Low Power Memristor Crossbar Based Winner Takes All Circuit

B Rasitha Fernando
Department of Electrical and Computer Engineering
University of Dayton
Dayton, USA
Email: fernandob1@udayton.edu

Raqibul Hasan
Department of Electrical and Computer Engineering
University of Dayton
Dayton, USA
Email: hasanm1@udayton.edu

Tarek M. Taha
Department of Electrical and Computer Engineering
University of Dayton
Dayton, USA
Email: tarek.taha@udayton.edu

***Abstract*— Edge devices often have to process data at low power and would benefit from being adaptable. Given that the data coming into these devices is generally unlabeled, unsupervised training on these devices is beneficial. This paper examines a low power approach to implement the winner takes all algorithm, for self-organizing maps through a memristor crossbar based circuit. A novel neuron circuit is designed for the winning neuron detection and lateral inhibition operations. Our experimental results show that the proposed system can self-organize based on unlabeled training data. The proposed design was around 0.002mm^2 in area and consumed about 0.2mW of power. When compared to a CPU, the design had a higher error rate, but was 100 times faster and consumed much lower area and power. Thus when area or power reduction are crucially important, this approach is quite viable.**

Keywords— Self-organizing-maps; memristor crossbars; winner takes all; online training.

I. INTRODUCTION

There is now a vast proliferation of edge devices that are generally low power and have limited internet connectivity. These devices often have to process data at low power and would benefit from being adaptable. Given that the data coming into these devices is generally unlabeled, unsupervised training on these devices is beneficial. This paper examines a low power approach to implement self-organizing map (a category of competitive learning networks that does not require labeled data).

To achieve low power, we utilize memristor based crossbar circuits for learning. The memristor [1] is a novel non-volatile device having a large varying resistance range. Physical memristors can be laid out in a high density grid known as a crossbar [2]. A memristor crossbar can perform many multiply-add operations in parallel and the conductance of multiple memristors in a crossbar can be updated in parallel [3,4]. Multiply-add operations are the dominant operations in neural networks and the training of neural networks requires the update of synaptic weights iteratively. As a consequence, memristors have a great potential as a synaptic element in a neural network based system design.

This paper implements a memristor crossbar based self-organizing system which learns from unlabeled data. We are using memristor crossbars for high synaptic weight density and for very low energy parallel analog processing. The system essentially implements the winner takes all learning algorithm. A novel neuron circuit is designed for the winner neuron detection and lateral inhibition operations (to inhibit other neurons from firing). The proposed system could be used as an online learning system for unlabeled data clustering.

The most recent results for memristor based neuromorphic systems can be seen in [3-6]. Alibart [3] and Preioso [4] examined memristor based linearly separable classifier designs. Soudry et al. [5] proposed the implementation of gradient descent based training on memristor crossbar neural networks. Choi et al. examined dimensionality reduction utilizing a memristor based PCA (principal component analysis) system [6]. They implemented Sanger's learning rule, a derivative of Hebb's rule, in memristor crossbar to obtain the principal components. Długosz et al. [7] examines CMOS implementation of winner takes all algorithm which utilizes Euclidean distance calculation and comparators. Proposed memristor based design would be significantly area and power efficient compared to the CMOS implementations.

The proposed design was around 0.002mm^2 in area and consumed about 0.2mW of power. When compared to a CPU, our design had a higher error rate, but was 100 times faster and consumed much lower area and power. Thus when area or power reduction are crucially important, this approach is quite viable. A preliminary version of this work without any area, power, timing, or accuracy analysis was presented in [8].

The rest of the paper is organized as follows: section II describes memristor based neuron circuit and implementation of the winner takes all algorithm for self-organizing system design. Section III describes experimental setup and results. Finally, future work in section IV and we concluded our work in section V.

II. MEMRISTOR BASED SELF-ORGANIZING SYSTEM

A. Learning Algorithm

This work utilizes a variant of the self-organizing-maps (SOM) algorithm [9] known as winner takes all [10], which is more suitable for memristor based implementations. In this algorithm, neurons in a layer compete with each other for activation. Only the neuron with the highest activation (or similarity) stays active while all other neurons stay inactive. The winner takes all learning algorithm is stated below:

Randomly initialize n neuron weights, w_j (for $j=1,2,..n$)
For each training data, p_i
  Calculate similarity, s_{ij} between p_i and w_j (for $j=1,2,..n$)
  $k=arg\ max_j\ s_{ij}$
  $w_k=(1-\alpha)\ w_k+\alpha\ p_i$ where α is the learning rate
Repeat until convergence

B. Neuron Circuit

Fig. 1 shows the memristor crossbar based neuron circuit used in this paper. In this figure, each column is implementing a neuron. The conductance of the memristor at each row of a column represents the synaptic weight for the corresponding input in that row. At the bottom of a neuron circuit, there is a capacitor which accumulates charge after an input is applied to the network. At steady state, the potential across the capacitor is the normalized dot product of the inputs and weights ($(A\sigma_A + B\sigma_B + C\sigma_C + 0.\sigma_L)/(\sigma_A+\sigma_B+\sigma_C+\sigma_L)$) for the neuron in the first column) as R_h is a high resistance. The capacitor is used to determine the winning neuron after an input is applied (detailed later). The neurons have a bias input of 0V and the conductance value corresponding to the bias input is σ_L. If all the inputs are 1 without this bias all the V_i would be very close and it would not be easy to distinguish the winner neuron for this input.

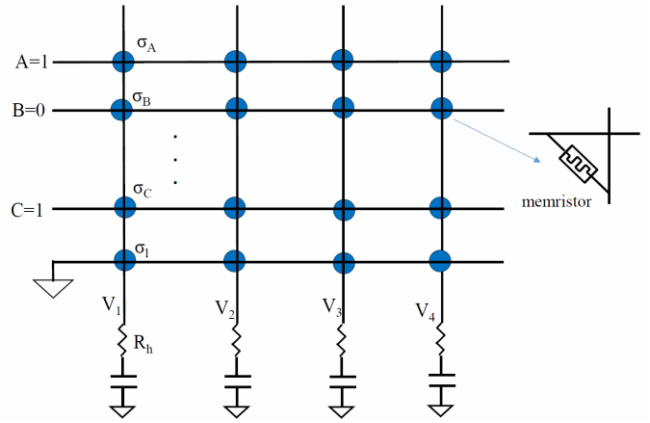

Fig. 1. Memristor crossbar based neuron circuit for self-organizing circuit design. A, B, C are the inputs applied as voltage and σ_A, σ_B, σ_C are the memristor conductances working as weights of a neuron.

C. Similarity Measure

The training algorithm requires an evaluation of the similarity between inputs and neuron weights. The cosine similarity is a popular metric for similarity measure. The cosine similarity between two vectors X and W_1 is $X.W_1/(||X||.||W_1||)$. If we need to compare the similarity between X and $W_1...W_n$, a comparison of $X.W_i/||W_i||$ for $i=1,2,..,\ n$ would provide the cosine similarity comparison. In this paper we propose to use the normalized dot product of the inputs and weights as the similarity metric between them. The normalized dot product approximates the cosine similarity by dividing the dot product by the *l-1 norm* of W_i instead of the *l-2 norm*.

Recently, several studies [11,12] used the dot product as a similarity metric for self-organizing circuit design. The dot product is a very rough similarity estimate and does not work well for clustering/self-organization. Table I shows a 2×2 weight matrix (columns 1 and 2 correspond to neurons 1 and 2 respectively, and each neuron has 2 inputs). Table II shows the similarity measures for the input $[1\ 0]^T$ by applying different methods. If the dot product is considered, *neuron* 1 (column 1) gives highest similarity over *neuron* 2. Considering cosine similarity and normalized dot product, *neuron* 2 is more similar to $[1\ 0]^T$ which is more reasonable.

TABLE I. NEURON WEIGHTS

	Neuron 1	***Neuron 2***
Input 1	1	0.9
Input 2	1	0.7

TABLE II. SIMILARITIES BETWEEN INPUT $[1\ 0]^T$ AND THE NEURONS

	Dot Product	**Cosine similarity**	**Normalized dot product**
Neuron 1	1	0.707	0.5
Neuron 2	0.9	0.789	0.562

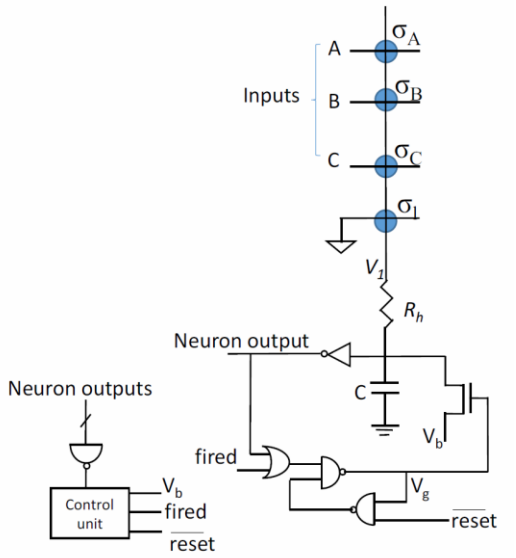

Fig. 2. Neuron circuit with additional components for training. V_g goes high only for the winner neuron and weights are updated only for that winner) neuron.

D. Winner Takes All Logic Implementation

In Fig. 1, after a new input is applied, the starting potential across the capacitor is approximately the normalized dot product of the inputs and weights ($V_1=[A\sigma_A + B\sigma_B + C\sigma_C + 0.\sigma_l]/[\sigma_A+\sigma_B+\sigma_C+\sigma_l]$ for the neuron in first column) as R_h is a high resistance. The capacitor corresponding to a higher V_i will charge faster than other capacitors and the capacitor which exceeds a particular threshold first is considered as the winning neuron. A CMOS inverter is used to detect when a capacitor's potential crosses a threshold of 0.1 V (see Fig. 1). The neuron whose capacitor crosses the threshold voltage first, switches voltage V_g from low to high (referred to as a neuron firing) and updates the weights in the weight update phase (detailed later). Fig. 2 shows the additional components for detecting firing and weight update. The control unit synchronizes the operations during training and is shared by all the neurons in a crossbar.

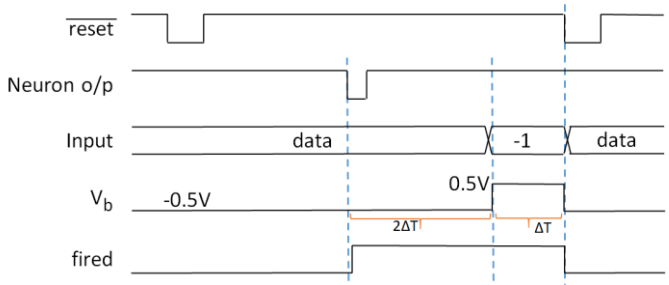

Fig. 3. Timing of the signals during training. After the *reset_bar* pulse the neurons start integrating currents. The winner neuron fires first and V_g goes high for only for the winner. Then *fired* signal goes high and no other neuron can switch V_g to high. Weight are only updated for the winner neuron.

Once a neuron fires for a particular input, it needs to inhibit other neurons from firing. This is important, because weights will be updated only for the winning neuron. The control unit takes the output of all the neurons and performs a NAND operation. If any of the neuron outputs switches from high to low, the output of the NAND gate will switch from low to high. The control unit propagates this firing event through the *fired* line after a small delay. As a result, any other neuron cannot switch its V_g to high, and thus cannot update its weight in the weight update phase.

E. Weight Update

In general, a certain energy (or threshold voltage, V_{th}) is required to enable the state change in a memristive device [13]. When the electrical excitation through a memristor exceeds the threshold, i.e., $V(t)>V_{th}$, the resistance of the device changes. Otherwise, a memristor behaves like a resistor. The device characterized in [13] has a threshold voltage of about 1.3V.

For the winner neuron, weights are updated in two steps. The conductance of the memristors corresponding to high inputs are increased by 2Δ. After that, the conductance of all the memristors of that neuron are decreased by Δ. Fig. 3 shows the amplitude of different signals during training. Assume that neuron i fires at time t. Just after time t, V_g of neuron i switches from low to high, which turns on the NMOS transistor and discharges the capacitor. Eventually, the potential at the bottom of the fired neuron becomes V_b=*-0.5 V* (see Fig. 3). Memristors corresponding to a high input get a potential across them of about 1.5 V (which is greater than the device threshold voltage) and increases their conductance. The control unit keeps V_b= *-0.5 V* till time $t+2\Delta T$.

At time $t+2\Delta T$, the control unit switches V_b to +0.5V and puts a potential of -1V at each of the input rows. This causes the potential across all the memristors (belonging to the winner neuron) to become -1.5V and the conductance of the memristors decrease. This scenario is sustained till time $t+3\Delta T$. At time $t+3\Delta T$, the *reset_bar* pulse is applied along with V_b= *-0.5V*, *fired*=0V, and the next training data is applied at the inputs. The net effect on the winner neuron, is that for the memristors corresponding a high input, there is an increase in conductance for ΔT time. For the memristors corresponding to low inputs, the conductance will decrease for time ΔT. The duration ΔT determines the learning rate.

III. Experimental Setup

The memristor crossbar circuits were simulated in SPICE so that the memristor grid could be evaluated very accurately considering the crossbar sneak-paths and wire resistances. A wire resistance of 5Ω between memristors in the crossbar is considered in these simulations. The memristors were simulated with an accurate model of the device published in [14]. This device has a minimum resistance of 10 kΩ, a maximum resistance of 100 kΩ, and the full resistance range of the device can be switched in 100 ns by applying 1.5 V across the device. Values of σ_l, R_h and C in Fig. 2 are 10 µS, 1 MΩ, and 100 fF respectively.

We implemented the theoretical SOM in MATLAB without the 'forgetting term' as in [15] to test the proposed memristor hardware device implementing winner takes all learning scenario, where the weight update was:

$$\frac{\partial w_{ij}(t)}{\partial t} = \begin{cases} \propto [x_i(t) - w_{ij}(t), & if\ j \in \eta(t) \\ 0 & if\ j \notin \eta(t) \end{cases}$$

Here,

α: learning rate,

$w_{ij}(t)$: existing weights

j: winner neuron

$\eta(t)$: range or size of bubble for the neighboring weights which is assumed as 1 in our work.

IV. Results

A. 2D Synthetic Data

We considered training synthetic data with 120 two dimensional vectors which were centered around (0, 1), (1, 0), and (1, 1). A 3×3 crossbar with randomly initialized conductances was used for the training. The third row of the crossbar was the bias input and was connected to ground. Each resistor in the third row had a conductance value of σ_l and was not trained. Fig. 4 shows the distribution of the inputs, initial and final trained weights. In different training trials, different clusters are learned by different neurons in the crossbar, based on the randomly initialized weights. While the theoretical approach had a misclassification error of 0%, the proposed design had an average misclassification rate of 27% (average error for 1000 trials). The key reason for this is due to the memristors reaching maximum or minimum conductance levels after a certain number of training pulses. This problem does not occur in the theoretical algorithm, and thus the error is not present there.

As shown in Fig 5(a), we tested another synthetic 2D data set centered around (0, 0.5), (0.7, 0.3), (0.9, 0.85). We observed that the proposed approach still gave the same range of average error (31%). We also observed that the final weight values leaned towards the extremes of the system (higher or lower conductance of the memristor) [15]. As shown in Fig 5(b), the final weights of our results were (0.2, 1), (1, 0.1) and (1, 1) when theoretical results should be at (0, 0.5), (0.7, 0.3) and (0.9, 0.85) as in 5(c).

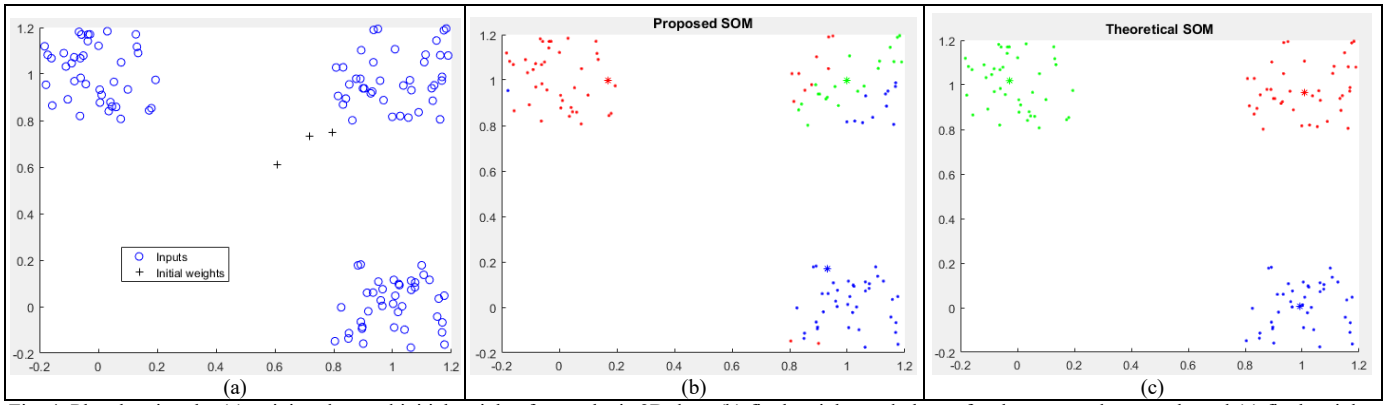

Fig. 4. Plot showing the (a) training data and initial weights for synthetic 2D data, (b) final weights and cluster for the proposed approach, and (c) final weights and cluster for the Matlab implementation of the SOM algorithm. Weights are plotted as conductance×10^4.

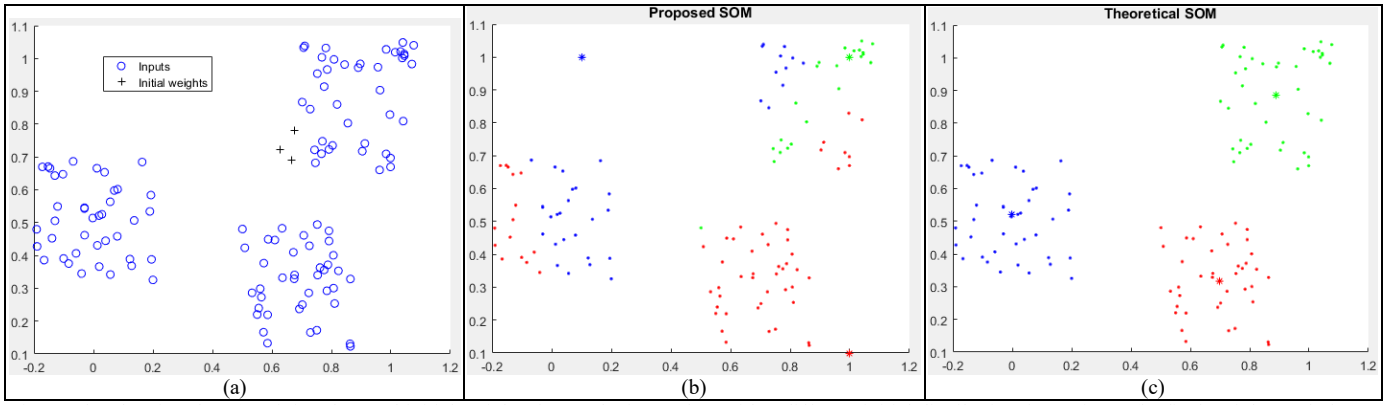

Fig. 5. Plot showing the (a) training data and initial weights for synthetic 2D data with a different initial distribution from Fig. 4, (b) final weights and cluster for the proposed approach, and (c) final weights and cluster for the Matlab implementation of the SOM algorithm. Weights are plotted as conductance×10^4.

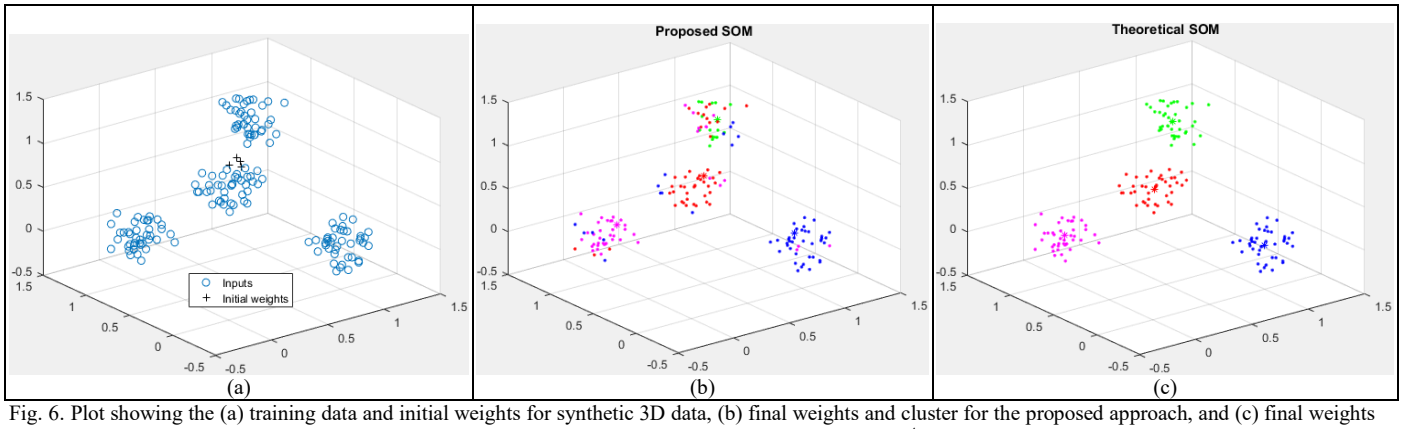

Fig. 6. Plot showing the (a) training data and initial weights for synthetic 3D data, (b) final weights and cluster for the proposed approach, and (c) final weights and cluster for the Matlab implementation of the SOM algorithm. Weights are plotted as conductance×10^4.

B. 3D Synthetic Data

This experiment considered training synthetic data with 160 three dimensional vectors which were centered around (0, 0, 1), (1, 0, 0), (0, 1, 0), and (1, 1, 1). A 4×4 crossbar with randomly initialized conductances was used for the training. The fourth row of the crossbar was the bias input and was connected to ground. Each resistor in the fourth row had a conductance value of σ_I and was not trained. Fig. 6 shows the distribution of the inputs, initial and final trained weights.Similar to results in the 2D synthetic data, we observe some error in our design as well as in the theoretical SOM implementation. The particular image for the theoretical SOM given below has zero error, but on average (after 1000 trials) has a 6.11% mis-classification error while our design has 29.76% error.

C. Wisconsin Breast Cancer Dataset

The Wisconsin Breast Cancer Dataset is a well-known dataset among the Self Organizing Maps community and created by Dr. William H. Wolberg (a physician) at the University of Wisconsin Hospitals, Madison in 1992 [16]. It has 9 different features scaling from 1 to 10 in integer format, and has two classes of cancer, benign and malignant. A 10×2

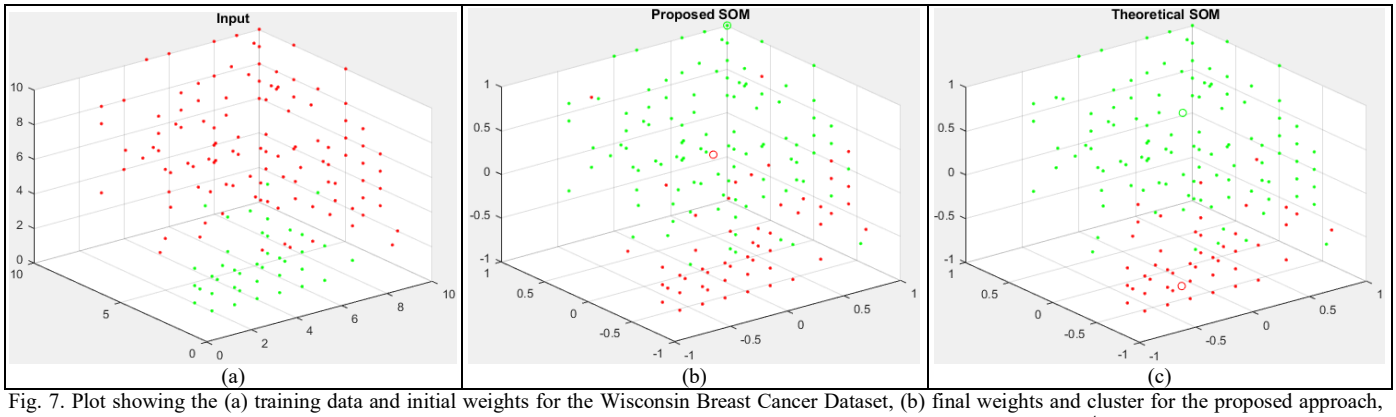

Fig. 7. Plot showing the (a) training data and initial weights for the Wisconsin Breast Cancer Dataset, (b) final weights and cluster for the proposed approach, and (c) final weights and cluster for the Matlab implementation of the SOM algorithm. Weights are plotted as conductance×10^4.

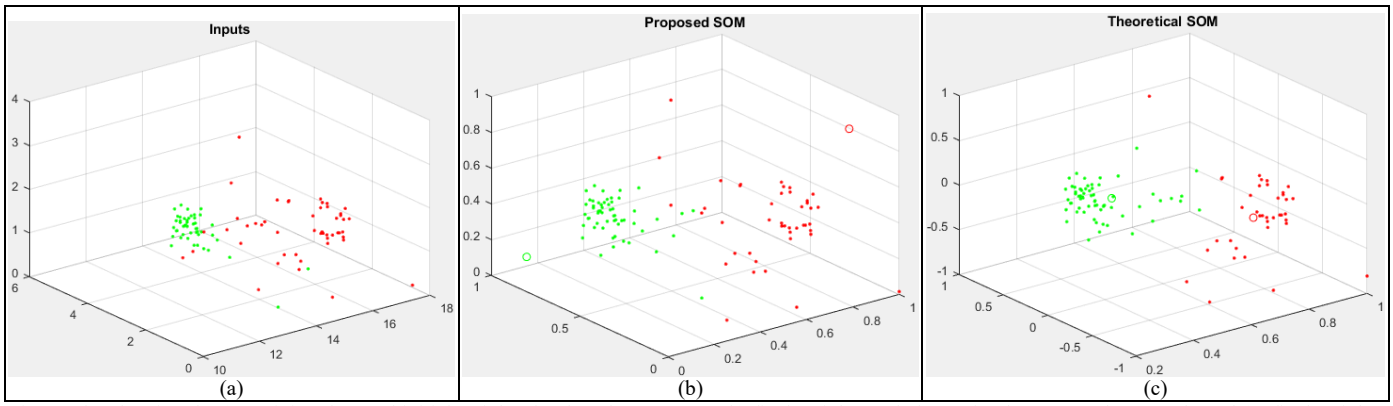

Fig. 8. Plot showing the (a) training data and initial weights for the Glass Dataset, (b) final weights and cluster for the proposed approach, and (c) final weights and cluster for the Matlab implementation of the SOM algorithm. Weights are plotted as conductance×10^4.

crossbar with randomly initialized conductances was used for the training. The tenth row of crossbar was the bias input and was connected to ground. Each resistor in the tenth row had the conductance value σ_I and was not trained. For this dataset, the proposed design give a 9.5% error while the theoretical SOM gave 4.2% error as shown in Fig 7.

D. Glass Identification Dataset

The Glass Identification dataset was created by B. German of the Home Office Forensic Science Service in 1987 [17]. It has features of glasses with the weight percent of 8 different periodic table elements in 7 types. All of them are a subset of two main classes: window glasses or non-window glasses. Dataset values vary in continuous range between 0 and 100. A 9×2 crossbar with randomly initialized conductances was used for the training. The ninth row of the crossbar was the bias input and was connected to ground. Each resistor in the ninth row had a conductance value of σ_I and was not trained. The proposed design had a higher accuracy than the theoretical SOM for the Glass dataset as shown in Fig 8. Errors were 7% and 14% for the proposed design and the theoretical SOM respectively.

E. Power and Performance

Table III shows the area, time and power of the benchmarks examined. Time and power are given for a single sample and a single iteration. Time is in the range of 0.26µs while power in the range of 0.2mW. The CPU implementation of the theoretical SOM using MATLAB gave an average time in the range of 26µs. When compared to the CPU in terms of area, time, and power, the proposed design has significant advantages.

TABLE III. AREA, TIME AND POWER

Dataset	Area (mm^2)	Time (µs)	Power (mW)	CPU Matlab Time (µs)
2D synthetic	0.00165	0.258	0.190	26.071
3D synthetic	0.00169	0.260	0.205	25.518
Breast Cancer	0.00175	0.260	0.235	26.282
Glass	0.00173	0.258	0.228	25.445

TABLE IV. MIS-CLASSIFICATION ERROR FOR DIFFERENT DATASETS

Dataset	Crossbar size (features × classes)	Memristor SOM Error %	Theoritcal SOM Error %
2D Synth	3 x 3	27.10	0.00
3D Synth	4 x 4	29.76	6.11
Breast Cancer	10 x 2	9.50	4.20
Glass Identification	9 x 2	7.20	14.30
Tic Tac Toe	9 x 2	45.75	50.00
Zoo	16 x 7	60.21	72.00
Veterabal	6 x 2	41.42	29.28
Balance	4 x 2	47.50	42.00
Diabetes	8 x 2	47.60	46.20
AVERAGE		**35.12%**	**29.34%**

Table IV represent the mis-classification error of the proposed design as well as the base line theoretical SOM. Results for five other datasets are presented, along with the four described earlier. These results show that the proposed memristor approach has on average about 5.8% more error than the theoretical approach (though for individual cases, the actual error may be much higher). The key source of this error is that the memristor conductance may reach a maximum or minimum value from multiple inputs.

V. Conclusion

This paper presented a memristor crossbar based winner takes all circuit design for self-organization. The proposed neuron circuit accumulates charge over time and the neuron closest (based on the normalized dot product) to the input, exceeds a threshold voltage before any other neuron. Once a neuron fires, it inhibits others from firing. Hence only weights of the winner neuron are updated. Our experimental results show that the proposed system can self-organize based on the unlabeled data applied to the system.

The proposed design has significant area, power, and speed advantages over a digital CPU based design, but has more error. Thus when area or power reduction are crucially important, this approach is quite viable. The error is due to the memristor conductance reaching a limit (max or min) from multiple resistance changing pulses (one per input). As future work, an approach is needed to avoid this saturation. The proposed approach could be used for unsupervised training of data at very low power consumption. Replacing the neuron output circuit with specialized devices could also reduce the area and power consumption of the system.

Acknowledgements

This work was supported through NSF grant CCF-1718633 and a University of Dayton Graduate Fellowship.

References

[1] L. O. Chua, "Memristor—The Missing Circuit Element," IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507–519 (1971).

[2] W. Lu, K.-H. Kim, T. Chang, S. Gaba, “Two-terminal resistive switches (memristors) for memory and logic applications,” in Proc. 16th Asia and South Pacific Design Automation Conference, 2011, pp. 217-223.

[3] F. Alibart, E. Zamanidoost, and D.B. Strukov, "Pattern classification by memristive crossbar circuits with ex-situ and in-situ training", Nature Communications, 2013.

[4] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, D. B. Strukov, “Training and operation of an integrated neuromorphic network based on metal-oxide memristors,” Nature, 521(7550), 61-64, 2015.

[5] D. Soudry, D. D. Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-Based Multilayer Neural Networks With Online Gradient Descent Training,” IEEE Trans. on Neural Networks and Learning Systems, issue 99, 2015.

[6] Choi, Shinhyun, Patrick Sheridan, and Wei D. Lu. "Data Clustering using Memristor Networks." Scientific Reports 5 (2015).

[7] Rafał Długosz, Tomasz Talaśka, Witold Pedrycz, and Ryszard Wojtyna. 2010. Realization of the conscience mechanism in CMOS mplementation of winner-takes-all self-organizing neural networks. Trans. Neur. Netw. 21, 6 (June 2010), 961-971.

[8] Raqibul Hasan and Tarek M. Taha, "Memristor Crossbar Based Winner Take All Circuit Design for Self-organization," Proceedings of the Neuromorphic Computing Symposium, July 2017.

[9] Kohonen, Teuvo. "The self-organizing map." Proceedings of the IEEE 78, no. 9 (1990): 1464-1480.

[10] Lemmon, Michael, and BVK Vijaya Kumar. "Competitive learning with generalized winner-take-all activation." IEEE transactions on neural networks 3, no. 2 (1992): 167-175.

[11] S. Yu, "Orientation classification by a winner-take-all network with oxide RRAM based synaptic devices," 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne VIC, 2014, pp. 1058-1061.

[12] S. N. Truong, K. Van Pham, W. Yang, K. S. Min, Y. Abbas, C. J. Kang, and K. Pedrotti, “Ta2O5-memristor synaptic array with winner-take-all method for neuromorphic pattern matching,” Journal of the Korean Physical Society, 69(4), 640-646.

[13] S. Yu, Y. Wu, and H.-S. P. Wong, "Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory," Applied Physics Letters 98, 103514 (2011).

[14] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, "Memristor SPICE Model and Crossbar Simulation Based on Devices with Nanosecond Switching Time," IEEE International Joint Conference on Neural Networks (IJCNN), August 2013.

[15] Yin, Hujun. "The self-organizing maps: background, theories, extensions and applications." Computational intelligence: A compendium. Springer Berlin Heidelberg, 2008. 715-762.

[16] UCI Machine Learning Repository: Breast Cancer Wisconsin (Original) Data Set, https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original).

[17] UCI Machine Learning Repository: Glass Identification Data Set, https://archive.ics.uci.edu/ml/datasets/glass+identification.