2011 11th IEEE International Conference on Nanotechnology
Portland Marriott
August 15-18, 2011, Portland, Oregon, USA

Multi-purpose Neuro-architecture with Memristors

Idongesit Ebong, Durgesh Deshpande, Yalcin Yilmaz, and Pinaki Mazumder
Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109
Email: idong@eecs.umich.edu, durgesh@umich.edu, yalciny@eecs.umich.edu, mazum@eecs.umich.edu

***Abstract* — An analog CMOS neuromorphic design utilizing spike timing dependent plasticity and memristor synapses is investigated for use in building a multi-purpose analog neuromorphic chip. In order to obtain a multi-purpose chip, a suitable architecture is established and several functions with the proposed architecture are shown. Using the IBM 90 nm CMOS9RF process, neurons are designed to interface with Verilog-A memristor synapse models to perform the XOR and Edge Detection functions.**

Index Terms* – *memristors, spike-timing-dependent-plasticity, neuromorphic network

I. Introduction

Chua postulated the concept of a missing element in 1971 [1], but this element was only recognized in 2008 by HP Labs [2]. The memristor has evoked research interest in the advent of limits to CMOS scaling according to Moore's Law. The memristor has been elected as an addition to revolutionize mass connectivity and programmable logic arrays (PLA) [3, 4], digital memory [5], chaotic circuits [6], and neuromorphic networks [7]. The digital memory approach is currently the most promising, since through fabricated arrays it has exposed drawbacks to memristor crossbar arrays, as well as proposed methods to counter these limitations. Knowledge from the memory approach will certainly aid in other memristor applications.

This paper uses the nonlinear analog memristor model for HP Labs' fabricated device in a neuromorphic architecture that is adapted through spike timing dependent plasticity (STDP) [7]. The adaptation provides a conceivable way of demonstrating learning circuits that can stabilize in the presence of noise. Learning is both structurally and environmentally motivated. The proposed neuromorphic structure or architecture highlights the resilience of memristors and provides a fabric to building larger systems. The functions performed by the simulated neuromorphic chip are XOR and edge detection. In addition to these two functions, the realization of a sampled Gaussian kernel for the use of a filter, AND operation, OR operation, and many other functions are possible with the proposed architecture.

The next section (Section II) provides a description of the architecture, how supervised learning is achieved, the neuron block level design and the synapse modeling, and Section III shows the simulation results for the XOR and Edge Detector.

II. Architecture

A. Overall Neuromorphic Architecture

The basic processing of the neuromorphic architecture is shown in Fig. 1. The neurons are shown in circle and implemented in CMOS while synapses are shown with arrowed lines and are implemented with memristors. Excitatory synapses are in red, and inhibitory synapses are in black. The architecture is amenable to STDP synapses whereby the spike timing between pre-neuron and post-neuron determine how much the memristive synapses will adjust. For more detail on STDP implementations with memristors, see [7].

The STDP implementation used here is based on a linear approximation seen in the response of mushroom bodies [8]. The linear approximation is more desirable to memristor design because it allows the use of constant pulse widths to change memristors. The adjustment on how much change the memristor undergoes depends on the number of pulses a memristor sees. This simplifies and reduces neuron size by allowing the use of a common external pulse generator for all neurons in the proposed architecture. This design decision was made to move the spiking behavior outside of the neuron since spike generation with respect to synapse changes consume most of the area in a CMOS design. More details on the spiking behavior are given in the neuron design presented in the next section.

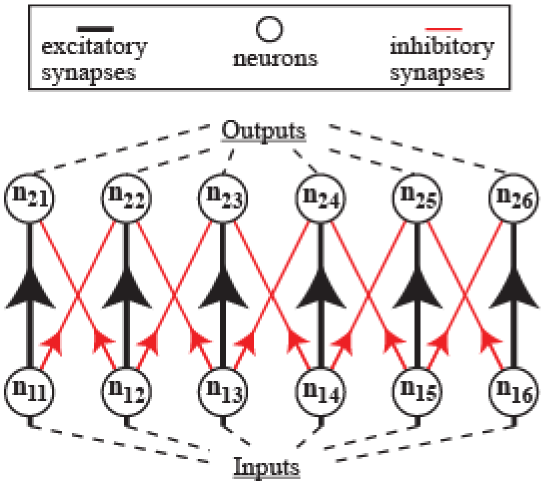

Fig. 1 Neuromorphic architecture showing neurons and synapses.

The authors would like to thank HRL, DARPA, and NSF for partly supporting this work.

The neuromorphic architecture is composed of input neurons and output neurons, and based on the chosen structure of inhibitory to excitatory synapses, various functions can be obtained. The XOR and Edge Detector have the same synaptic weight profile, but a different function, like a Gaussian kernel would have a different profile. The basic architecture shown needs pre- and post-processing circuits to interface with other systems. The post-processing side may contain adders and integrators to convert spiking outputs of the spiking neurons to leveled signals, while the pre-processing side would convert DC level signals to spiking inputs for the neurons. Each function will have different post-processing requirements; the architecture is meant to be barebones to allow for different functionality based on synaptic weight adjustments.

The training process involves using input patterns to adjust memristors to the desired relative values between excitatory and inhibitory synapses. On simulation startup, weights can either be initialized to a low value, or initialized to a random pattern and learned to low values. Using different input patterns, the memristors can be trained to predetermined weights or relative weights between excitatory and inhibitory synapses. For example, starting in a low weight state, $\mathbf{n_{11}}$, $\mathbf{n_{13}}$, and $\mathbf{n_{15}}$ can be made to spike at frequencies that cause $\mathbf{n_{21}}$, $\mathbf{n_{23}}$, and $\mathbf{n_{25}}$ to spike thereby strengthening excitatory synaptic connection between these neurons according to STDP rules. This input pattern will not affect the inhibitory synapses due to the rules of STDP requiring pre-neuron and post-neuron to spike. After these synapses are trained to weights approximately twice the inhibitory synaptic weights, neurons $\mathbf{n_{12}}$, $\mathbf{n_{14}}$, and $\mathbf{n_{16}}$ are used to train the excitatory synapses between $\mathbf{n_{22}}$, $\mathbf{n_{24}}$, and $\mathbf{n_{26}}$.

This training scheme is designed for the XOR and edge detection profile and allows the tuning of excitatory synapses without affecting inhibitory synapses as shown in Fig. 2. The synapse naming follows the convention "pre-neuron_post-neuron." In Fig. 2, the XOR training is done for 30 ms to get a resistance profile for the excitatory neurons around ~5.6 MΩ. The tuning of the memristors to exact resistance values is hard to do therefore in a system, a timer would be used to stop training. This training scheme hints that the neurons have two different modes determined by a control signal deciding on either a training mode or running mode. The difference between the two modes lies in the voltage levels used for both. The training mode uses voltage levels that influence the memristors more than the running mode. The simulation results shown for the XOR and edge detection operation use the learned memristor resistance values of ~10 MΩ and ~5.6 to 6.8 MΩ for the inhibitory synapses and the excitatory synapses, respectively. The simulation results are shown in the run-mode – learning has stabilized, and voltages adjusted so memristors are fairly static.

B. Neuron Design

The neuron design used is based on previous work [9]. The design uses a leaky-integrate-and-fire (LIF) neuron coupled with external pulses to relay information from one layer to another. Fig. 3 provides a block diagram view of the components of the neuron. The integrate and fire circuitry holds a storage node and a comparator that provides a bi-level spike signal whenever the storage node reaches the comparator threshold. As hinted in Fig. 1, the neuron has excitatory inputs that add on to the storage capacitive node, and inhibitory inputs that subtract charge from the storage node. The external pulse is used to determine the spike pattern on the output after a neuron spikes. The spiking

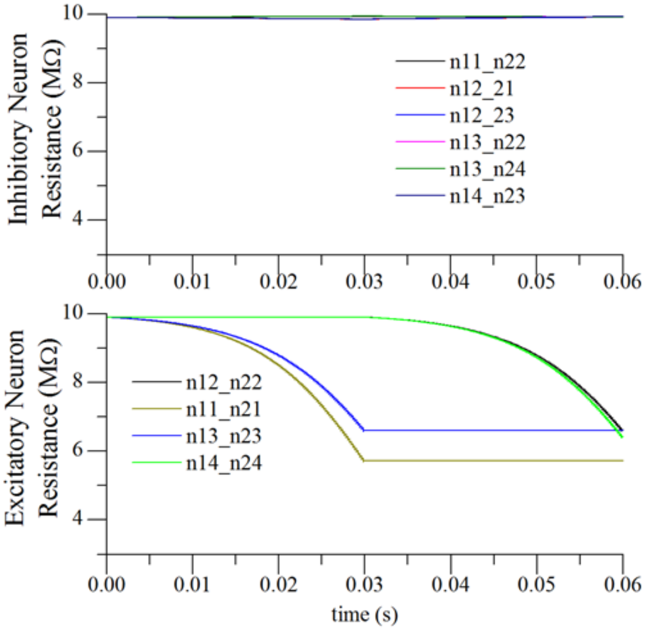

Fig. 2 Training mode using prescribed XOR training scheme
(Top) Inhibitory synapses unchanged during training
(Bottom) Excitatory synapse trained using timestamps

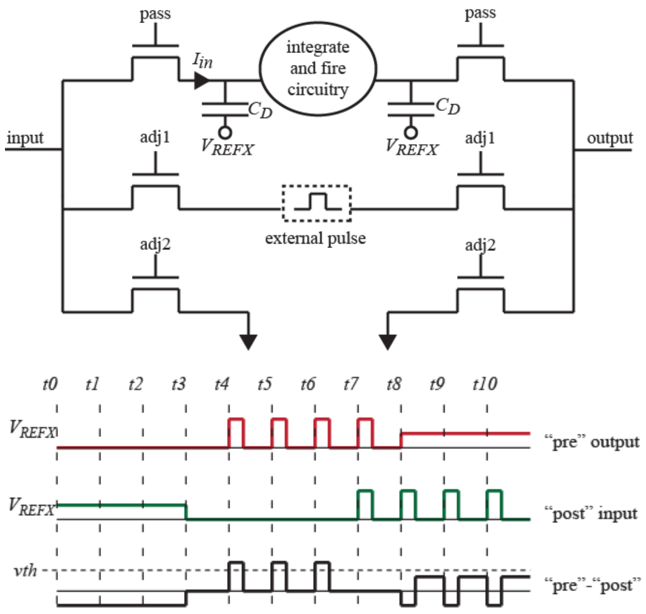

Fig. 3 (Top) Neuron block diagram showing important features
(Bottom) Neuron spiking patterns used to achieve STDP

activity is governed by a finite state machine (FSM) that is not shown.

The spiking scenario is further explicated with the bottom timing diagram in Fig. 3. At the presence of a spike, the spiking neuron first moves both its output and input from a reference voltage (V_{REFX}) to 0V for four clock periods and then emits four pulses that directly bias memristors seeing these pulses before returning to its resting potential of V_{REFX}. The sample spike pattern shown in Fig. 3 hints at the nature of the synapse used in this design. A threshold is designed within the synapse that allows only a voltage difference that exceeds the threshold to register as a valid change to the memristor between the "pre-neuron" output and the "post-neuron" input. Depending on the order of the spikes, the memristor state either reduces or increases resistance.

C. Synapse Modeling

The synapse model is based on the nonlinear drift diffusion model of a charge based memristor presented in [10]. The synapse consists of memristors that are coupled with diodes in an anti-parallel configuration in order to instill the makings of a threshold device. This synaptic realization is due to the fact that crossbar devices are leaky and MIM diode isolation is one of the methods used to combat the effect of leaky devices and mitigating the effects of sneak paths.

The nonlinear behavior of the sandwiched doped/ undoped TiO_2 HP Labs memristor is governed by (1) and (2) and implemented behaviorally in Verilog-A.

$$\frac{dw}{dt} = \frac{\mu_D R_{ON}}{D} i(t) F\left(\frac{w}{D}\right) \qquad (1)$$

$$F_p(x) = 1 - (2x - 1)^{2p} \qquad (2)$$

The doped region width w is modulated according to (1) with the window function definition expressed in (2). For SPICE simulation the memristor model was implemented with parameter $p = 4$, memristor width $D = 10$ nm, and dopant mobility $\mu_D = 10^{-9}\ \mathrm{cm^2/V \cdot s}$. R_{ON} = 20 kΩ in this case, but with proper operation, the memristor is not driven to its lower limit of R_{ON}.

III. Simulation Results

A. XOR Simulation

The neuromorphic architecture is simulated in Cadence Analog Environment with IBM 90nm CMOS9RF process. The XOR simulation setup does not use all six input-output neuron pairs. Four neuron pairs are needed for the XOR operation. For example, to find the XOR between logic signals A and B, input A would be given to $\mathbf{n_{11}}$ and $\mathbf{n_{12}}$ while input B would be passed on to $\mathbf{n_{13}}$ and $\mathbf{n_{14}}$. The outputs would be read from the sum of $\mathbf{n_{22}}$ and $\mathbf{n_{23}}$. Fig. 4 provides the results for the XOR operation for all cases. Fig. 4a provides results for the case when both inputs A and B are Logic "0" thereby producing no spiking behavior at the outputs. Fig. 4b and Fig. 4d provide the scenarios when one input is Logic "1" and the other input is Logic "0." In

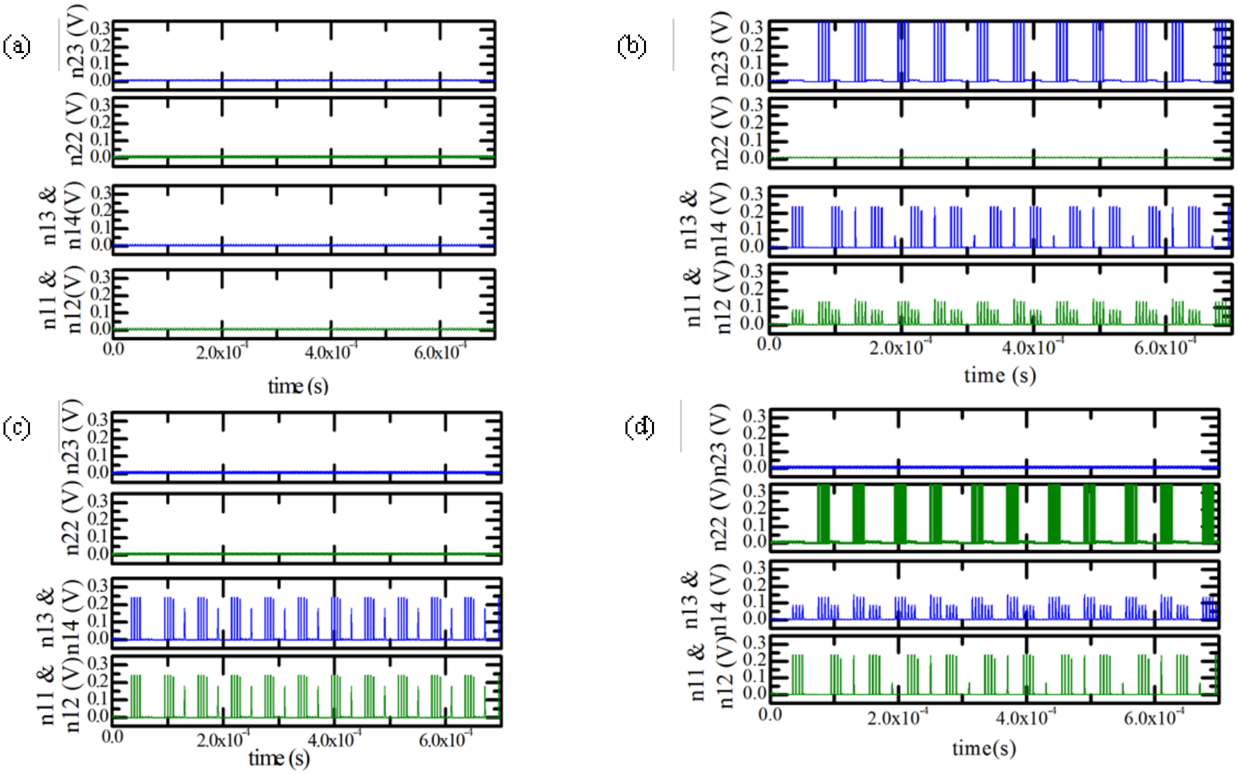

Fig. 4 XOR simulation results showing: (a) Input A ($\mathbf{n_{11}}$ and $\mathbf{n_{12}}$) = "0," and Input B ($\mathbf{n_{13}}$ and $\mathbf{n_{14}}$)= "0," so Output ($\mathbf{n_{22}}$ or $\mathbf{n_{23}}$)= "0." (b) Input A ($\mathbf{n_{11}}$ and $\mathbf{n_{12}}$) = "0," and Input B ($\mathbf{n_{13}}$ and $\mathbf{n_{14}}$)= "1," so Output ($\mathbf{n_{22}}$ or $\mathbf{n_{23}}$)= "1." (c) Input A ($\mathbf{n_{11}}$ and $\mathbf{n_{12}}$) = "1," and Input B ($\mathbf{n_{13}}$ and $\mathbf{n_{14}}$)= "1," so Output ($\mathbf{n_{22}}$ or $\mathbf{n_{23}}$)= "0." (d) Input A ($\mathbf{n_{11}}$ and $\mathbf{n_{12}}$) = "1," and Input B ($\mathbf{n_{13}}$ and $\mathbf{n_{14}}$)= "0," so Output ($\mathbf{n_{22}}$ or $\mathbf{n_{23}}$)= "1."

Fig. 4b, Input A is Logic “0” and Input B is Logic “1.” The result from the simulation shows that $\mathbf{n_{23}}$ spikes in a pattern that signifies Logic “1” while $\mathbf{n_{22}}$ does not spike at all. The XOR post processing will integrate and add the results of $\mathbf{n_{22}}$ and $\mathbf{n_{23}}$ to obtain a final verdict. The spiking behavior of either $\mathbf{n_{22}}$ and $\mathbf{n_{23}}$ should be deciphered as a Logic “1” by the post-processing circuitry. Fig. 4d provides results and works in a similar way to Fig. 4b except this time, instead of $\mathbf{n_{23}}$ spiking and $\mathbf{n_{22}}$ not spiking, $\mathbf{n_{23}}$ does not spike but $\mathbf{n_{22}}$ spikes. The results from post-processing will be the same as the previous case.

Lastly, Fig. 4c shows the case when both inputs A and B are Logic “1.” The results show that neither $\mathbf{n_{22}}$ or $\mathbf{n_{23}}$ spikes therefore providing output results similar to Fig. 4a. As expected, the XOR operation is verified with all test cases and shows that the neuromorphic architecture works as expected. Due to the bidirectional nature of the output node, Logic “0” when inputs do not induce spiking is different from Logic “0” when inputs induce spiking. For example, the Logic “0” seen for $\mathbf{n_{11}}$ & $\mathbf{n_{12}}$ in Fig. 4d looks different from that of Fig. 4a. The disturbance seen is directly related to the spiking behavior of the second layer of neurons that directly cause a disturbance in the output node of the input neurons.

B. *Edge Detection and XOR Simulation*

The edge detector operation is similar to XOR as shown in Fig. 5. In Fig. 5, the input neurons $\{\mathbf{n_{11}},\ldots,\mathbf{n_{16}}\}$ receive “011110” respectively, and they cause the output neurons $\{\mathbf{n_{21}}, \ldots, \mathbf{n_{26}}\}$ to produce “010010” respectively. In the input pattern, there are two edges, i.e. between $\mathbf{n_{11}}$ and $\mathbf{n_{12}}$ and between $\mathbf{n_{15}}$ and $\mathbf{n_{16}}$, and the neural network configuration was able to extract these edges in the output spiking pattern. The post-processing on the edge detector will integrate each output to determine output logic level.

The verification of the edge detector is done by showing another pattern with input neurons $\{\mathbf{n_{11}},\ldots,\mathbf{n_{16}}\}$ receiving “100110” respectively. This pattern clearly has two edges between $\mathbf{n_{13}}$ and $\mathbf{n_{14}}$ and between $\mathbf{n_{15}}$ and $\mathbf{n_{16}}$. Another observation here is that since there is no wrap-around effect in the neural architecture, the neural network identifies Logic “1” values at the extremes as edges. This design decision is architecture dependent and behavior may be changed by modifying the synaptic weights of the synapses controlling neuron behavior at the extremes. The result for the input pattern “100110” turns out to be “100110” and the post-processing for the edge detection should be able to extract the position of the edges quite clearly.

The CMOS neuron is scaled down from 130 nm CMOS process to 90 nm. In addition to process migration, the inclusion of synaptic weight dependent excitatory inputs was made. The neuron design migrated from a mostly saturation design to a mostly subthreshold design to improve power efficiency. When conducting the simulation for edge detection and XOR, the average power consumption per neuron during the spiking is about 0.3 µW which is more than an 8X saving over [9].

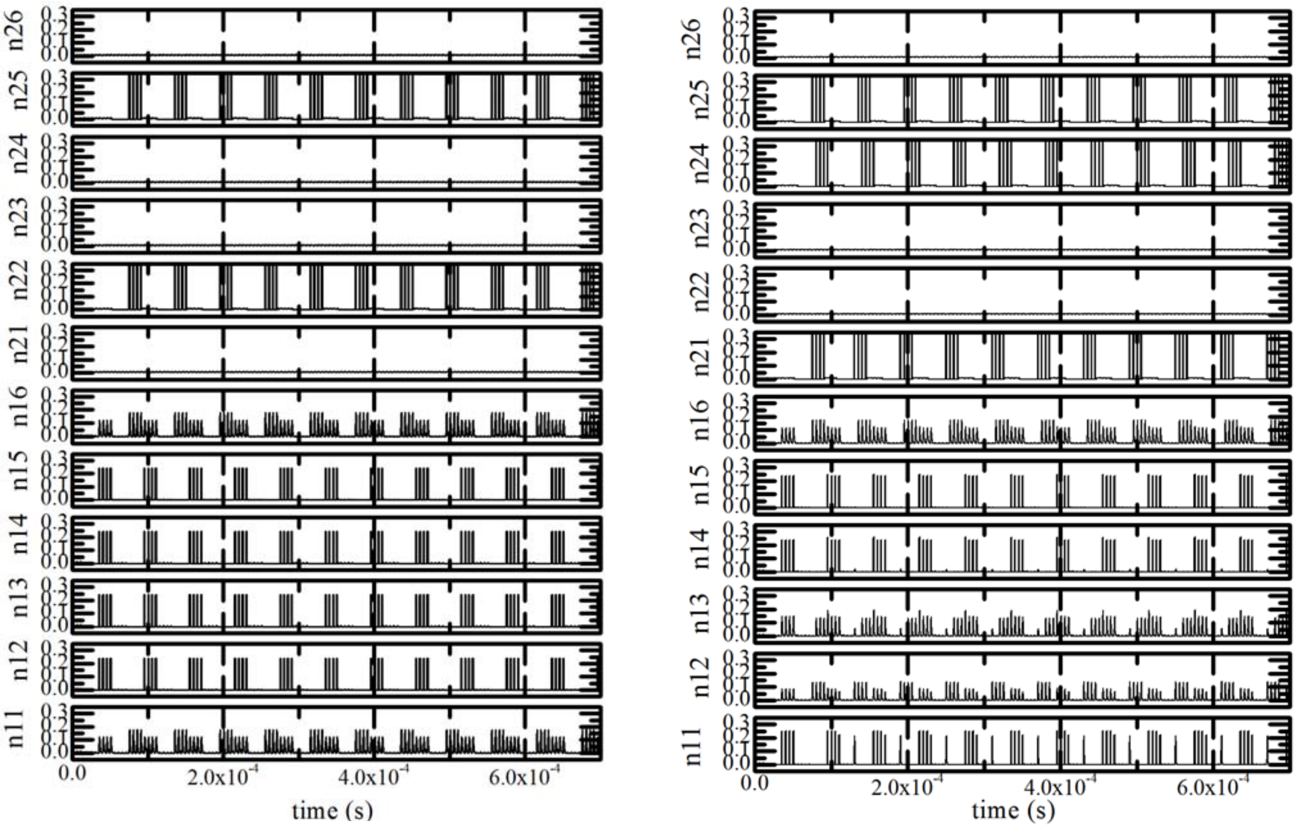

Fig. 5 (Left) Edge detection simulation results for input pattern “011110” produces output pattern “010010.” (Right) Edge detection simulation results for input pattern “100110” produces output pattern “100110”

III. Conclusion

A neuromorphic architecture made with memristors is presented. The architecture is shown to perform the XOR and edge detection operations after a supervised learning process. The design is simulated in 90 nm IBM CMOS process with power consumption while spiking at 0.3 μW. The amenable architecture is great for the memristor crossbar design, allowing the area savings possible with building crossbars above CMOS circuitry. The overall purpose of this work is to explore low level computing components that can utilize nanodevices in a manner that encourages parameter adjustment to facilitate on-site tuning when necessary.

References

[1] L. O. Chua, "MEMRISTOR - MISSING CIRCUIT ELEMENT," *IEEE Transactions on Circuit Theory,* vol. CT18, no. 5, pp. 507-&, 1971.

[2] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature,* vol. 453, no. 7191, pp. 80-83, May, 2008.

[3] K. Iniewski, G. S. Rose, and H. Manem, "A Hybrid CMOS-Nano FPGA Based on Majority Logic: From Devices to Architecture," *CMOS Processors and Memories*, Analog Circuits and Signal Processing, pp. 139-161: Springer Netherlands.

[4] G. S. Snider, and R. S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," *Nanotechnology,* vol. 18, no. 3, pp. 11, Jan, 2007.

[5] M. Harika, S. R. Garrett, H. Xiaoli, and W. Wei, "Design considerations for variation tolerant multilevel CMOS/Nano memristor memory," in Proceedings of the 20th symposium on Great lakes symposium on VLSI, Providence, Rhode Island, USA.

[6] B. Muthuswamy, and P. P. Kokate, "Memristor-based chaotic circuits," *IETE Technical Review,* vol. 26, no. 6, pp. 417, 2009.

[7] B. Linares-Barranco, and T. Serrano-Gotarredona, "Memristance can explain Spike-Time-Dependent-Plasticity in Neural Synapses," in Nature Precedings <http://hdl.handle.net/10101/npre.2009.3010.1>, 2009.

[8] S. Cassenaer, and G. Laurent, "Hebbian STDP in mushroom bodies facilitates the synchronous flow of olfactory information in locusts," *Nature,* vol. 448, no. 7154, pp. 709-U12, 2007.

[9] I. Ebong, and P. Mazumder, "Memristor based STDP Learning Network for Position Detection," in 22nd International Conference on Microelectronics (ICM), Cairo, Egypt, 2010, pp. 292-295.

[10] Y. N. Joglekar, and S. J. Wolf, "The elusive memristor: properties of basic electrical circuits," *European Journal of Physics,* vol. 30, no. 4, pp. 661-675, Jul, 2009.