A CMOS-Memristive Self-Learning Neural Network for Pattern Classification Applications

Melika Payvand, Justin Rofeh, Avantika Sodhi, Luke Theogarajan
University of California Santa Barbara
Santa Barbara, CA, U.S.A
ltheogar@ece.ucsb.edu

Abstract— **Memristors have proven to be powerful analogs of neural synapses. While there have been some efforts to exploit this feature, the intrinsic analog nature of the memristive element has not been fully utilized. This paper presents a hardware-efficient neuromorphic CMOS-memristor pattern classifier. The system takes advantage of the memristor as a true analog memory, and Spike Timing Dependent Plasticity (STDP) is utilized to program memristors in a recurrent neural network. System co-simulations are performed in Verilog-AMS with CMOS devices and previously published memristive models. The results indicate the power of this approach in pattern classification using unsupervised learning.**

Keywords— ***Neural networks; Unsupervised learning; Adaptive learning; VLSI learning circuits; Spike Timing Dependent Plasticity (STDP); Memristors***

I. Introduction

Realizing a computational framework rivaling the scale and complexity of biological neural networks is extremely challenging. This stems from the enormous connectivity in the brain utilized for massive parallel processing, which is further exacerbated with the need for a nonvolatile memory or synthetic synapse. Advances in nonvolatile storage such as Flash memories, though exciting, are extremely hard to co-integrate in a dense fashion with CMOS. Recently, memristors have emerged as a dense nonvolatile memory [1], capable of being directly integrated with CMOS in a hybrid fashion [2,3].

Lately, several attempts have been made to synthesize neuromorphic computational platforms utilizing memristors and CMOS circuits [5]-[11]. However, they either need synchronization which is not a biologically plausible mechanism [4,5], require a global and/or local teacher [6], are computationally expensive [4-6, 9] or depend on absolute spike shape leading to variable learning rates [10-11]. Moreover, most of these examples primarily utilize the digital design framework, which does not exploit the memristor's potential as a true analog memory with a continuum of programming levels. Furthermore, none of these examples have used memristors in neural networks for pattern recognition or clustering applications.

We present here a hardware-efficient neuromorphic platform utilizing principles borrowed from computational neuroscience and biology. The principal idea is the efficient implementation of spike-based learning algorithm utilizing a biomimetic circuit approach. Using this principle we have realized simple pattern recognition architectures.

The paper is organized as follows: In section II, we present a brief background on the learning algorithm and the memristor model used in this work. Section III describes the proposed architecture for the spike-based unsupervised neural network. Details of the circuit-level implementation are discussed in section IV with the simulation results provided in section V. Finally, concluding remarks about the potential of such a platform is provided in section VI.

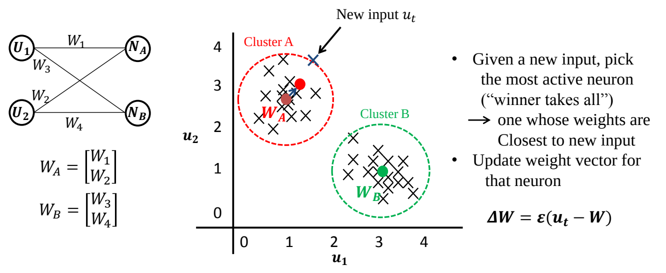
II. Background

A. Unsupervised Learning

In unsupervised neural networks the clustering or classification takes place based on the input patterns given to the network. As shown in Fig. 1, each output neuron represents a cluster. Therefore, the weight vector $W_A(W_B)$ is the center of cluster A(B). When a new input is presented to the network, an output neuron (or compute node) emerges as a winner due to the random initial state of the weights in the network. The most active output neuron will be the one whose weight vector is the closest to the input vector, u_i. The learning algorithm minimizes the distance between the winning neuron's weight vector and the input pattern. Over the course of many patterns, features get mapped into the weight vectors. In other words weights are the running average of the input patterns:

$$\begin{aligned} W_A(t) &= (\sum_{i=1}^{t} u_i)\frac{1}{t} = (\sum_{i=1}^{t-1} u_i + u_t)\frac{1}{t} \\ &= \frac{t-1}{t} * W_A(t-1) + \frac{u_t}{t} \\ &= W_A(t-1) + \frac{1}{t}(u_t - W_A(t-1)) \\ &\rightarrow \Delta W = \varepsilon(u_t - W) \end{aligned} \tag{1}$$

Where ε is the learning rate [12].



New input u_t
Cluster A
Cluster B
W_A
W_B
u_1
u_2
$W_A = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix}$
$W_B = \begin{bmatrix} W_3 \\ W_4 \end{bmatrix}$
Given a new input, pick the most active neuron ("winner takes all")
one whose weights are Closest to new input
Update weight vector for that neuron
$\Delta W = \varepsilon(u_t - W)$

Fig. 1. Competitive Learning- Each output neuron represents a cluster. N_A and N_B represent cluster A and B respectively and W_A and W_B are the centers of the clusters. Upon the arrival of every input pattern, the winner neuron's weights adjust in a way to get closer to the input pattern. u_1 and u_2 are the firing rates of input neurons U_1 and U_2. Adapted from [12].

B. *Spike Timing Dependent Pasticity*

Spike Timing Dependent Plasticity (STDP) is a biological learning mechanism found to exist in the brain [Bi and Poo, 2001]. The synaptic strength changes as a result of relative timing of spikes between the pre- and postsynaptic neurons. The weights undergo Long Term Potentiation (LTP) and strengthen if the pre- and postsynaptic neurons both depolarize and fire simultaneously. Conversely, the connection between neurons weaken when there is uncorrelated firing between the postsynaptic and presynaptic neuron, termed Long Term Depression (LTD).

Previous works have demonstrated memristors as a good candidate for mimicking this learning mechanism (STDP) in the brain. [14]

Our approach utilizes the mathematical modeling of STDP from computational neuroscience coupled with the structural biological framework of integrate and fire neurons and memristive synapses to design an adaptive neural network.

C. *Memristor Model*

The memristor model used in this work is based on the Linear Ion Drift (LID) model proposed by Strukov *et al.* [1]. Simulation modeling of memristive behavior was performed using Verilog-AMS code modified from the simulation model (LID option) presented by Kvatinsky *et al.* [15].

This model assumes two resistors in series: R_{on} and R_{off}. R_{on} and R_{off} represent the doped and un-doped regions of the active area of the device. Fig. 2 shows the simplified memristor model and the inset represents the memristor's symbol. The state variable *w* is bounded between 0 and D (the length of the active region) and moves between the doped and un-doped region under the application of the electric field and changes the resistance of the device.

The so-called memristance of the device follows equation 2:

$$\boldsymbol{M(w) = \frac{w}{D} R_{\text{on}} + \left(1 - \frac{w}{D}\right) R_{\text{off}}} \qquad (2)$$

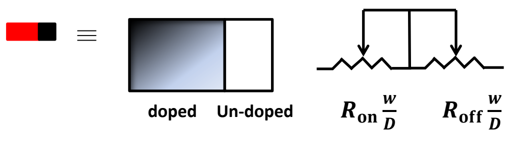


Fig. 2. Linear Ion Drift model of the memristors proposed in [1].

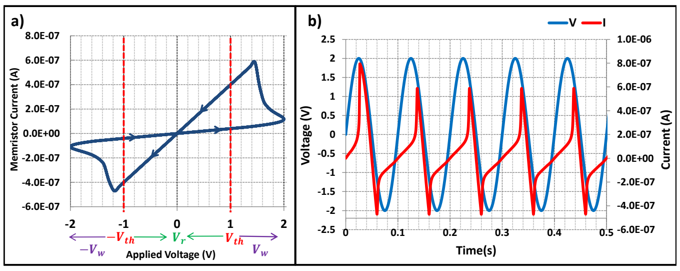


Fig. 3. Memristor I-V characteristics modeled in Verilog AMS and simulated in Cadence Spectre. a. current-voltage and b. time dependent characteristics of the memristor model.

The I-V characteristic of the memristor modeled in Verilog AMS and simulated in Cadence Spectre is shown in Fig. 3. Below the threshold $|V_{\text{th}}|$, the change in the state of the device is negligible, which is desirable for reading ($V_{\text{read}} = V_{\text{r}}$) since the memristor acts as a resistance in this region. As the voltage across the device increases above $V_{\text{th}}(-V_{\text{th}})$, the device state changes, increasing (decreasing) the conductance. This region is suitable for writing on the device ($V_{\text{write}} = V_{\text{w}}$). Table 1 shows the values chosen for the memristor model in this work.

TABLE1. MEMRISTOR CHARACTERISTICS IN THIS WORK

R_{ON}	R_{off}	D	μ_v	V_{th}
On Resistance	Off Resistance	Device's Length	Dopant Mobility	Device's Threshold
2.5 MΩ	25 MΩ	10 nm	1E-13 m^2/Vs	1 V

III. NEURAL NETWORK ARCHITECTURE

We present here a neural network architecture in which memristors' weights follow the input patterns presented to the network in an unsupervised fashion. Dense implementation of the network can be achieved using crossbar arrays integrated on the CMOS chip [3,4]. Preliminary efforts in integrating CMOS die with memristors have been performed by us, demonstrating the feasibility of the approach. The microscopic image in Fig. 4a shows an example of a CMOS chip designed in our group before and after the integration of memristors.

A. *Crossbar Array*

Fig. 4b shows the overall architecture of the neural network in a crossbar configuration. Blue and green circles represent pre- and postsynaptic CMOS neurons respectively. The smaller red circles describe the memristive synapses between the junction of the pre- and postsynaptic neurons.

Input patterns are fed to the presynaptic neurons, while the output is read from the postsynaptic neuron. Relative spiking of the pre- and postsynaptic neurons changes the corresponding synaptic weight between them. The details of the architecture for a small network are explained below.

B. *Simple Case: 2X2 Network*

Fig. 4c represents the block diagram of a self-learning 2X2 network. When the input image is received, using an on-chip photodiode array, the current output from the photodiode is given to an integrate and fire block (whose input is pinned to V_{cm}) and presynaptic neurons start spiking. The height of

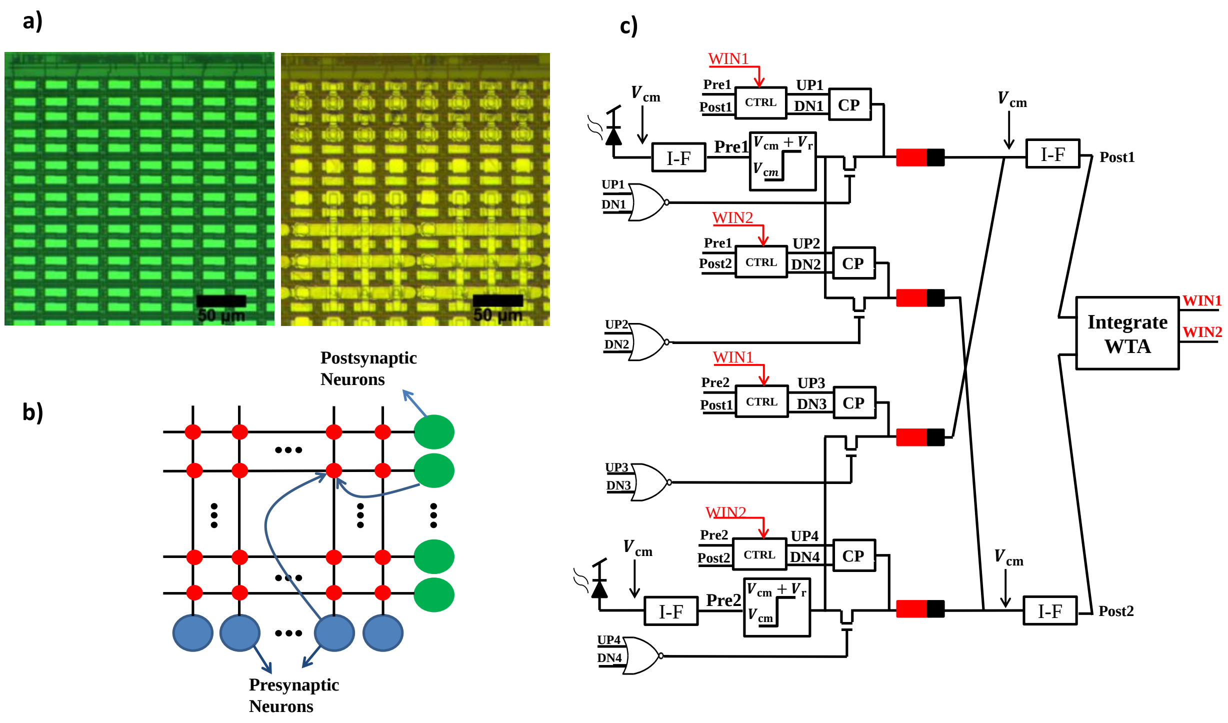


Fig. 4. Overall architecture of the neural network used in this work. CMOS neurons are connected via the memristive synapses in a crossbar configuration. a) Preliminary efforts in integrating memristors with a CMOS die. On the left is the CMOS chip before the integration and the image on the right shows single devices along with crossbars fabricated on top of the same CMOS chip. The scale bars are 50 µm. b) Pre- and postsynaptic neuron in a crossbar array. Relative timing of the pre- and postsynaptic spikes decides the connection strength between them. c) The architecture of a self-learning 2X2 network proposed in this work.

the spike is level shifted to V_{cm} when low, and to $V_{cm} + V_r$ when high, so that in normal operation the voltage across the memristors is V_r and therefore, the state of the memristor does not change (Fig 3). In contrast, the spiking behavior of the postsynaptic neurons depends on the initial condition of the memristors. The responses of the postsynaptic neurons are compared using an integrate-winner-take-all circuit (section IV) and the winner decides which weights should be changed.
The STDP CTRL block gets inputs from pre- and postsynaptic neurons and generates an UP signal if the post and the pre are spiking simultaneously (LTP). If postsynaptic neuron is firing while presynaptic is not, the CTRL block creates a DOWN (DN) signal (LTD).
The Charge Pump (CP) receives the UP and DN signals from the CTRL block and charges or discharges the left (positive) side of the memristor to beyond $V_{\mathrm{cm}} + V_{\mathrm{th}}$ and $V_{\mathrm{cm}} - V_{\mathrm{th}}$ respectively. This will result in voltages higher than $\pm V_{\mathrm{th}}$ appearing across the device. Hence, the memristor's state (stored as weights) increases or decreases accordingly. The details of the circuits are discussed in section IV.

IV. VLSI LEARNING CIRCUIT

The circuits are designed to locally change the weights in an adaptive fashion using a feedback loop. The details of each block are explained below.

A. Integrate and Fire Model for Neurons

Fig. 5 shows the conventional integrate and fire neuron model. In this model, the input current gets integrated by a capacitor and when the integrated charge results in a voltage greater

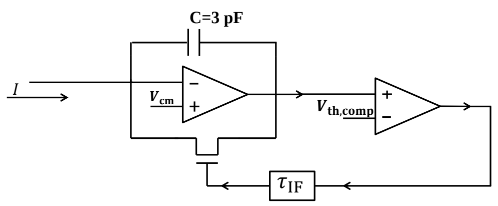


Fig. 5. Integrate and fire model of the neuron. Current gets integrated into the capacitor C. The comparator is triggered when the integrated charge in the capacitor results in a voltage greater than $V_{\mathrm{th,comp}}$. The capacitor is reset after a τ_{IF} delay.

than $V_{\mathrm{th,comp}}$, a spike with a pulse width of τ_{IF} is generated.

B. STDP Control Block

This block gets inputs from the pre- and postsynaptic neurons and decides if the corresponding weight between them should increase or decrease. The module is designed to mimic the way the synapse's strength changes in the brain, namely imitating LTP and LTD.

a. STDP UP Control

In order to ensure that the simultaneous firing of the pre- and postsynaptic neurons is captured, the presynaptic spike is first latched for a period of τ_{UP}[1] and then ANDed with the post-synaptic spike (Fig 6). The latch will reset after a τ_{UP} delay using a pulse generator and waits for the next Pre spike.

[1] $\tau_{\mathrm{UP}} \geq T_{\mathrm{Post,max}}$. $T_{\mathrm{Post,max}}$ is the maximum period of the post spikes.

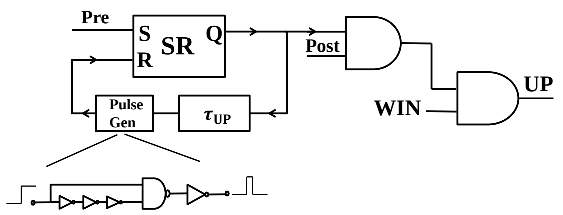


Fig. 6. STDP UP CTRL Block. Every time the Presynaptic neuron (Pre) spikes, it is latched and ANDed with the Postsynaptic (Post) spikes. If the corresponding postsynaptic neuron (output neuron) is winning, an UP signal will be generated.

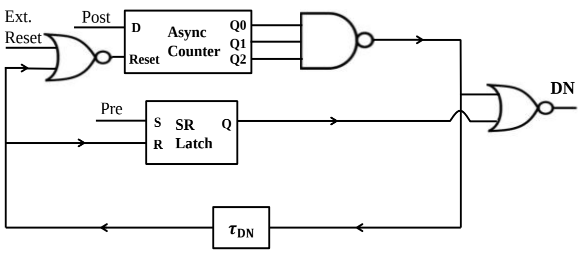


Fig. 7. STDP DOWN CTRL block. When Post spikes 8 times without any spike from Pre, a DN signal with a pulse width of τ_{DN} gets generated. Ext. Reset is the chip reset signal.

An UP signal will be produced if the postsynaptic neuron is also the winning neuron for the current pattern.

b. STDP DOWN Control

A DOWN (DN) signal is generated given that postsynaptic neuron (Post) is firing while presynaptic neuron (Pre) is not firing. Fig. 7 shows the circuit diagram of STDP DOWN block.
As a time-out mechanism, Post spikes are counted through a 3-bit counter. Since spike durations are shorter than the time-out window, Pre signal is latched. If there is no Pre spike while the counter output reaches 7 (time-out condition), a DN signal is generated. The counter is reset every time it reaches the maximum value and it waits for the next Post spike.

C. Charge Pump

Fig. 8 presents the circuit schematic of the charge pump block. The positive side of the memristor (Node VCP) is either controlled by the input coming from the switch M_1 or by the charge pump. In normal operation, when there is no signal from the CTRL block, M_2 and M_3 are open and the charge pump is bypassed. When M_1 is closed the spikes from the presynaptic neuron will pass through the memristor and reach the postsynaptic neuron without changing the weight.
However, under the condition in which the charge pump receives the UP or DN pulses from the STDP control block, switch M_1 will open and the charge pump will take over the node VCP and charges or discharges it accordingly (Fig. 9).
It is desirable to pre-charge the charge pump capacitor to a value just below the memristor's threshold to eliminate dead-time. However, since the pre-charge voltage is different for increasing or decreasing the memristor state, the charge pump is split into UP and DOWN halves. For example, the charge pump capacitors are pre-charged to $V_{\mathrm{cm}} \pm V_{\mathrm{th}}$ so that when

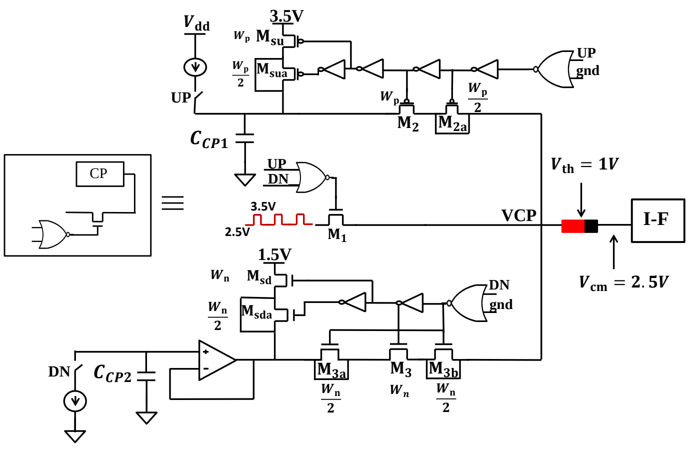


Fig. 8. Charge pump used to change the state of the memristor. C_{CP1} is pre-charged to 3.5V which is just at the edge of changing the device's state. Similarly C_{CP2} is pre-charged to 1.5V. The delays for disconnecting the charge pump capacitor from the sources are set so as to reduce the capacitor drooping as a result of loading.

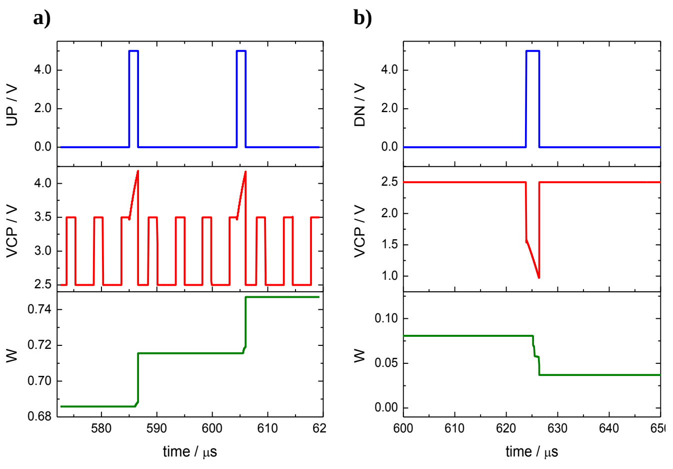


Fig. 9. a) Upon the arrival of the UP signal node VCP starts charging and when it hits the device's threshold, the memristor's weight increases. b) DN signal causes node VCP to discharge and the device's state decreases.

the control pulses are received, the memristor's weight starts changing immediately. In this design, $V_{\mathrm{th}} = 1$ V and $V_{\mathrm{cm}} =$ 2.5 V, thus C_{CP1} and C_{CP2} are pre-charged to 3.5 V and 1.5 V respectively. The voltage on the capacitors can drop due to timing overlap between the closing of M_2(M_3) and the opening of M_1. To prevent this, the signals to M_2(M_3) and M_{su}(M_{sd}) are delayed via an inverter chain and a NOR gate configured as an inverter (used for matching purposes). M_{2a}, M_{3a}, M_{3b}, M_{sua} and M_{sda} are half-sized transistors in order to reduce the switching noise when M_2, M_3, M_{su} and M_{sd} turn on and off.

D. Integrate Winner-Take-All

The winning output neuron decides which connections are to be increased. Therefore, a circuit determining which output neuron is the winner is needed.

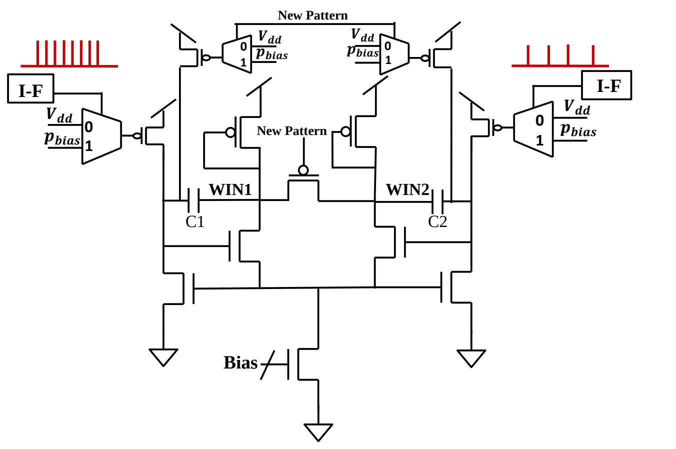


Fig. 10. Integrator WTA. The spikes are integrated and fed to a WTA circuit. With the arrival of each new pattern, the WTA gets reset and waits for the new spikes.

Since we are dealing with spikes, the most intuitive way would be to count and compare them. However, as the number of the output neurons increases, the hardware-complexity will proportionally increase. An elegant alternative would be to use analog design; i.e. integrate the spikes, and use a winner-take-all (WTA) circuit to decide the winner. Fig. 10 shows the details of the circuit combining the classical WTA and a conventional integrator.

V. RESULTS

A. 2X2 Network

In a simple 2X2 network, the goal is to classify between the 2-pixel images from Fig. 11b,c.

To feed the input patterns to the network, P1 and P2 from Fig. 11a are given as inputs to U_A and U_B respectively. Let us assume that image from Fig. 11*b* gets randomly (initial condition) assigned to output neuron N_A. This should result in strengthening the connection between U_A and N_A because of the simultaneous spiking of N_A and U_A (U_A is spiking since the black pixel is given to its input). However, in this case, image b should not trigger output neuron N_B, thus the weight between U_A and N_B should decrease.

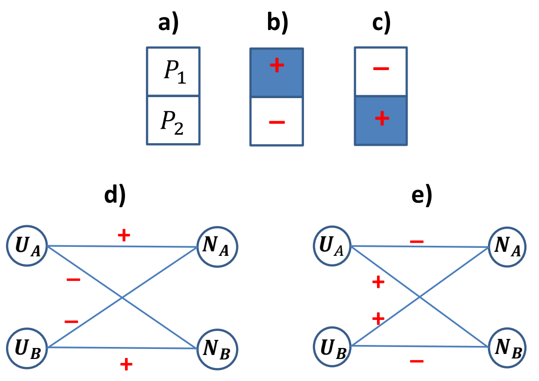


Fig. 11. 2X2 network. a) pixel P1 is fed to U_A and P2 is fed to U_B. b and c) Images to be classified. d and e) Desirable weights depending on the pattern which gets assigned to different output neurons

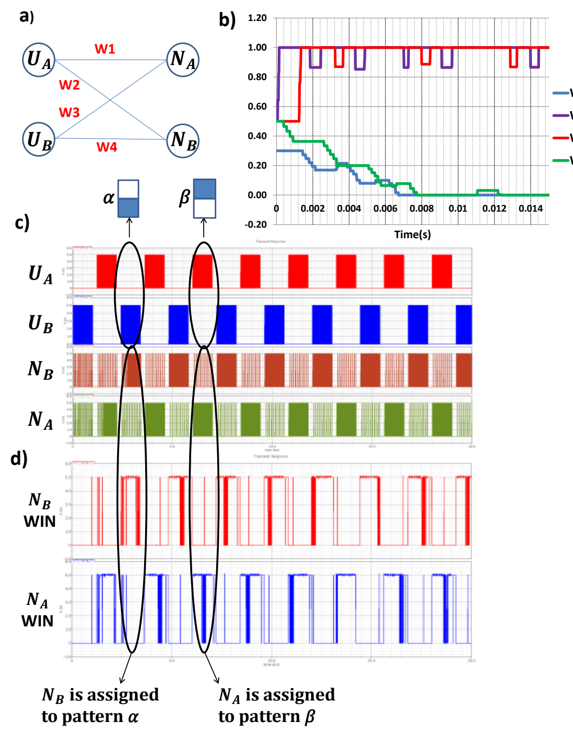


Fig. 12. 2X2 network classification results. a) 2X2 neural network and corresponding weights between neurons b) Weight evolution. Weights converge to the expected values from Fig. 11e. c) Input and output neuron's spikes. Patterns α and β are assigned to neurons N_B and N_A respectively. The winning neuron for the given input pattern fires the most. d) The outputs of the WTA. N_A WIN and N_B WIN are high (5V) for input pattern β and α, correspondingly.

A similar reasoning can also be applied to image in Fig. 11c. Therefore, the weights should converge to Fig. 11d or 11e depending on which input image gets assigned to which output neuron.

As seen from Fig. 12 the results of the classification corresponds to the expected behavior from Fig. 11e. In Fig. 12b the evolution of weights are shown. W1 and W4 are strengthened and W2 and W3 are weakened. W1 and W4 are dithering around 0.95, because even in the converged state UP and DOWN signals are constantly being generated to prevent the network from being stuck in a local minima. Fig. 12c,d show the assignment of patterns α and β to neurons N_B and N_A respectively.

B. 4X2 Network

A 4X2 network in Fig. 13 classifies the patterns shown in Fig. 13a and 13b. These two patterns enable the classification of lines with different angles, which is the building block for recognizing more complicated patterns.

The weights are expected to converge to Fig. 13c or 13d based on the pattern that gets assigned to different outputs. For example, if pattern in Fig. 13a is assigned to N_A the weight vector of the output neuron N_A converges to [weak, strong, strong, weak]. The results of the classification in the 4X2 network are shown in Fig. 14. Patterns α and β are assigned to output neurons N_B and N_A respectively and thus the weights converge accordingly.

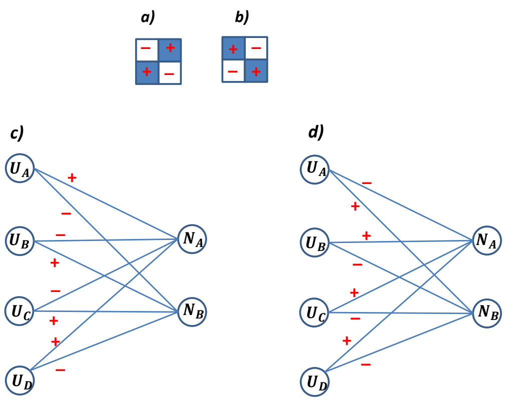


Fig. 13. 4X2 neural network. a and b) patterns to be classified are 2 lines with different angles. c and d) The expected weights to be converged in the neural network in order to classify patterns a and b.

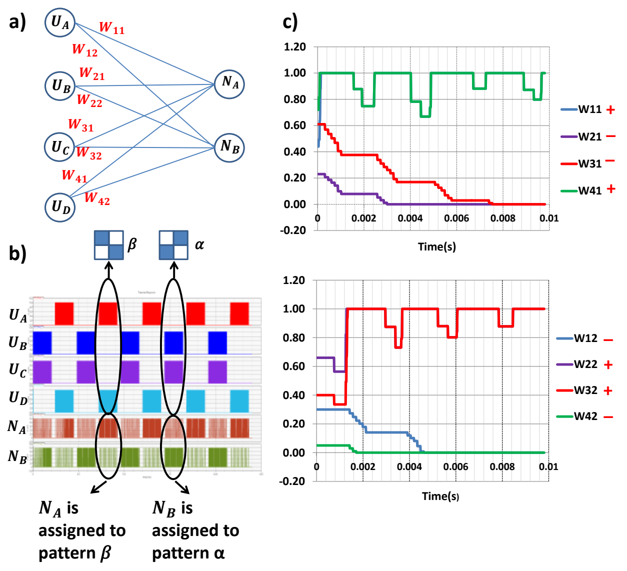


Fig. 14. 4X2 network classification. a) A 4X2 neural network and corresponding weights between neurons. b) Classification between patterns α and β. N_A and N_B fire the most with patterns β and α respectively. c) Weight evolution and convergence to the expected values from fig 13.c.

VI. Conclusion

We have proposed a hardware efficient architecture for classifying patterns in an unsupervised neural network. Spiking neuromorphic circuits are combined with adaptive nano-devices as synapses to design a compact platform for clustering applications. Our implementation utilizes the analog nature of the memristive synapse without resorting to D/A-A/D conversion resulting in an energy efficient implementation. Work is ongoing in our lab to realize an integrated CMOS-memristor chip for fast pattern classification.

Acknowledgment

This work is funded by the Air Force Office of Scientific Research (AFOSR) under the MURI grant FA9550-12-1-0038.

References

[1] D.B. Strukov, G.S. Snider, D.R.Stwart, and R.S.Williams, "Missing memristor found," *Nature*, vol. 453, pp. 80-83, May 2008

[2] K.K. Likharev," Crossnets: Neuromorphic Hybrid CMOS/Nanoelectronic Networks," *Science of Advanced Materials*, vol. 453, pp. 80-83, 2008

[3] D.B. Strukov, K.K. Likharev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices,"*Nanotechnology*, Vol 16, pp. 888-900, March 2005

[4] G.S. Snider, "Spike-Timing-Dependent Learning in Memristive Nanodevices", *Prof. of IEEE International Symposium on Nanoscale Architectures 2008 (NANOARCH), pp. 85-92, 2008*

[5] G.S. Snider, "Self-organized computation with unreliable memristive nanodevices," *Nanotechnol.*, vol. 18, no. 36, 2007.

[6] G.S. Rose, R.Pino, Q. Wu, "A Low-Power Memristive Neuromorphic Circuit Utilizing a Global/Local Training Mechanism", *Proc. of International Joint Conference on Neural Networks 2011*, August 2011

[7] K.D. Cantley, A.Subramaniam, H.J.Stiegler, R.A.Chapman, E.M.Vogel, "Hebbian Learning in Spiking Neural Networks With Nanocrystaline Silicon TFTs and Memristive Synapses", *IEEE Transactions on Nanotechnology* ,Vol. 10, No. 5, September 2011.

[8] I.E.Ebong, P. Mazumder, "CMOS and memristor based neural network design for position detection", *Proc. of IEEE*, Vol. 100. pp. 2050-2060, Jun. 2012

[9] Jose M Cruz-Albrecht, Timothy Derosier and Narayan Srinivasa, "A scalable neural chip with synaptic electronics using CMOS integrated memristors", *Nanotechnology* 24, September 2013

[10] J.A. Pérez-Carrasco, C.Zamarreno-Ramos, T.Serrano-Gotarredona, B.Linares-Barranco,"On neuromorphic spiking architectures for asynchronous STDP memristive systems," *Proc. Of 2010 IEEE Int. Symp. Circuits Systems (ISCAS)*, pp. 1659-1662, 2010.

[11] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis4, G. Indiveri and B. Linares-Barranco," STDP and STDP variations with memristors for spiking neuromorphic learning systems", *Frontiers in neuroscince*,Vol. 7, February 2013

[12] R. P. N. Rao, A. Fairhall (2013, Apr), *Computational Neuroscience*, [Lecture 7.2], Available: https://class.coursera.org/compneuro-001

[13] P.Dyan, L.Abbot, "Theoretical neuroscience: Computational and Mathematical Modeling of Neural Systems", *MIT Press*, 2005.

[14] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B. Bhadviya, Pinaki Mazumder, and Wei Lu, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems", *Nano Letters*, Vol 10, pp. 1297-1301, 2010

[15] S. Kvatinsky, E.G.Friedman, " Models of Memristors for SPICE Simulations ", 2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel, 2012

[16]] Y.Dan, M.Poo, "Spike Timing Dependent Plasticity: From Synapse to Perception", *American Psychological Society*, 2006

[17] W.Gerstner, W.Kistler,Spiking Neuron Models, Single neurons, Populatons, Plasticity, 3rd ed. Cambridge University Press, 2006