3D Memristor Crossbar Architecture for a Multicore Neuromorphic System

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

Yangjie Qi
Department of Electrical and Computer Engineering
University of Dayton
Dayton, OH, USA
qiy001@udayton.edu

Chris Yakopcic
Department of Electrical and Computer Engineering
University of Dayton
Dayton, OH, USA
cyakopcic1@udayton.edu

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

***Abstract*—Compact online learning architectures can be used to enhance internet of things devices, allowing them to learn directly on received data instead of sending data to a remote server for learning. This saves communication energy and enhances privacy and security, as the data is not shared. This paper presents a 3D memristor based multicore architecture capable of on-line learning, where a mixed signal design utilizes memristor crossbars to both store synaptic weights and to carry out analog dot product computations. A 3D memristor architecture is one where several memristor crossbars are stacked vertically on top of each other to reduce the area footprint of a chip. A variety of deep learning applications can be processed using the proposed architecture, and we show how the simulated architecture is able to learn and classify several datasets. In this work we also examine how changing the number of vertically stacked crossbars within a neural core impacts energy, time, and area requirements of the architecture.**

Keywords–Low power architecture; Memristor crossbars; Deep neural network; Neuromorphic computing, 3D-stack

I. Introduction

With the increasingly large volumes of data being generated in nearly all sectors including commerce, science, medicine, security, and social networks, one of the key application drivers of future computing systems is the processing of these data. This process is typically described as data analytics and often involves supervised learning to analyze and categorize the data. Deep learning algorithms are very popular for this task.

In supervised learning, we know what data classes exist and the expected sample data for each class is available (often described as labeled data). Supervised machine learning algorithms use labeled datasets to train neural networks. Once trained, the algorithms can then classify new unlabeled data into the existing set of classes. Data processing using supervised learning algorithms is typically executed on clusters of GPUs. One of the major problems with these systems is their high power consumption and the need for higher throughput. Due to the limits of Dennard scaling and Moore's Law, alternative architectures are gaining interest.

Compact online learning architectures could be used to enhance internet of things devices to allow them to learn directly on received data, instead of having to transmit data to a remote server for learning. This saves communication energy and enhances privacy and security, as the data is not shared.

Several studies have examined low power, high throughput architectures for deep learning applications. Examples of these include DaDianNao [1], SOCRATES-D [2], and NeuroCube [3], which accelerate machine learning algorithms through digital circuits. The majority of recent specialized neural architecture studies [4-17] examined only recognition phase. Of the limited set of neural architecture papers examining learning [1-3], none presented memristor [18-21] based systems. The most recent results for memristor based neuromorphic systems are presented in [22,29]. Memristor based systems have the potential to operate at significantly lower power than digital systems due to their ability to perform highly parallel analog arithmetic operations. Furthermore, due to the non-volatile properties of memristors, learned weights are retained in memristor crossbars without a constant power source.

Memristors are used most effectively in a crossbar structure to rapidly evaluate multiply-add operations in parallel in the analog domain [23,24] (these are the dominant operations in neural networks). This enables high areal density neuromorphic systems with great computational efficiency [30]. To implement training, gradient descent learning in memristor based neural networks has also been proposed [22,25,31,32,33].

In this work we investigate a 3D memristor crossbar based architecture. To reduce the area footprint of the crossbars, multiple crossbars are stacked vertically and surrounding control circuitry is multiplexed to promote sharing of the digital circuitry. Previous research [34] suggests that multiple crossbars can be can be patterned on top of one another, and we examine the implications this will have on multicore neuromorphic architectures in terms of energy, time, and area.

Key novel aspects of this work include: 1) the proposed multicore architecture capable of on-line learning for executing deep neural networks, 2) a novel technique for locally mapping different neural networks for different applications onto the multicore system, 3) exploring the impact of stacking multiple crossbars within neural cores of a memristor based architecture. Then, examining time, energy, area, and accuracy of the system depending on the number of crossbars in a 3D stack design.

The rest of the paper is organized as follows: Section II describes each of the components in the proposed neural core architecture, including the back-propagation circuit. Section III describes how neural networks can be mapped to the proposed architecture, and Section IV describes the vertically stacked 3D memristor core. Section V describes experimental setup of our simulated vertically stacked crossbar studies, and Section VI discusses results. Finally, we conclud our work in Section VII.

II. System Overview

The presented architecture utilizes layers of memristor crossbars to implement neural networks. To increase areal density, this architecture provides the ability to stack memristor crossbars three dimensionally. In this section, we provide a description of each of the components in the proposed memristor neural core upon which this architecture is based.

A. Memristor Device

Memristors are essentially nanoscale variable resistors, where the resistance is modified if a voltage greater than a write threshold voltage (V_{th}) is applied [35]. Hence memristor resistance can be read by applying a voltage below V_{th} across the device, and device state will theoretically remain unchanged. Several research groups have demonstrated memristive behavior using different materials. The device modeled for this study is composed of HfOx and AlO_x [36], has a high on state resistance (RON $\approx$ 10kΩ), a very high resistance ratio (ROFF/RON $\approx$ 1000), and a write threshold voltage of about 1.3 V.

B. Memristor Crossbar

Physical memristors can be laid out in a high density grid known as a crossbar. The schematic and layout of a memristor crossbar are displayed in Fig. 1. A memristor in the crossbar structure occupies an area of $4F^2$ including the required spacing between devices (where F is the device feature size, or wire width). This is 36 times smaller than an SRAM memory cell. 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. Multiply-add operations are the dominant operations in neural networks, and neural network training requires updates of synaptic weights iteratively. Crossbar structures substantially aid in the ability to parallelize memristor based analog computation.

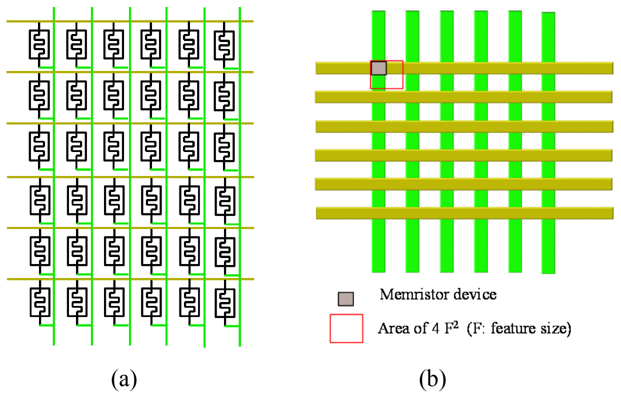

Fig. 1. (a) Memristor crossbar schematic and (b) memristor crossbar layout.

C. Memristor Neuron Circuit

In this work a two column neuron circuit is used as shown in Fig. 2. Memristors are typically utilized in neuromorphic systems to approximate the concept of synaptic connectivity. Thus, memristors can be used to store the connection strength between a neuron and all its incoming connections. This circuit requires two memristors to represent a single weight because the dynamic resistance of a memristor can really only be used to store a single positive bounded value. The left column of memristors represents a positive excitatory connection and the right column represents an inhibitory connection. In a given row, if $\sigma_{i+} > \sigma_{i-}$ then a net positive synaptic weight is observed, otherwise a negative synaptic weight will be present [22].

Assume that the value of the dot product (DP_{j}) can be calculated according to equation (1) as the voltage difference between the left and right column wires. Thus, each memristor crossbar in this system essentially performs a set of dot product calculations between the neuron input voltages and the net conductance of each memristor pair.

The output, y_j in Fig. 2 represents the neuron output. The sigmoid presented in equation (2) is typically used as the activation function in deep learning. However, in the presented memristor based neuromorphic system the approximated sigmoid function in equation (3) is used, as it is easier to generate using an amplifier circuit. TO approximate the sigmoid function, the power rails of the op-amps, V_{DD} and V_{SS}, are set to 0 and 1 V respectively. Fig. 3 displays the continuous sigmoid function along with the approximation used in this work.

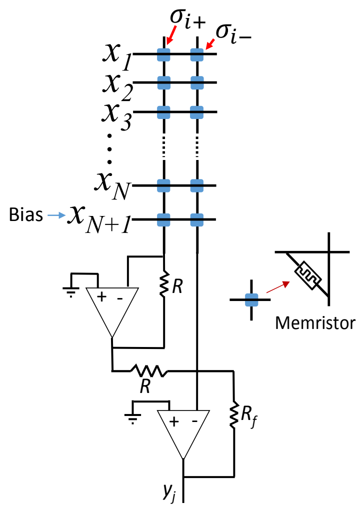

Fig. 2. Circuit diagram for a single memristor based neuron.

$$DP_j = \sum_{i=1}^{N+1} x_i \times \left(\sigma_{ij}^{+} - \sigma_{ij}^{-}\right) \tag{1}$$

$$f(x) = \frac{1}{1+e^{-x}} \tag{2}$$

$$g(x) = \begin{cases} 1, & x > 2 \\ 0.25x + 0.5, & |x| \le 2 \\ 0, & x < 2 \end{cases} \tag{3}$$

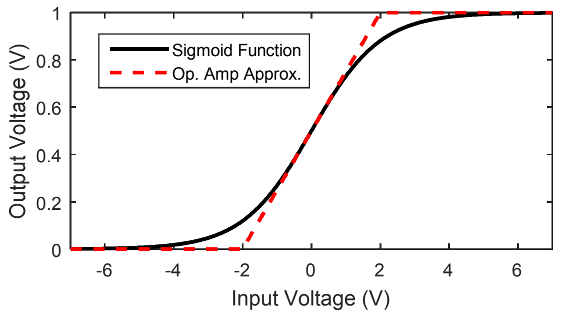

Fig. 3. Plot displaying the traditional sigmoid function along with the approximation used in this work.

D. *Memristor Based Neural Network*

Fig. 4 displays a circuit diagram that shows how multiple neuron circuits can be patterned using a memristor crossbar. One of these circuits will be required for each layer in the neural network.

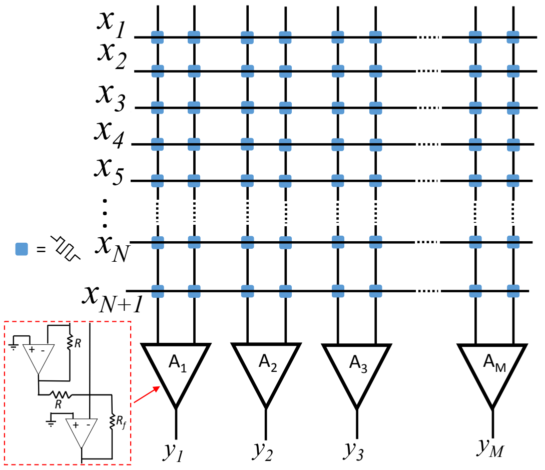

Fig. 4. Circuit diagram displaying how a memristor crossbar can implement one layer of the proposed memristor neural network.

In a multi-layered feed-forward neural network with four inputs as shown in Fig. 5 (a), all the neurons within a single layer utilize the same set of inputs. Thus, the memristor neuron circuit in Fig. 5 (a) can be replicated using memristor crossbars as shown in Fig. 5 (b). Each layer of the neural network in Fig. 5 (a) is replicated using a separate memristor crossbar in Fig. 5 (b).

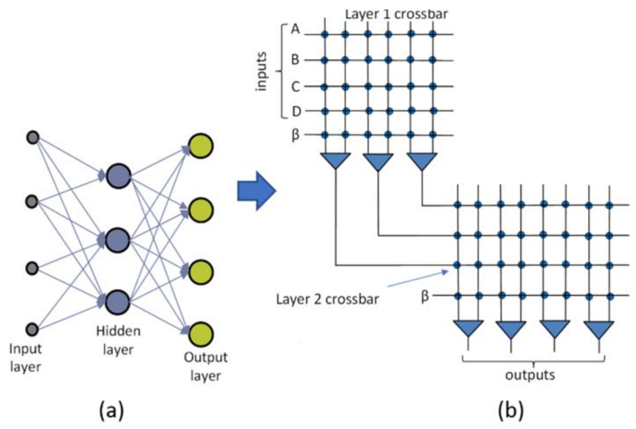

Fig. 5. (a) Two-layer network for learning a four input odd parity function and (b) the memristor crossbar circuit required for implementation.

The advantage of the memristor crossbar-based neuron circuit in Fig. 5 (b) is that all computations related to a single neural network layer can be carried out in one cycle within a crossbar. This includes all multiplications and additions required to generate a set of neuron outputs. Thus, the small size of memristors yields a high compute density on chip. This leads to both area efficiency and high computation throughput. Furthermore, the non-volatile nature of memristors allows circuits to be turned off when not in use, thus reducing static power consumption.

E. *Training Memristor Crossbar Circuits*

The memristor circuit implementation of the training algorithm used in the neural network in Fig. 5 can be broken down into following major steps:

Step 1: Apply inputs to layer 1 and record the layer 2 neuron output errors.

Step 2: Back-propagate layer 2 errors through the second layer weights and record the layer 1 errors.

Step 3: Update the synaptic weights.

The circuit implementation of these steps is detailed below:

Step 1: A set of inputs is applied to the layer 1 neurons, and the layer 2 neuron outputs are measured. Neuron outputs will be obtained after a complete forward pass of the circuit shown in Fig. 5. Errors are discretized into 8 bit representations (with one sign bit and 7 bits for magnitude).

Step 2: The layer 2 errors ($\delta_{L2,1}$ and $\delta_{L2,2}$) are applied to the layer 2 weights after conversion from digital to analog form as shown in Fig. 6 to generate the layer 1 errors ($\delta_{L1,1}$ to $\delta_{L1,6}$). The memristor crossbar in Fig. 6 is essentially the layer 2 crossbar in Fig. 5. Assume that the synaptic weight associated with input i, neuron j (layer 2) is $w_{ij}=\sigma_{ij}^{+} - \sigma_{ij}^{-}$ for $i=1,2,3$ and $j=1,2,..,4$. During the backward pass we evaluate equation (4) to obtain the error values.

$$\begin{aligned}\delta_{L1,i} &= \Sigma_j w_{ij}\,\delta_{L2,j} \quad \text{for } i=1,2,3 \text{ and } j=1,2,..,4. \\ &= \Sigma_j(\sigma_{ij}^{+} - \sigma_{ij}^{-})\delta_{L2,j} \\ &= \Sigma_j\sigma_{ij}^{+}\delta_{L2,j} - \Sigma_j\sigma_{ij}^{-}\delta_{L2,j}\end{aligned} \tag{4}$$

The circuit in Fig. 6 is carrying out equation (4), applying both $\delta_{L2,j}$ and $-\delta_{L2,j}$ to the crossbar columns for $j=1,2,...,4$. The back propagated errors are discretized using ADCs and are stored in buffers. To reduce the training circuit overhead, we can multiplex the back propagated error generation circuit as shown in Fig. 7. In this circuit, enabling the appropriate pass transistors, back propagated errors are sequentially generated and stored in the buffer. Access to the pass transistors will be controlled by a shift register.

Step 3: The weight update procedures for both layers are similar. They take neuron inputs, errors, and intermediate neuron outputs (DP_j) to generate a set of training pulses. To update a synaptic weight by an amount Δw, the conductance of the memristor connected to the first column of the corresponding neuron will be updated by the amount $\Delta w/2$ (where $\Delta w/2 = \eta \times \delta_j \times f'(DP_j) \times x_i$) and the memristor connected to the second column of the corresponding neuron will be updated by the amount $-\Delta w/2$.

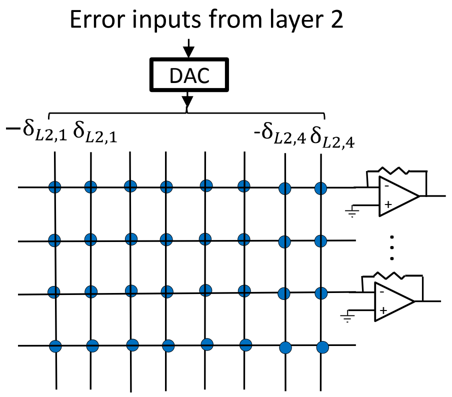
Error inputs from layer 2
DAC
$-\delta_{L2,1}\ \delta_{L2,1}$
$-\delta_{L2,4}\ \delta_{L2,4}$

Fig. 6. Schematic of the neural network for back propagating errors to layer 1.

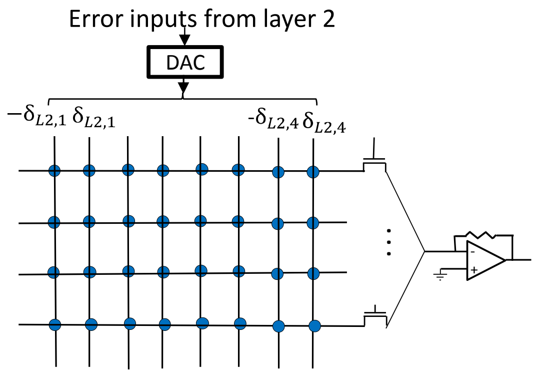
Error inputs from layer 2
DAC
$-\delta_{L2,1}\ \delta_{L2,1}$
$-\delta_{L2,4}\ \delta_{L2,4}$

Fig. 7. Diagram demonstrating the backward pass of the back propagation phase with a multiplexed error generation circuit.

Next, the derivative of the activation function in eq. (2) is evaluated in the memristor hardware to obtain the dot product (DP_j) of the neuron inputs and weights. The DP_j value of neuron j is essentially the scaled difference of the currents through the two columns of a given neuron. The difference between these column currents is stored in a buffer after converting to digital form. The derivative of the activation function (f') is evaluated using a lookup table. A digital multiplier is utilized to multiply the neuron error (δ_j) by $f'(DP_j)$. The value of $\delta_j \times f'(DP_j)$ is converted into analog form and is applied to the training pulse generation unit as in [22]

III. Neural Network to Crossbar Mapping

Given the chosen device properties and our previous studies, we believe that the largest single memristor crossbar that can be patterned for this system should have a size of 256×256 memristors (thus 256 inputs, and 128 output neurons). When a software network layer is too large to fit onto a single crossbar (either because it requires too many inputs, or requires too many neurons), the network layer must be split between multiple crossbars. Fig. 8 demonstrates this process. For instance, let's consider a neural network designed to process the COIL-20 dataset. The network structure is as follows: 1024→512→256→128→20.

Each sample in the dataset requires 1024 inputs, and after three layers of hidden neurons, the network is able to classify data as one of 20 types. Thus, we proposed mapping this neural network according to the layout in Fig. 8. L_n represents layer number of the network and C_n represents crossbar number of the layer. As shown, the weights in the first neural network layer are mapped to four different crossbars. However, in a fully connected network this layer requires 524,288 weights, and these four crossbars can store a maximum of 131,072 weights. Thus, inputs and outputs are locally mapped within crossbars using a reduced number of weights. In L_1 in Fig. 8, inputs 0 through 255 are connected to neurons 0 through 127 using crossbar C_1, inputs 256 through 511 are connected to neurons 128 through 255 using C_2, and so on. This local mapping technique is used in any layer where the required connectivity cannot be placed within a single crossbar [2].

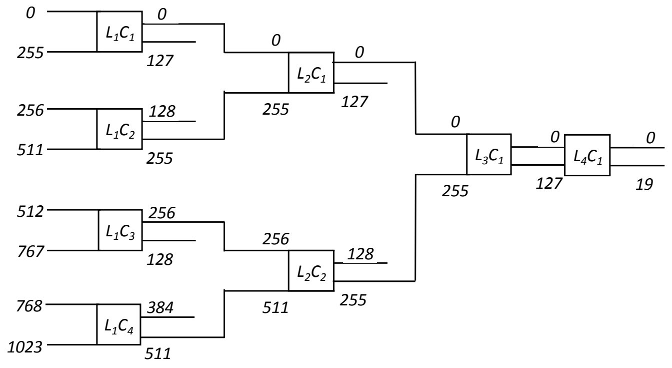

Fig. 8. Crossbar mapping for a neural network designed to learn the COIL-20 dataset.

IV. 3D Memristor Crossbar Architecture

Fig. 9 displays a diagram of the proposed memristor based neural core architecture. In an architecture such as this one, chip area can be quickly consumed by a large number of memristor crossbars, especially as neural network size increases. Thus, we propose an architecture that utilizes vertically stacked memristor crossbars to alleviate area constraints.

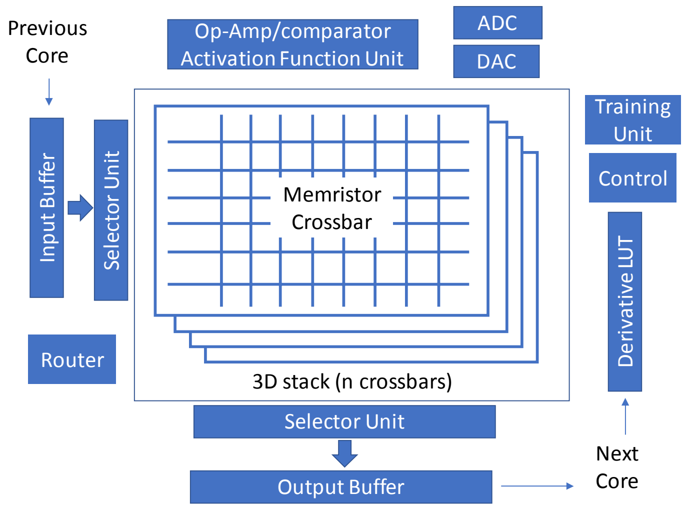

Fig. 9. 3D memristor core architecture.

This architecture is based on a digital core design that was proposed in [2]. The key difference is that analog computation is performed using memristors in place of digital computation components. Due to the compact memristor unit cell and vertically stacked crossbars, chip area in this design can be substantially reduced. Furthermore, the crossbars can be evaluated in both the normal and transposed form (as shown in Figs. 6 and 7), so no eDRAMS are required to store the synaptic weights as in the digital design [2]. Additionally, there are no multipliers required for the forward pass, as multiplication is carried out by the crossbar itself.

Within a neural core, every crossbar in a 3D-stack shares the ADC, DAC, Training Unit, Router, Selector Units (mux and de-mux), and crossbar column amplifiers. A global controller is used to determine the correct crossbar address during computation and communication. Each neural core processes a single but entire crossbar within one computation cycle.

During execution, an input image is loaded into an input buffer, and the Selector Unit determines which crossbar to evaluate in a given core. The input image will pass through a DAC before crossbar evaluation. After crossbar processing, the column amplifier outputs will pass through an ADC.

During training, data is then sent to the learning rule derivative LUT (which is used to compute weight update values when learning is enabled), and this data will be stored in a buffer and routed to the next desired crossbar as needed. Similarly, during the backward pass, a selector unit will determine which crossbar should be selected. The training unit will calculate the error and route it to the previous core and update the existing crossbar weights. It should be noted that in this architecture, routing requires significant processing time.

V. Experimental Setup

In the remainder of this paper, the proposed neural core that utilizes vertical crossbar stacking is examined in terms of time, energy, and area relative to the number of crossbars present in a 3D stack.

A. Application Set

To evaluate this neural core, neural networks were developed to learn and evaluate three standard datasets including MNIST [37], COIL-20 [38], and COIL-100 [39]. For COIL-20 we resized the images to 32×32, and COIL-100 images were resized to 32×32×3.

To examine the processing capability of the neural core in cases of strenuous resource requirements, synthetic datasets were used to execute extremely large neural networks purely from the point of view of resource requirements.

B. Mapping Neural Networks to Crossbars

The neural core is not able to time multiplex neurons as their synaptic weights are stored directly within the neural core crossbars. Hence a neural network's structure may need to be modified to fit into a crossbar. When splitting the weights of a neuron across multiple crossbars, the network needs to be trained with the splitting technique in mind. As the network topology is determined prior to training (based on the neural hardware architecture), neurons split across multiple cores are trained correctly.

C. Area, Power, and Timing Calculations

The area and power of the buffers in the neural cores were calculated using CACTI [40] with the low operating power transistor option utilized. We assumed a 45 nm process. Detailed SPICE simulation was used for power and timing calculations of the analog circuits (including the drivers, crossbars, and activation function circuits). These simulations considered the wire resistance and capacitance within the crossbar as well. Since, the crossbar evaluation time is very small, the majority of time required in the proposed system is spent transferring neuron outputs between cores through the routing network. We assumed that routing would run at 200 MHz. The routing link power was calculated using Orion [41] (assuming 8 bits per link).

D. System Configurations

We assumed 4-bit A/D and D/A converters were used within the cores. Additionally, a 4-bit routing bus for inference (the forward pass) and an 8-bit routing bus for the backward pass were utilized between cores. Our simulations showed that higher precision (8-bit) was required for the backward pass.

VI. Results

This section describes the results obtained when implementing neural networks on the simulated 3D stacked crossbar architecture. In this study, the number of crossbars stacked within each core was varied to examine the impact this design decision has on area, time, and energy consumption. When using the mapping technique described in Section III to port each neural network to the memristor neural cores, only minor reductions in overall accuracy are observed. The left column in Table I displays accuracies assuming the entire network can be ported to memristor crossbars per the method in [42]. The right column in Table I displays the accuracies when using the proposed localized mapping technique.

TABLE I. Accuracy Comparison Between a Fully Connected Neural Network and the Proposed Crossbar Mapping for Three Different Datasets

Dataset	**Accuracy (Fully Connected)**	**Accuracy (Proposed Mapping)**
MNIST	97.65	96.53
Coil-20	97.92	96.79
Coil-100	99.51	97.92

A. COIL-20

The neural network configuration for COIL-20 requires 5 layers, and was constructed as follows: 1,024→512→256→128→20. This neural network requires 8 crossbars, where the number of crossbars required to store the weights between each layer is as follows: 4→2→1→1. This should be interpreted as 4 crossbars required in store the Layer 1 weights, 2 crossbars to hold the Layer 2 weights, 1 crossbar each to hold the weights in Layers 3 and 4. Plots in Fig. 10 show the time, energy, and area required to implement this problem as the number of vertically stacked crossbars in each neural core is increased. Time and energy increase by a small amount until they reach the optimum number of 3D-stacked crossbars, and then they remain constant. On the other hand, area (see Fig. 10 (c)) decreases until the optimal number of crossbars is reached (which is 8), and then area beings to increase again. Area increases as more crossbars are added to the neural core because a larger number of vertically stacked crossbars requires larger multiplexers and control circuitry.

B. *COIL-100*

The neural network configuration for COIL-100 requires six layers and has the following structure: 3,072→1,536→ 768→256→128→100. Crossbars required for each layer are as follows: 12→6→3→1→ 1, a total of 23 memristor crossbars. For the COIL-100 dataset, the results in Fig. 10 follow the same trend as that of the COIL-20 dataset. However, since the neural network associated with this dataset requires 23 crossbars instead of 8, the time, area, and energy required is slightly larger. The optimal number of stacked crossbars is 32 because this is the lowest number larger than the number of crossbars required to implement the network that corresponds to a valid bit width (as 5-bit control circuitry implies indexing 32 crossbars).

C. *MNIST*

The neural network configuration used to learn and evaluate the MNIST dataset was constructed as follows: 784→512→256→10. Using the proposed architecture, 7 crossbars are required to implement this network, where the crossbars required per layer is as follows: 4→2→1. Here 8 is the optimal number of stacked crossbars within a core, as this problem requires at least 7 crossbars to store the network. As the MNIST example requires nearly the same number of crossbars as the COIL-20 example, power, area, and time results are very similar.

D. *Synthetic Data*

In this study, synthetic data was passed though extremely large neural networks to examine what implications this would have on the 3D crossbar neural core architecture. This allows us to test cases where neural network layers must be split across multiple cores, each with a number of vertically stacked crossbars.

1) Synthetic Network 1 has a 9-layer feedforward layout as follows: 8,388,608→4,194,304→2,097,152→1,048,576→ 524,288→262,144→131,072→65,536→32,768 outputs. To implement this network in the proposed architecture, each layer requires the following number of crossbars: 32768→16384→ 8192→ 4096→ 2048→1024→ 512→ 256, for a total of 65,280 crossbars.

2) Synthetic Network 2 contains 5 layers and has the following layer structure: 8,388,608→4,194,304→2,097,152 →1,048,576→524,288. This network requires 61,440 crossbars, and the number of crossbars required for each layer is as follows: 32768→16384→8192→4096.

3) Synthetic Network 3 is a 5-layer network with the following configuration: 524,288→262,144→131,072→ 65536→32768. This network requires 3,480 crossbars to implement, and requires the following number of crossbars per layer: 2,048→1,024→ 512→ 256.

The plots in Fig. 11 show that there is negligible change in time and energy as more stacked crossbars are utilized. Thus, the analog computation of the crossbars and column amplifier

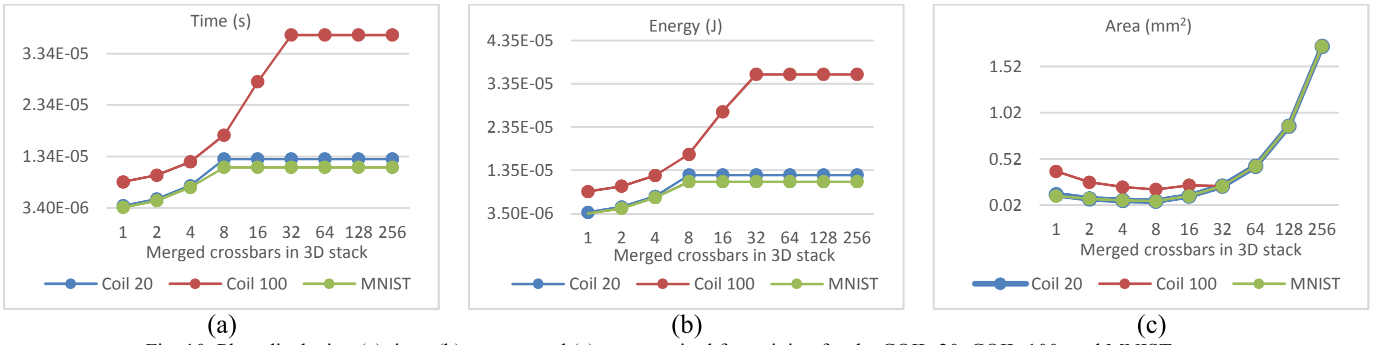

Fig. 10. Plots displaying (a) time, (b) energy, and (c) area required for training for the COIL-20, COIL-100, and MNIST cases.

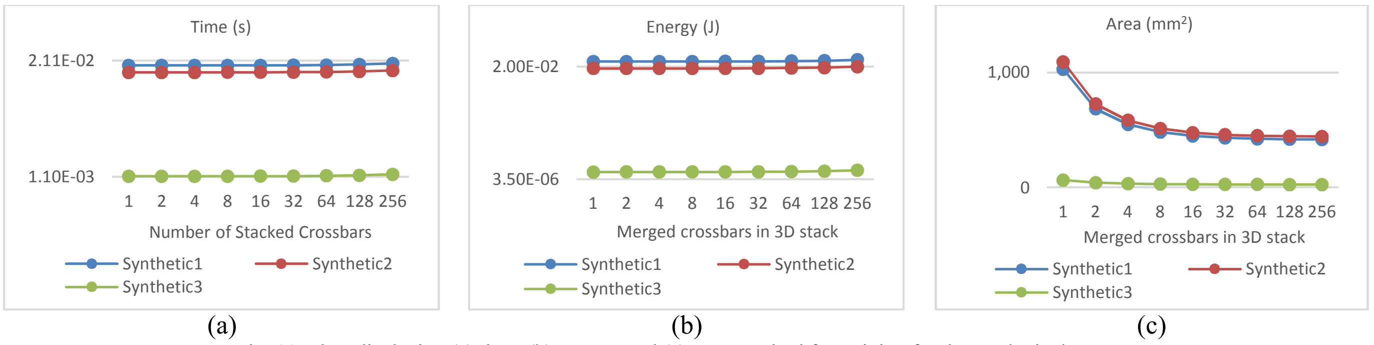

Fig. 11. Plots displaying (a) time, (b) energy, and (c) area required for training for the synthetic data case.

circuits dominate the time and energy expense relative to the routing circuitry. However, significant area savings are observed as these networks are large enough to take advantage of a large number of stacked crossbars. For each of the three synthetic data cases, a 3D crossbar stack is capable of providing an area reduction 2.5× in the proposed neural cores, which are also capable of online training. This area reduction is the result when compared to a comparable architecture incapable of stacking crossbars.

VII. Conclusion

In this work, we describe a multicore neural architecture based on vertically stacked memristor crossbars. The architecture is capable of on-line training, and takes advantage of the memristor crossbar structure to perform rapid, parallel, neural network traversal using analog computation. When porting neural networks to this architecture, a local mapping approach is used to port neural networks to the proposed architecture that reduces the required neuron interconnectivity while maintaining accuracy. A system analysis is performed to determine changes in area, energy, and timing, based on the total number of vertically stacked crossbars within each neural core. In the future we plan to examine the implications of stacking multiple memristor crossbars from a materials point of view, ensuring that electronic characteristics of memristor crossbars remain constant when several are stacked back to back.

VIII. Acknowledge

The authors would like to thank the National Science Foundation and the University of Dayton Graduate School Summer Fellowship program for their support of this work.

References

[1] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. 2014. DaDianNao: A Machine-Learning Supercomputer. In Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-47). IEEE Computer Society, Washington, DC, USA, 609-622.

[2] Y. Qi, R. Hasan, R. Fernando and T. Taha, "Socrates-D: Multicore Architecture for On-Line Learning," 2017 IEEE International Conference on Rebooting Computing (ICRC), Washington, DC, 2017, pp. 1-8.

[3] Kim, D., Kung, J., Chai, S., Yalamanchili, S., & Mukhopadhyay, S. (2016, June). Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory. In Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on (pp. 380-392). IEEE.

[4] Albericio, J., Judd, P., Hetherington, T., Aamodt, T., Jerger, N. E., & Moshovos, A. (2016, June). Cnvlutin: ineffectual-neuron-free deep neural network computing. In Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on (pp. 1-13). IEEE.

[5] Shafiee, Ali, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars." In Proc. ISCA. 2016.

[6] Chi, Ping, Shuangchen Li, Z. Qi, P. Gu, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. "PRIME: A Novel Processing-In-Memory Architecture for Neural Network Computation in ReRAM-based Main Memory." In Proceedings of ISCA, vol. 43. 2016.

[7] Han, Song, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. "EIE: efficient inference engine on compressed deep neural network." arXiv preprint arXiv:1602.01528 (2016).

[8] LiKamWa, Robert, Yunhui Hou, Julian Gao, Mia Polansky, and Lin Zhong. "RedEye: analog ConvNet image sensor architecture for continuous mobile vision." In Proceedings of the 43rd International Symposium on Computer Architecture, pp. 255-266. IEEE Press, 2016.

[9] Reagen, Brandon, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. "Minerva: Enabling low-power, highly-accurate deep neural network accelerators." In Proceedings of the 43rd International Symposium on Computer Architecture, pp. 267-278. IEEE Press, 2016.

[10] Chen, Yu-Hsin, Joel Emer, and Vivienne Sze. "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks." (2016).

[11] Liu, S., Du, Z., Tao, J., Han, D., Luo, T., Xie, Y., & Chen, T. (2016, June). Cambricon: An instruction set architecture for neural networks. In Proceedings of the 43rd International Symposium on Computer Architecture(pp. 393-405). IEEE Press.

[12] Sharma, H., Park, J., Amaro, E., Thwaites, B., Kotha, P., Gupta, A., & Esmaeilzadeh, H. DNNWEAVER: From High-Level Deep Network Models to FPGA Acceleration.

[13] Rhu, M., Gimelshein, N., Clemons, J., Zulfiqar, A., & Keckler, S. W. (2016). Virtualizing Deep Neural Networks for Memory-Efficient Neural Network Design. arXiv preprint arXiv:1602.08124.

[14] Judd, P., Albericio, J., & Moshovos, A. (2016). Stripes: Bit-serial deep neural network computing. IEEE Computer Architecture Letters.

[15] Ji, Y., Zhang, Y., Li, S., Chi, P., Jiang, C., Qu, P., & Chen, W. NEUTRAMS: Neural Network Transformation and Co-design under Neuromorphic Hardware Constraints.

[16] Shi, L., Pei, J., Deng, N., Wang, D., Deng, L., Wang, Y., ... & Zeng, F. (2015, December). Development of a neuromorphic computing system. In 2015 IEEE International Electron Devices Meeting (IEDM) (pp. 4-3). IEEE.

[17] Basterretxea, K., J. M. Tarela, and I. Del Campo. "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons." IEE Proceedings-Circuits, Devices and Systems 151.1 (2004): 18-24.

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

[19] D. Chabi, W. Zhao, D. Querlioz, J.-O. Klein, "Robust Neural Logic Block (NLB) Based on Memristor Crossbar Array" IEEE/ACM International Symposium on Nanoscale Architectures, pp.137-143, 2011.

[20] C. Zamarreño-Ramos, L. A. Camuñas-Mesa, J. A. Pérez-Carrasco, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco, "On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex," Frontiers in Neuroscience, Neuromorphic Engineering, vol. 5, Article 26, pp. 1-22, Mar. 2011.

[21] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing Memristor found," Nature, vol. 453, pp. 80–83 (2008).

[22] Hasan, Raqibul, Tarek M. Taha, and Chris Yakopcic. "On-chip training of memristor based deep neural networks." Neural Networks (IJCNN), 2017 International Joint Conference on. IEEE, 2017.

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

[24] 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.

[25] 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.

[26] Hassan, A. M., Yang, C., Liu, C., Li, H. H., & Chen, Y. (2017, March). Hybrid spiking-based multi-layered self-learning neuromorphic system based on memristor crossbar arrays. In 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 776-781). IEEE.

[27] Burr, G. W., Shelby, R. M., Sebastian, A., Kim, S., Kim, S., Sidler, S., ... & Sanches, L. L. (2017). Neuromorphic computing using non-volatile memory. Advances in Physics: X, 2(1), 89-124.

[28] Negrov, D., Karandashev, I., Shakirov, V., Matveyev, Y., Dunin-Barkowski, W., & Zenkevich, A. (2017). An approximate backpropagation learning rule for memristor based neural networks using synaptic plasticity. Neurocomputing, 237, 193-199.

[29] Nair, M. V., Muller, L. K., & Indiveri, G. (2017). A differential memristive synapse circuit for on-line learning in neuromorphic computing systems. Nano Futures, 1(3), 035003.

[30] T. M. Taha, R. Hasan, C. Yakopcic, and M. R. McLean, "Exploring the Design Space of Specialized Multicore Neural Processors," IEEE International Joint Conference on Neural Networks (IJCNN), 2013.

[31] 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.

[32] Zyarah, A. M., Soures, N., Hays, L., Jacobs-Gedrim, R. B., Agarwal, S., Marinella, M., & Kudithipudi, D. (2017, May). Ziksa: On-chip learning accelerator with memristor crossbars for multilevel neural networks. In Circuits and Systems (ISCAS), 2017 IEEE International Symposium on (pp. 1-4). IEEE.

[33] Rosenthal, E., Greshnikov, S., Soudry, D., & Kvatinsky, S. (2016, May). A fully analog memristor-based neural network with online gradient training. In Circuits and Systems (ISCAS), 2016 IEEE International Symposium on (pp. 1394-1397). IEEE.

[34] Adam, G. C., Hoskins, B. D., Prezioso, M., Merrikh-Bayat, F., Chakrabarti, B., & Strukov, D. B. (2016). 3-D memristor crossbars for analog and neuromorphic computing applications. IEEE Transactions on Electron Devices, 64(1), 312-318.

[35] X. Dong, C. Xu, S. Member, Y. Xie, and N. P. Jouppi, “NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory,” IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, vol. 31, no. 7, pp. 994-1007, July, 2012.

[36] 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).

[37] http://yann.lecun.com/exdb/mnist/

[38] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[39] http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php

[40] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, “Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0” In Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 40), 2007.

[41] A. B. Kahng, B. Li, L. S. Peh, and K. Samadi, “ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration,” Design, Automation & Test in Europe Conference & Exhibition, pp.423-428, 20-24 April 2009.

[42] R. Hasan, T. M. Taha, and C. Yakopcic, “On-chip Training of Memristor Crossbar Based Multi-layer Neural Networks,” Microelectronics Journal, vol. 66, no. 8, pp. 31-40, Aug. 2017.