Simulating and Estimating the Behavior of a Neuromorphic Co-Processor

Catherine D. Schuman, Raphael Pooser, Tiffany Mintz
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
[schumancd,pooserrc,mintztm]@ornl.gov

Md Musabbir Adnan, Garrett S. Rose
University of Tennessee
Knoxville, Tennessee 37996
[madnan,garose]@utk.edu

Bon Woong Ku, Sung Kyu Lim
Georgia Institute of Technology
Atlanta, Georgia
[bwku,limsk]@gatech.edu

ABSTRACT

Neuromorphic computing represents one technology likely to be incorporated into future supercomputers. In this work, we present initial results on a potential neuromorphic co-processor, including a preliminary device design that includes memristors, estimates on energy usage for the co-processor, and performance of an on-line learning mechanism. We also present a high-level co-processor simulator used to estimate the performance of the neuromorphic co-processor on real applications. We discuss future use-cases of a potential neuromorphic co-processor in the supercomputing environment, including as an accelerator for supervised learning and for unsupervised, on-line learning tasks. Finally, we discuss plans for future work.

CCS CONCEPTS

• **Hardware → Neural systems**; *Analysis and design of emerging devices and systems*;

ACM Reference format:
Catherine D. Schuman, Raphael Pooser, Tiffany Mintz, Md Musabbir Adnan, Garrett S. Rose, and Bon Woong Ku, Sung Kyu Lim. 2017. Simulating and Estimating the Behavior of a Neuromorphic Co-Processor. In *Proceedings of PMES'17: Second International Workshop on Post Moore's Era Supercomputing, Denver, CO, USA, November 12–17, 2017 (PMES'17),* 7 pages.
https://doi.org/10.1145/3149526.3149529

1 INTRODUCTION

As we move into post-Moore's law era computing, there are a variety of potential technologies that may be incorporated into supercomputers of the future. We have already seen the emergence of heterogeneity into both supercomputers and clusters through the successful inclusion of graphics processing units (GPUs) into the computational infrastructure of machines such as Oak Ridge Leadership Computing Facility's Titan and the inclusion of programmable architectures such as FPGAs into commercially available clusters such as Amazon Web Services. One of the technologies that is likely to be included in the heterogeneous supercomputing nodes of the future is neuromorphic computing [26]. Neuromorphic computer architecture and operation are inspired by biological brains. Neuromorphic computers are attractive in a post-Moore's law era computing landscape because they offer the potential for lower power, the incorporation of beyond CMOS technologies, and native machine learning capabilities, which are becoming significantly more important in data-heavy supercomputing applications of the future.

When considering the implications of including a neuromorphic computer as a supercomputing co-processor or in a high-performance compute node, several characteristics must be considered. One of those components is the design of the neuromorphic computer system itself, including the selection of circuitry that can perform appropriate operations on a small power budget. Another consideration is how the neuromorphic co-processor will be integrated with a traditional CPU, in terms of a communication infrastructure. Both of these characteristics are primarily hardware-level concerns, but they are inextricably linked to software, algorithms, and applications-level concerns, as we consider the appropriate use cases of the neuromorphic system. In determining the operation of the neuromorphic system, on-chip learning capabilities, how they might be utilized, and how they affect performance are also major considerations.

In this work, we describe a neuromorphic co-processor model, including a hardware implementation based on memristors and an on-chip learning mechanism based on spike-timing dependent plasticity. We present some preliminary work on this project, including results based on low-level and high-level simulations. We also discuss some potential applications for this system, and we speculate on the implications of including such systems in future supercomputers.

2 RELATED WORK

Memristors have become increasingly popular in the neuromorphic computing community for the last decade due to their low energy operation, potential for high density, and their behavioral similarities to biological synapses [17, 19, 23]. Implementations utilizing memristors have been used as both independent synapse implementations [1, 12] and in memristive crossbars [2, 8]. Metal

Notice: This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

PMES'17, November 12–17, 2017, Denver, CO, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-5126-3/17/11…$15.00
https://doi.org/10.1145/3149526.3149529

oxide memristors have been commonly used [25], though there is also exploration of other memristor types, including organic and polymer-based memristors [4, 9]. We estimate the energy calculations in this work based on hafnium-oxide memristors, but any memristor model could be substituted in our simulation framework. Perhaps more importantly, our high-level simulator does not rely on memristor-based operation, so other low-level circuit types could be utilized as well.

High-level simulators of neuromorphic systems have also been developed, such as Xnet [6] and MNSIM [34], both of which are simulators for memristor-based hardware. Perhaps the most related high-level simulator to the one developed in this work is the NeMo simulator [29], which has been specifically used to simulate spiking neural networks of the structure that are used in IBM's TrueNorth neuromorphic system [3]. NeMo also utilizes a parallel discrete event simulator, in particular Rensselaer's Optimistic Simulation System (ROSS) [7]. Our initial high-level simulator implementation does not utilize optimistic execution as part of the parallel discrete event simulation, though it can be extended in the future to do so in order to improve performance as needed.

3 NEUROMORPHIC CO-PROCESSOR SIMULATION DESCRIPTION

A key component to our co-processor co-design is the development of both low-level and high-level simulation models. We have utilized circuit simulators such as Cadence Spectre to define how simple circuits behave, but it is not realistic to utilize circuit-level simulators to estimate the behavior of a device on the scale of a co-processor, which will contain at least hundreds of circuit elements. As such, we have developed a high-level simulator, where behaviors of large systems can be analyzed. The results from the low-level simulator, including energy estimates for different behaviors, can be utilized in the high-level simulator to predict the behavior of a real co-processor. Our high-level simulator is written in C++, and models leaky integrate-and-fire neurons and synapses with spike-timing dependent plasticity (STDP). The activity of a network is simulated using a discrete-event simulation. To allow for large network sizes to be implemented, we have developed a parallel version of our simulator. In this parallel discrete event simulation, a network is divided across multiple compute nodes, and events are communicated between nodes using the message passing interface (MPI). We have created our own neuromorphic simulator so that we may easily include different hardware characteristics, restrictions, and performance estimates in the future, as we plan to use the same simulator for other hardware implementations. For example, although we currently utilize a metal oxide memristor-based synapse, we may explore other synapse and neuron implementations in the future, such as optical neurons or polymer or organic memristors.

One of the key characteristics of neuromorphic computers is their potential for on-line, on-chip learning. Though there are neuromorphic systems that do not implement on-chip learning, including TrueNorth [3], we believe that it is highly important that on-chip learning be included as part of a neuromorphic system. Moreover, we also believe that it is important to consider training and/or learning as part of the overall estimates of performance of neuromorphic systems. In most reported results of neuromorphic computers, performance estimates are given for a pre-trained network or for the inference step only. Since training and learning can be of considerable computational cost, we want to be able to quantify how they affect performance of a neuromorphic co-processor. As such, we include the ability to collect performance results during both our training and learning steps.

4 PRELIMINARY RESULTS

In this section, we present some preliminary results of our investigation of a neuromorphic co-processor. We first present an initial exploration of chip layout for a neuromorphic coprocessor. Then, we discuss utilizing a low-level simulation to obtain estimates on energy required for completing certain operations. We briefly discuss the implementation of an on-chip learning mechanism based on spike-timing dependent plasticity (STDP) and discuss the implications of choosing different clock rates for the co-processor on the STDP process. We then present results on the scalability of a high-level simulator and give a brief discussion of how we can integrate results from the low-level and high-level simulators to estimate performance of the system.

4.1 Low-Level Design Exploration

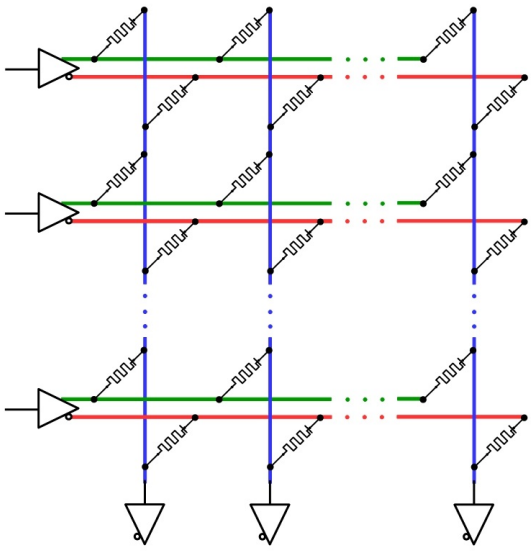


Figure 1: Neurons and synapses laid out in crossbar fashion.

An overview of the memristive crossbar-based neuromorphic fabric considered here is illustrated in Figure 1. The key advantage of using a crossbar implementation is the high density achieved by implementing synapses as memristors that exist at each cross-point. The crossbar also acts as an analog matrix-vector multiplier, a common operation for neuromorphic algorithms, but with dramatically reduced hardware relative to CMOS counterparts [16]. This usage of a memristive crossbar also helps reduce the overall energy-consumption of the neuromorphic co-processor. For the architecture considered, twin memristors are used as synapses and

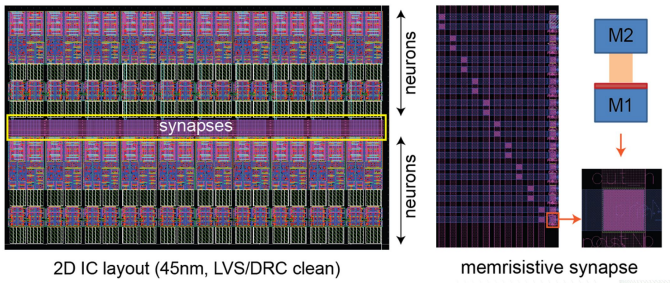


Figure 2: Full-chip layout of our 10x10 memristive neural network design.

are driven by complimentary signals from the pre-neuron to realize both positive and negative weights. Spikes driven through the twin memristor synapses are multiplied by the weights (memristor conductance) via Ohm's Law with the weighted inputs summed along the columns via Kirchoff's Law. The post-neurons (bottom of Figure 1) are then responsible for integrating this resulting sum and comparing against a threshold to determine if a firing condition has been met.

We have explored an initial chip layout for the neuromorphic co-processor to specify the hardware design constraints at early stage. Figure 2 shows the full-chip layout of our 10x10 memristive neural network. We have utilized NCSU FreePDK45 (45nm Process Design Kit) [13] and added the design rules for the memristor and analog components including a Metal-Insulator-Metal capacitor to guarantee the functionality and manufacturability of the layout. The memristor-based synapses ($< 1\mu m^2$) are very small relative to individual neurons ($\sim 100\mu m^2$), leading to very high density connections between pre-neurons and post-neurons. Thanks to the high fabrication density of memristors, this crossbar architecture not only reduces the entire footprint of neuromorphic co-processor, but also provides layout regularity, which aids the scalability of the overall network.

4.2 Low-Level Energy Estimates

We have executed low-level simulations of neuromorphic circuits to determine the performance of a neuromorphic co-processor using Cadence Spectre. These simulations allow us to evaluate simple circuits and simple operations. One factor we can measure using this simulation is the energy consumption of a circuit. For example, there are three major operations that make up the behavior of our proposed neuromorphic fabric: accumulation, training, and idle. The accumulation phase is when a spike traveling along the synapse has reached the post-neuron and its weight is being added to the post-neuron's charge. The training phase occurs during the STDP phase (see Section 4.3 for more detail).

Energy requirements for our implementation for the different operating phases are presented in Table 1. The values listed here account for the memristive synapse as well as the synaptic control circuit of the pre-neuron. While our implementation is capable of tracking 5 clock cycles before and after a post-neuron fire for STDP purposes, we could restrict our circuit to track fewer cycles. Correspondingly, we have shown the energy requirements when we restrain our circuit to 3 cycle or 1 cycle tracking. As expected, there is a reduced energy requirement if we track for less number of cycles. It should also be noted that energy calculations are based on memristance range of 5kΩ to 50kΩ with a clock frequency of 25MHz. While these values have been assumed conservatively for the device mentioned in [5], with a higher clock frequency and higher value of memristance, energy requirements will be even less.

Table 1: Energy per spike of a neuromorphic system including the proposed synapse

Tracking	Accumulation (pJ)	Training (pJ)	Idle (pJ)
1 cycle	1.25	2.42	0.07
3 cycle	1.32	3.11	0.61
5 cycle	1.48	3.36	0.69

4.3 STDP Results

One of the key components of neuromorphic architectures is their ability to do on-line, on-chip learning. This is commonly implemented with spike-timing dependent plasticity (STDP). Different hardware characteristics can result in different behavior of STDP. In this section, we demonstrate that our proposed circuit can accomplish STDP, and we also show the effect of different clock rates on how STDP performs.

A simulation result from Cadence Spectre on a simple neuron-synapse-neuron circuit is shown in Figure 3 to demonstrate how potentiation and depression work. Different levels of potentiation and depression conditions are generated in the simulation. Assuming the synapse is excitatory, the initial weight of the synapse is positive. In the beginning, the pre-neuron and post-neuron fires are farther apart in time. Hence, the potentiation and depression is less, as it can be seen from G_{eff} waveform. When the fires are closer, potentiation and depression is larger. For the pre-neuron fires that occur immediately before (after) the post-neuron fire, potentiation (depression) is the strongest. For that case, G_{eff} rises and falls very sharply. In Figure 3, the synapse is first depressed and then potentiated so that depressions always begin from the same initial positive weight.

Using the conductance values from Spectre simulation, exponential STDP behavior can be seen in Figure 4. The graph shows the change of conductance as a percentage of maximum conductance (G_{max}) versus timing difference between pre-neuron and post-neuron fires.Using the same memristor model in MATLAB, a similar graph can be generated. This MATLAB model closely approximates the simulation data but slightly overestimates the conductance change. This is a consequence of the voltage drop across the transistors in the Cadence circuit level implementation that is not present in the MATLAB model.

The change in synaptic weight is directly related to the switching time of memristors. Also, the change in memristance is proportional to the amount of time for which voltage across the memristor exceeds the switching threshold. Hence, the clock frequency has significant impact on STDP behavior, as the pulse width defines

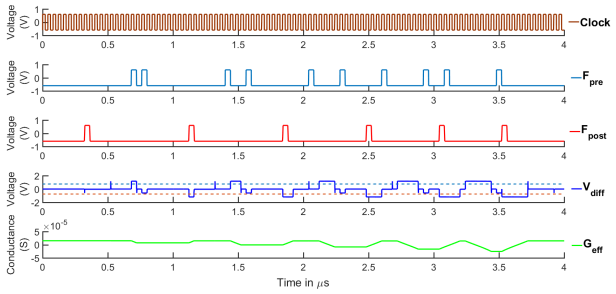


Figure 3: Waveforms depicting the weight change of the synapse in accordance with the STDP rule

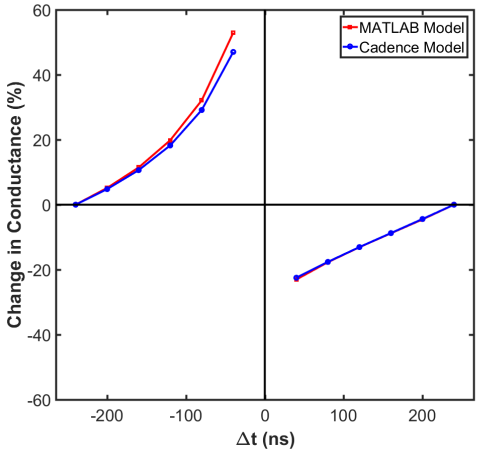


Figure 4: Dependence of the STDP behavior of the proposed synapse on the clock frequency

how long the synapse is depressed or potentiated. In Figures 3 and 4, the conductance change is achieved with a clock frequency of 25 MHz. If we increase the clock frequency to 100 MHz, we get smaller changes in memristance since the pulse widths are smaller. The small changes in memristances cause small change in synapse conductance. Also, as we are tracking the same number of firing events for a higher clock frequency, we are tracking for less amount of time as compared to that for a lower frequency.

We utilize a combination of these results from the low-level simulator and our high-level simulator to study whether we need smaller changes in memristance to accomplish desired tasks. In general, higher clock rates will correspond to greater power consumption. Different applications may have different requirements in terms of power restrictions. Using a combination of our high and low level simulations, one can study the effect of the changes in STDP on performance and determine the appropriate clock rate for their performance needs.

4.4 High-Level Simulator Results

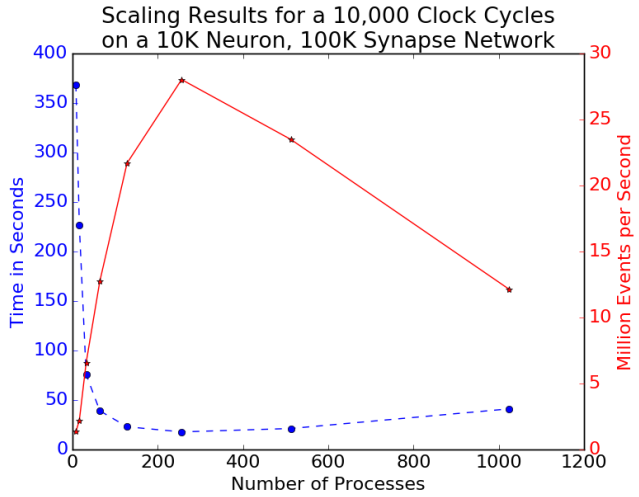
The high-level simulator for our project is written in C++ to allow for speed of simulation, and we utilize MPI to enable communication between compute nodes running in parallel. The simulation is implemented as a discrete event simulation, with the distributed version employing a simple, conservative synchronized parallel discrete event simulation algorithm. A high-level simulator is important for estimating the behavior of a co-processor for relatively large networks (greater than 100 neurons) in a reasonable amount of time. For example, our low-level simulations in Sections 4.2 and 4.3 require approximately three hours to simulate 320 cycles on a network with nine neurons and seven synapses. Using the high-level simulator, we can simulate the behavior of large networks (greater than 10,000 neurons and and greater than 100,000 synapses) in a relatively short amount of time (less than ten minutes for a 10,000 cycle simulation).

For larger network sizes and longer simulation runs, it is beneficial to utilize our distributed neuromorphic network simulation. Figures 5a and 5b shows the time (in seconds) to simulate a 10,000 neuron, 100,000 synapses network and a 100,000 neuron, 1,000,000 synapses network (respectively) for 10,000 clock cycles on the supercomputer Titan[2] at Oak Ridge National Laboratory. As can be seen in the figure, increasing the number of processes (each of which is assigned to its own core on Titan) improves performance to a point. The best results for the smaller network size (results shown in Figure 5a) is achieved using 256 cores, with a total processing of over 28 million events per second. The performance decreases for 512 and 1024 cores, indicating that there are communication bounds for a network of that size and connectivity level. With the larger network size (results shown in Figure 5b), the best performance is achieved at 512 cores, indicating that as the network size increases, it may be beneficial to increase the number of cores. We have not yet explored different connectivity levels, though we intend to pursue that in future work.

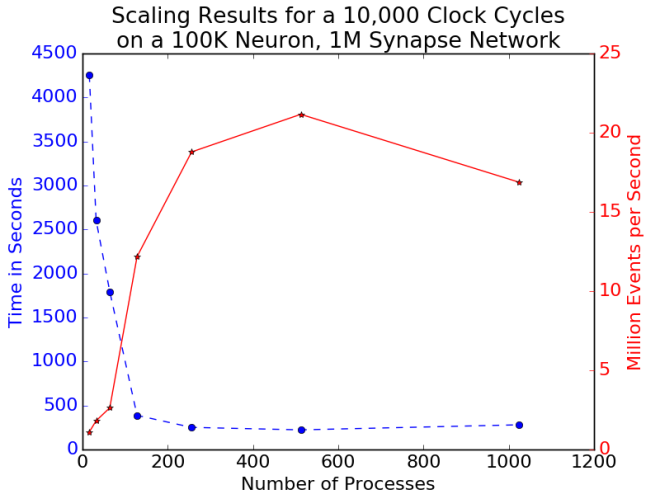
The events processed here are the accumulation events that occur at a neuron when one of its incoming synapses is activated and fire events. We do not include adjustments of synaptic weights due to STDP as types of events, but it is worth noting that, on average, synaptic depression occurs on every accumulation event on a neuron for all incoming synapses on that neuron, and synaptic potentiation occurs for every fire on a neuron for all incoming synapses for that neuron. If the event calculation metric includes those types of events, then the number of events processed per second for this network are multiplied by approximately 10 (because of the 10:1 synapse-to-neuron ratio), to processing 280 million events per second for the 10,000 neuron, 100,000 synapse network.

Utilizing this simulation in the future, we will be able to map our more detailed events measurements to the energy estimates discussed in Section 4.2 to estimate the overall energy or power for different applications. Additionally, this simulation will also allow us to explore what effect different hardware decisions has on performance for real applications. For example, the length of the learning cycle training (as discussed in Table 1) is on such feature where we can evaluate how the length of the learning cycle actually affects learning on a real application, both in terms of the network's

[2]Titan: https://www.olcf.ornl.gov/computing-resources/titan-cray-xk7/



(a) Scaling results for a 10,000 neuron, 100,000 synapse network for 10,000 clock cycles.



(b) Scaling results for a 100,000 neuron, 10,00,000 synapse network for 10,000 clock cycles.

Figure 5: Dependence of scaling results for our parallel discrete event simulator on network size. The two y-axes show the time in seconds to complete the simulation (on the left, in blue) and million of events processed per second (on the right, in red).

ability to learn and on the resulting energy usage. Similarly, we can evaluate the different STDP behavior described in Figure 4 in terms of both effect on learning ability of networks and on speed of performance of a real chip (for different clock rates), as there may be applications that require higher clock rates but less fidelity in the learning itself (or vice versa). On the whole, the combination of low- and high-level simulators allow us to explore implications of hardware decisions in a real way, rather than relying on either low-level estimations alone (which are only feasible for trivial networks) or on high-level estimations alone (which do not provide the level of detail required for some performance estimations).

5 FUTURE USE CASES

There are a variety of potential applications of a neuromorphic co-processor within a real high performance computing (HPC) or supercomputing environment. Most of these applications have to do with intelligent data processing. One potential clear application of spiking neuromorphic co-processors is for deploying spiking neural networks on supervised data classification tasks. Spiking neural networks have been trained for a variety of classification tasks, including image classification [11, 14], speech classification [28], and numerical data classification [33]. One of the main uses of supercomputing systems is for modeling and simulation of various scientific phenomena [27]. These modeling and simulation tasks typically generate a large amount of data that may need to be analyzed and/or classified [30]. Pre-trained neuromorphic co-processors on HPC compute nodes alongside the architectures that are completing the modeling and simulation tasks and generating data, could analyze the data on the compute node itself, reducing data transfer and storage costs, and providing scientists with real-time results for their modeling and simulation experiments.

Depending on the training method utilized for the supervised classification task, we may also be able to utilize the neuromorphic co-processor as part of the training process (without changing the underlying architecture). For example, our neuromorphic co-processor implements spike-timing dependent plasticity as an on-chip learning mechanism, which can be utilized as a training mechanism. Training mechanisms that require testing an instance of a neuromorphic system on a particular task, including evolutionary/genetic algorithms [21, 33], can also utilize the co-processor during the training step. For networks trained utilizing an algorithm such as back-propagation, the "forward-pass" of the algorithm can also be executed on the chip itself. As such neuromorphic co-processors are also well-suited for completing or aiding in completing machine learning-style tasks on supercomputers.

An increasingly specific use of spiking neuromorphic systems is as the reservoir in reservoir computing applications [24], such as liquid state machines. Liquid state machines have been shown to be useful in pattern recognition tasks, including speech recognition [32] and other temporal and spatiotemporal recognition tasks [15]. Spiking neuromorphic networks, including those that utilize memristors in their implementation, have been shown to perform well as reservoirs in liquid state machine applications.

Much of the data generated by scientific simulations falls into the category of temporal or spatiotemporal. State-of-the-art machine learning methods such as convolutional neural networks have been shown to perform well on spatial data, but the results on temporal data are relatively limited, though there is significant work in this area including three-dimensional convolutional neural networks [18] and long-short term memory networks [31]. Because of their native temporal processing power, spiking neuromorphic systems can be utilized as part of an overall machine learning co-processor structure to help deal with temporal data collection and analysis, perhaps alongside a custom convolutional neural network processor, like Google's Tensor Processing Unit [20] or graphics

processing units (GPUs) that have been optimized for convolutional neural networks [22].

A key capability of many spiking neuromorphic systems, including the co-processor structure presented here, is the ability to do on-line, on-chip learning. This is especially useful in a data-rich, label-poor environment, in which unsupervised machine learning methods are most useful. Spiking neuromorphic systems with STDP or STDP-like learning mechanisms have already been shown to have on-line, unsupervised capabilities in terms of applications such as data clustering [10]. We expect that the full unsupervised learning capabilities of spiking systems are still unknown, but that they will continue to improve as spiking neuromorphic systems become increasingly available for more general use through the development of simulators and hardware as well as the associated software environments for utilizing them.

Similarly, we expect that there are non-neural network applications that will be able to utilize the architectural and computational characteristics of spiking neuromorphic systems. One example for non-neural network applications for neuromorphic systems is to implement graph algorithms. Since networks (i.e., graphs of nodes and edges) underly the architectural description of spiking neuromorphic systems, there are likely creative ways to map graph analysis on those systems. Once again, as spiking neuromorphic systems and their associated software environments (such as the ones described herein) become increasingly available, we expect that more and more users will develop custom applications for those systems, including both neural network and non-neural network applications.

6 DISCUSSION AND FUTURE WORK

We believe that 3D IC will be a key technology to support communications between our neuromorphic co-processor and its von Neumann host. The currently popular through-silicon-via (TSV) or the newly emerging monolithic inter-tier via (MIV) technologies offer massive connections between the tiers that can be exploited for ultra-high bandwidth and low energy communication. We are currently working to build models and circuit elements to support inter-tier communication in a 3D IC, where a neuromorphic learning chip is stacked with a von Neumann multi-core chip. These models and designs along with their simulation results will be used in our high-level neural network simulator to obtain accurate energy and bandwidth benefits in large-scale neural networks described in Section 5.

For our high-level simulator, we plan to explore optimistic event execution models as part of our simulation framework to improve the overall performance of the simulator. We also intend to continue to improve portability for the system, so that new low-level energy estimates can be easily evaluated in the structure. As part of that evaluation, we will explore the incorporation of other memristor types, such as spin-based memristors or organic memristors. We also tend to investigate at least one implementation that does not utilize memristors in our evaluation framework in order to compare performance.

In Section 5, we described a variety of future use cases of a neuromorphic co-processor. We are in the process of implementing a reservoir computing application for our neuromorphic co-processor, and we intend to explore both supervised and unsupervised learning approaches within our low- and high-level simulation frameworks, specifically as applied to time-series or other temporal data. We plan to coordinate with computational scientists who utilize modeling and simulation code bases on existing supercomputers such as Titan and explore how neuromorphic systems may be utilized in analyzing data generated by those systems.

7 CONCLUSION

In this work, we presented a potential neuromorphic co-processor design and discuss some results associated with high- and low-level simulations of that system. We speculate on potential future use-cases for such a co-processor. It is clear that future computers will have increasingly heterogeneous architectures. Based on the results presented herein and our continued work, we believe that neuromorphic systems are not only viable as co-processors, but they exhibit a variety of characteristics that make them suitable for solving real tasks that exist in today's systems and will exist in future computing systems as well.

ACKNOWLEDGEMENTS

Research sponsored in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy.

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

REFERENCES

[1] Shyam Prasad Adhikari, Hyongsuk Kim, Ram Kaji Budhathoki, Changju Yang, and Leon O Chua. 2015. A circuit-based learning architecture for multilayer neural networks with memristor bridge synapses. *IEEE Transactions on Circuits and Systems I: Regular Papers* 62, 1 (2015), 215–223.

[2] A Afifi, A Ayatollahi, and F Raissi. 2009. Implementation of biologically plausible spiking neural network models on the memristor crossbar-based CMOS/nano circuits. In *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*. IEEE, 563–566.

[3] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. 2015. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 10 (2015), 1537–1557.

[4] Fabien Alibart, Stéphane Pleutin, Olivier Bichler, Christian Gamrat, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Dominique Vuillaume. 2012. A memristive nanoparticle/organic hybrid synapstor for neuroinspired computing. *Advanced Functional Materials* 22, 3 (2012), 609–616.

[5] Karsten Beckmann, Josh Holt, Harika Manem, Joseph Van Nostrand, and Nathaniel C Cady. 2016. Nanoscale Hafnium Oxide RRAM Devices Exhibit Pulse Dependent Behavior and Multi-level Resistance Capability. *MRS Advances* (2016), 1–6.

[6] Olivier Bichler, David Roclin, Christian Gamrat, and Damien Querlioz. 2013. Design exploration methodology for memristor-based spiking neuromorphic architectures with the Xnet event-driven simulator. In *Nanoscale Architectures (NANOARCH), 2013 IEEE/ACM International Symposium on*. IEEE, 7–12.

[7] Christopher D Carothers, Kalyan S Perumalla, and Richard M Fujimoto. 1999. Efficient optimistic parallel simulations using reverse computation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 9, 3 (1999), 224–253.

[8] Djaafar Chabi, Weisheng Zhao, Damien Querlioz, and Jacques-Olivier Klein. 2011. Robust neural logic block (NLB) based on memristor crossbar array. In *Proceedings of the 2011 IEEE/ACM International Symposium on Nanoscale Architectures*. IEEE Computer Society, 137–143.

[9] Yu Chen, Gang Liu, Cheng Wang, Wenbin Zhang, Run-Wei Li, and Luxing Wang. 2014. Polymer memristor for information storage and neuromorphic applications. *Materials Horizons* 1, 5 (2014), 489–506.

[10] Peter U Diehl and Matthew Cook. 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience* 9 (2015).
[11] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 1–8.
[12] Chao Du, Wen Ma, Ting Chang, Patrick Sheridan, and Wei D Lu. 2015. Biorealistic implementation of synaptic functions with oxide memristors through internal ionic dynamics. *Advanced Functional Materials* 25, 27 (2015), 4290–4299.
[13] NCSU EDA. 2011. NCSU FreePDK45. (2011). http://www.eda.ncsu.edu/wiki/FreePDK:Contents
[14] Steve K Esser, Rathinakumar Appuswamy, Paul Merolla, John V Arthur, and Dharmendra S Modha. 2015. Backpropagation for energy-efficient neuromorphic computing. In *Advances in Neural Information Processing Systems*. 1117–1125.
[15] Eric Goodman and Dan Ventura. 2006. Spatiotemporal pattern recognition via liquid state machines. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*. IEEE, 3848–3853.
[16] Miao Hu, Hai Li, Yiran Chen, Qing Wu, Garrett S. Rose, and Richard W. Linderman. 2014. Memristor Crossbar-Based Neuromorphic Computing System: A Case Study. *IEEE Transactions on Neural Networks and Learning Systems* 25, 10 (2014), 1864–1878.
[17] Giacomo Indiveri, Bernabé Linares-Barranco, Robert Legenstein, George Deligeorgis, and Themistoklis Prodromakis. 2013. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology* 24, 38 (2013), 384010.
[18] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2013), 221–231.
[19] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B Bhadviya, Pinaki Mazumder, and Wei Lu. 2010. Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters* 10, 4 (2010), 1297–1301.
[20] Norm Jouppi. 2016. Google supercharges machine learning tasks with TPU custom chip. *See https://cloudplatform. googleblog. com/2016/05/Google-supercharges-machine-learning-tasks-with-custom-chip. html* (2016).
[21] Nikola Kasabov, Nathan Matthew Scott, Enmei Tu, Stefan Marks, Neelava Sengupta, Elisa Capecci, Muhaini Othman, Maryam Gholami Doborjeh, Norhanifah Murli, Reggio Hartono, et al. 2016. Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: design methodology and selected applications. *Neural Networks* 78 (2016), 1–14.
[22] Sik Kim and Yongjin Kwon. 2017. Unified Platform for AI and Big Data Analytics. *Journal of Computer and Communications* 5, 08 (2017), 1.
[23] Bernabé Linares-Barranco and Teresa Serrano-Gotarredona. 2009. Memristance can explain spike-time-dependent-plasticity in neural synapses. (2009).
[24] Mantas Lukoševičius, Herbert Jaeger, and Benjamin Schrauwen. 2012. Reservoir computing trends. *KI-Künstliche Intelligenz* (2012), 1–7.
[25] Baker Mohammad, Maguy Abi Jaoude, Vikas Kumar, Dirar Mohammad Al Homouz, Heba Abu Nahla, Mahmoud Al-Qutayri, and Nicolas Christoforou. 2016. State of the art of metal oxide memristor devices. *Nanotechnology Reviews* 5, 3 (2016), 311–329.
[26] Don Monroe. 2014. Neuromorphic computing gets ready for the (really) big time. *Commun. ACM* 57, 6 (2014), 13–15.
[27] Tim Palmer. 2015. Build imprecise supercomputers: energy-optimized hybrid computers with a range of processor accuracies will advance modelling in fields from climate change to neuroscience. *Nature* 526, 7571 (2015), 32–34.
[28] S Park, A Sheri, J Kim, J Noh, J Jang, M Jeon, B Lee, BR Lee, BH Lee, and H Hwang. 2013. Neuromorphic speech systems using advanced ReRAM-based synapse. In *Electron Devices Meeting (IEDM), 2013 IEEE International*. IEEE, 25–6.
[29] Mark Plagge, Christopher D Carothers, and Elsa Gonsiorowski. 2016. NeMo: A Massively Parallel Discrete-Event Simulation Model for Neuromorphic Architectures. In *Proceedings of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation*. ACM, 233–244.
[30] Daniel A Reed and Jack Dongarra. 2015. Exascale computing and big data. *Commun. ACM* 58, 7 (2015), 56–68.
[31] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 4580–4584.
[32] Benjamin Schrauwen, Michiel DâĂŹHaene, David Verstraeten, and Jan Van Campenhout. 2008. Compact hardware liquid state machines on FPGA for real-time speech recognition. *Neural networks* 21, 2 (2008), 511–523.
[33] Catherine D Schuman, James S Plank, Adam Disney, and John Reynolds. 2016. An evolutionary optimization framework for neural networks and neuromorphic architectures. In *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 145–154.
[34] Lixue Xia, Boxun Li, Tianqi Tang, Peng Gu, Xiling Yin, Wenqin Huangfu, Pai-Yu Chen, Shimeng Yu, Yu Cao, Yu Wang, et al. 2016. MNSIM: Simulation platform for memristor-based neuromorphic computing system. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*. IEEE, 469–474.