuit depends on the number of currently active presynaptic neurons. The sensing circuit should be able to detect the activation of even a single presynaptic neuron. Thus as the number of presynaptic neurons increases, the minimum $V_{\text{in}}$ of the sensing network should detect decreases. As very low signal values would be unable to trigger the inverter $I_{\text{in}}$ in Fig. 9, it is necessary to amplify $V_{\text{in}}$. We accomplished this by using a differential pair amplifier with current mirror load, using the circuit shown in Fig. 11. The sensing circuit input has a DC bias signal of $V_{\text{bias}} = 0.33$ [V]. The outputs from the presynaptic neurons were applied to $v_{G1}$ input of the differential amplifier in Fig. 12 such that

$$v_{G1} = \frac{V_{\text{DD}}G_B + \sum_{i=1}^{n} V_{\text{in}}G_{in}}{G_B + \sum_{i=1}^{n} G_{in}} \quad [15]$$

where $n$ is the number of presynaptic neurons and all $G_{in} = 1$ mS. $v_{G2}$ input is obtained in a very similar way as

$$v_{G2} = \frac{V_{\text{DD}}G_B}{G_B + \sum_{i=1}^{n} G_{in}} \quad [16]$$
In this circuit, activation of a single presynaptic neuron gives rise to the input voltage of the differential pair amplifier equal to

$$v_{G1} - v_{G2} = \frac{V_{in}G_{in}}{G_B + \sum_{i=1}^{n} G_{1n}}$$

(17)

The amplifier gain must be set to detect this voltage difference. Depending on the number of presynaptic neurons we can calculate the required value of $G_B$ to obtain the required DC offset voltage form

$$\frac{V_{DD}G_B}{G_H + \sum_{i=1}^{n} G_{1n}} = 0.33 [V]$$

(18)

so

$$G_B = \frac{\sum_{i=1}^{n} G_{1n}}{0.33} - 1$$

(19)

The output $V_{S1}$ amplifies $v_{G1} - v_{G2}$ difference, so $V_{S1}$ is high if there was activation on the sensing circuit input or low in case of no activity. Subsequently, $V_{S1}$ is used to trigger the $In_2$ inverter at the input of the sensing circuit (Fig. 9).

All transistors were of minimum sizes with $W/L = 120 \text{ nm}/100 \text{ nm}$ and $W/L = 240 \text{ nm}/100 \text{ nm}$ for NMOS and PMOS respectively. The capacitors and resistors were sized to provide a RC time constant of 4.6 ms for the sensing circuit and 4.45 ms for the input of the postsynaptic neurons.

V. REAL-TIME TESTING AND WEIGHT ADJUSTMENT

We will illustrate changes in memristor values and development of self-organizing networks using MNIST data base \[34\]. Our aim is to explore the properties of memristive neural networks and their training rather than top recognition performance, so this data base was selected for illustrative purpose only. MNIST data base contains two datasets—training dataset with 60 000 images and testing dataset with 10 000 images. Each digit image has $28 \times 28 = 784$ pixels with gray scale pixel intensities from 0 to 255. We scale them to [0-1] interval for the purpose of memristor training. Since this data base represents 10 classes, we use a feedforward neural network with 784 input neurons and number of output neurons greater or equal to 10.

Each output neuron is initialized to represent one randomly selected training data. Thus normalized memristor values that correspond to this selected training data are set between 0 and 1. With these initial conditions the input signals that have rise and fall times of $10 \mu s$, ON time of 4.5 ms and OFF time of 2.25 ms are applied. We use a semisupervised training in which input neurons adjust their weights based on the training signal values generated by the input signals and feedback signals provided by the output neurons. Then, the memristor values change depending on the training regime.

Let us first illustrate memristor weight changes. Fig. 13 shows an example of the training data that was selected for one of the output neurons. Subsequently, 2600 training data points were applied one by one to the neural network input each one with ON time of 4.5 ms and OFF time of 2.25 ms and changes of all memristor values were recorded.

Changes in the memristance conductance values in winner-takes-all learning are shown in Fig. 14 for pixels in row number 17 of digit 6. Memristors represent pixel values in columns 8, 9, 10, 11, 14, 17. As we can see, the adjustments in this output neuron’s memristor values correspond to the training samples nearest to the weights of this neuron.

Memristor changes are infrequent, since only the winning output neuron adjusts its input weights. Once training, with 2600 data points, was completed and memristor values were adjusted we applied test set of 10 000 data points from MNIST database. With 300 output neurons the average correct recognition level, after 25 runs with random selection of the output seed neurons, was 80.2% with a standard deviation 0.8%.

Changes in the memristance conductance values in winner-takes-all learning are shown in Fig. 14 for pixels in row number 17 of digit 6. Memristors represent pixel values in columns 8, 9, 10, 11, 14, 17. As we can see, the adjustments in this output neuron’s memristor values correspond to the training samples nearest to the weights of this neuron.

Memristor changes are infrequent, since only the winning output neuron adjusts its input weights. Once training, with 2600 data points, was completed and memristor values were adjusted we applied test set of 10 000 data points from MNIST database. With 300 output neurons the average correct recognition level, after 25 runs with random selection of the output seed neurons, was 80.2% with a standard deviation 0.8%.

A single layer feed-forward neural network using memristive devices in crossbar architecture and simulated at system level using C++ code has been shown to reach 93.5% recognition with 300 output neurons \[29\]. This network was using spiking...
neurons and synaptic weights and neuron firing thresholds were individually adjusted in such way that each neuron fired at pre-specified firing rate. Control structures for individual neuron’s threshold adjustment to achieve pre-specified firing rates necessitates complex circuitry and was neither shown nor discussed in the paper. A traditional neural network implementation with 300 hidden neurons and back propagation reaches 95% [35], but implementation of back propagation in hardware is difficult and requires complex control circuit for adjusting memristor weights.

In both cases the classification results were obtained by training the networks with the complete training set of 60 000 data points repeated three [29] and 20 times [35] respectively (vs 2600 data points and no repetitions in our work). Even higher recognition rates were reported using deep neural networks [36]. Recognition rate exceeding 99.5% was obtained on the augmented MNIST data base using 5 layers neural network with 2500, 2000, 1500, 1000, 500 hidden neurons respectively, with total 12 107 500 interconnection weights [36]. Thus the recognition rate is clearly a function of the neural network structure and depends on the number of training data used.

Thus, performance obtained in our work while clearly modest has been achieved using a very limited training set and without use of lateral inhibition, backpropagation or multilayer structure. In addition, the simulations were performed using HSPICE, accounting for device physics of all circuit elements/components. The emphasis in our work was on simplicity of memristor learning rules, on-line weight adjustments, crossbar network architecture and supporting control circuitry. In our work the network was trained with only 2600 data points instead of the complete training set of 60 000 data points since HSPICE simulation time for this circuit was very long. It took us about 23 hours on an Intel Xeon X5650 CPU.

To test robustness of the memristor based neural network we corrupted test data with uniform noise signal from 5% to 2500% of the original testing image values. Fig. 15 shows a sample of test signal and its noisy image for the noise to signal ratio of 200%. Test was performed using a single layer neural network with 784 inputs and with 250 output neurons. Fig. 16 shows the test results. The results show that even when the noise level is 10 times the signal test recognition level is still better than chance (for this data set the chance level is 10%), although a significant drop in recognition is observed when the noise level exceeds 200% of the signal strength and levels of when the noise level is more than 600% of the signal strength. To test the robustness of the circuit the effects of crosstalk and device variations were analyzed.

Absolute changes in the conductivity are not important as long as the ratio in (9) remains unchanged. Critical for neural network performance are local variations within the same die. Local variations within a die are small and correlated, thus their

To analyze the effects of a crosstalk we considered three long parallel wires present in the crossbar grid in our design. Each wire had geometry similar to that from [37], 210 μm long, 100 nm wide and 100 nm thickness, with 784 via’s provides a total resistance of 1712 Ω assuming M1 sheet resistance (106 mΩ/sq) and via resistance (1.9 Ω/via) from [33]. Assuming M1 wire capacitance of 210 pF/m [38] provides an inter-wire capacitance of 44 fF.

To simulate crosstalk conditions, the middle wire received 9 ms pulse with 50% duty-cycle with neighboring wires set to constant high (1 V) and constant low (0 V), respectively. HSPICE simulation showed negligible effect of cross-talk producing two glitches at the beginning and at the end of the signal onset. Duration of each glitch is short (10 ns) compared to 4.5 ms programming cycle. The average flux injected through the crosstalk was also negligible (80 pWb), thus we concluded that crosstalk has no effect on the accuracy of memristor values adjustment. Fig. 17 shows the effect of crosstalk on signal at a neighboring wire, a magnified view of the first glitch is shown in Fig. 17(b).

To test the effect of process variations on the designed neural network learning we performed Monte Carlo analysis changing nominal values of all memristors. While memristors fabricated on different dies will have significant deviations from their nominal values, crossbar memristors are fabricated on the same die and will have memristance parameters tracking each other.

Fig. 17. (a) Voltage-time plot showing effect of crosstalk on the neighboring signal (victim) due to a pulse in the middle (aggressor). (b) Magnified view of the first glitch on (a).

Fig. 15. A sample test image and noisy image with noise to signal ratio equals 200%.

Fig. 16. Correct recognition performance on 10000 test data with various noise level.

Fig. 17. (a) Voltage-time plot showing effect of crosstalk on the neighboring signal (victim) due to a pulse in the middle (aggressor). (b) Magnified view of the first glitch on (a).
The effect on the performance of a neural network for handwritten digit recognition will be also small. To test the robustness of our approach to process variations we considered the effects of variations of memristor device thickness (D) and migration coefficient ($\mu_V$) on recognition rates. Varying the device thickness and migration coefficient causes variations in $R_{ON}$ and $R_{OFF}$ values and flux required to obtain weight adjustment.

In testing the robustness to device variations we performed simulations to account for the worst case (up to 100% tolerance) exceeding the level of the worst case tolerances expected realistically in a mature fabrication technology. We performed 100 tests, each starting from the same training data set (based on 2000 randomly selected digits) but with randomly varied memristor characteristics within specified tolerance. The neural network had 300 output neurons. At each subsequent test the tolerances on memristor parameters, device thickness (D) and migration coefficient ($\mu_V$), were increased from 0.5% until 100% of their nominal values. In each test the same 10 000 test data points were used and recognition performances were recorded. Fig. 18 shows the test results.

We can observe that recognition performance quickly goes down, so the network is more sensitive to variations in memristor parameters that it was to the input noise. However, since typically, within the same die, the memristances do not vary more than 5% (a reasonable assumption for medium size dies with present CMOS manufacturing capabilities [39], and a reasonable assumption for memristive fabrication as technology matures), the loss of the recognition performance is small. The random variation of memristive device parameters can, sometimes, cause the effects to cancel out; the jagged edges in Fig. 19 result from this.

VI. CONCLUSION

In this paper, we introduced new neural network architecture with memristors placed in a tight crossbar structure that preserves high density of synaptic connections. We presented memristor training scheme in the crossbar organization such that neural network interconnection weights can be implemented and controlled during the network operation. This is obtained without separate adjustment of individual memristor values.

We also presented design and simulation of neural network circuits that include sensing circuit and winner-takes-all control. Sensing circuit controls feedback signals of postsynaptic neurons. Using proper timing of the feedback signals and presynaptic neuron output signals we can automatically obtain desired changes of memristor resistances during the network operation. Only a single such sensing circuit is required for the whole crossbar architecture that implements synaptic connection scheme between two layers of a self-organizing neural network.

We performed HSPICE simulations to test the proposed organization of neural network on noisy digit recognition. We showed results of HSPICE simulation and presented analysis of resulting signal control for adjusting memristor weights. The results obtained verified correct adjustment of memristor values for the selected training data. We performed crosstalk test and Monte Carlo analysis of memristor values to test robustness of the proposed architecture in neural network training.

In this work the focus was on developing correct training scheme for crossbar architecture, rather than on specific data bases or advanced neural network application. Hence the networks were tested on MNIST data base only to confirm the validity of this neural network organization. The proposed solution improves available in the literature architecture and training methods for memristive neural networks.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Dr. Savas Kaya from Ohio University for providing access to the HSPICE circuit simulator and helpful suggestions regarding simulations; and to graduate student Soumyasanta Laha for his help in running the simulator.

REFERENCES

12 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS


Janusz A. Starzyk (SM’83) received the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the Warsaw University of Technology, Warsaw, Poland, and the Habilitation degree in electrical engineering from the Silesian University of Technology, Gliwice, Poland.

He was an Assistant Professor with the Institute of Electronics Fundamentals, Warsaw University of Technology. He spent two years as a Postdoctoral Fellow and a Research Engineer with McMaster University, Hamilton, ON, Canada. Since 1991, he has been a Professor of Electrical Engineering and Computer Science with Ohio University, Athens, OH, USA, and the Director of Embodied Intelligence Laboratories. Since 2007, he has been the Head of the Information Systems and Technology, Gaithersburg, MD, USA, in the area of testing and mixed signal fault diagnosis for eight years. His current research interests include embodied machine intelligence, motivated goal-driven learning, self-organizing associative spatiotemporal memories, active learning of sensory-motor interactions, machine consciousness, and applications of machine learning to autonomous robots and avatars.

Basawaraj received the B.E. degree in electronics and communication engineering and the M.Tech. degree in digital electronics and communication. He is currently pursuing the Ph.D. degree in the School of Electrical Engineering and Computer Science at Ohio University, Athens, OH, USA. His current research interests include spatiotemporal memories for embodied machine intelligence.