SAND2019-12139C

Exploring Spiking Neural Networks on Crossbars: Co-Design using Whetstone and CrossSim

Christopher Bennett*, Ryan Dellana*, Suma G. Cardwell, William Severa, James B. Aimone
Center for Computing Research, Sandia National Laboratories
Albuquerque, New Mexico, USA
Email: cbennet,rdellan,sgcardw,wmsever,jbaimon@sandia.gov

***Abstract*—Spiking Neural Networks with event-driven processing and energy gains are promising candidates to implement deep neural networks. Analog resistive memory crossbars (ReRAM) have been shown to provide computational efficiency gains in building compact and efficient neural accelerators. In this paper, we leverage high density non-volatile analog memories with spike based digital communication, for efficient neuromorphic processing. We publish preliminary results by combining two tools Whetstone and CrossSim to evaluate this hypothesis. Whetstone is a tool that converts deep neural networks to spiking neural networks. CrossSim is a crossbar simulation tool that we will use to simulate analog Resistive memory crossbars(ReRAM) using Whetstone trained weights. Based on our preliminary results, we find that making Whetstone generated network noise-aware gives significant performance gains for the crossbar inference engine. We also briefly introduce *Mosaics*, a simple yet powerful concept for extending neuromorphic computing, to be more flexible and enable both robust learning and powerful programming. Building this co-design framework is a step towards that direction.**

I. Introduction

Neuromorphic computing is an increasingly attractive alternative as Moore's law slows down. Spiking Neural Networks (SNN) touted as the third generation of neural networks [1], are inspired by the brain with low power computation, and even-driven processing as key properties. Compared to SRAM-based accelerators using analog crossbars to build neural accelerator show energy, area and latency gains [2]. Our driving hypothesis is that a combined platform could amplify the advantages of both high-density analog memory and spike-based digital communication in a neuromorphic architecture, while mitigating each of the other approaches' limitations. The proposed co-design approach will maximize the benefits of these emerging complementary technologies, ideally charting a path to widespread applicability of brain-inspired computing from embedded domains to large-scale simulation tasks. It will also afford the user more flexibility to co-design both the algorithm and crossbar architecture.

In section II, we review emerging memory technologies and challenges associated with using ReRAM crossbars. In section III, we detail the co-design framework to combine the benefits from rapid advances in both high-density analog memories and spiking neuromorphic architectures. In section IV, we discuss the results of using Whetstone generated nets on ReRAM crossbar simulation for MNIST using CrossSim and discuss how performance was improved by our co-design approach. In section V, introduce the *Mosaics* concept for resource reuse, extracting more out of neuromorphic architectures than would be otherwise expected.

*The two authors contributed equally to this paper.

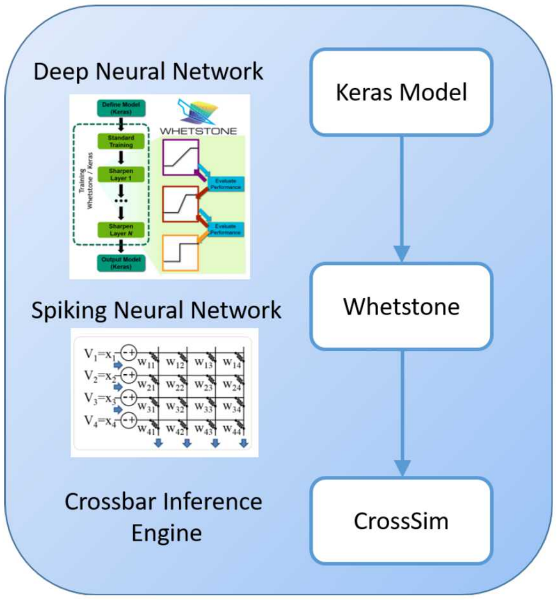

Fig. 1. Combining Whetstone and CrossSim to generate and evaluate a Crossbar inference engine. Some part of image reproduced from [3], [4]

II. Emerging Memory Technologies

Most neural computations and neuromorphic architectures rely on the combination of synaptic memories (representing weights in a neural network) and arithmetic circuitry to implement neuron-based computation. As illustrated in Fig. 4, for a number of applications (such as deep artificial neural networks (ANNs)), the overall size of an algorithm may be very large, but parts of the necessary computation are repeated. There are many existing proposals to implement next-generation hardware learning systems using emerging non-volatile memory (NVM) devices such as resistive/filamentary ReRAM, phase-change memory (PCM) or magnetic memory devices such as spin-transfer-torque magnetic tunnel junctions (STT-MTJs). These approaches typically focus on fully-connected-networks (FCNs) in the context of industrially-proven algorithms being used in deep/machine learning, such as gradient descent with backpropagation (GD-BP) through layers. However, emerging memory devices are intrinsically stochastic due to various atomic, fabrication, and electrical effects, creating non-standard weight updates between devices and between write cycles, and in the case of analog devices (e.g. ReRAM , PCM), intrinsically non-linear updates, meaning that weight updates are heavily dependent on the starting conductance/state. In contrast, systems learning with GD-BP have a stringent requirement of uniform, highly linear devices/weights. Lastly, while the arrays of devices themselves are extremely compact, companion analog to digital and digital to analog circuitry at the array periphery remains a hurdle to realizing energy and speed requirements [2]. Our approach alleviates some of these by including noise in the training of our spiking neural nets, treating it as an inherent property of the system.

Electrically integrating NVM devices in dense arrays, where they can be most effectively used for time multiplexing, nevertheless involves some practical circuit design challenges. Traditionally, variability in off/HRS and on/LRS states caused severe difficulty in correctly reading to and writing from dense arrays with these devices within a set sensing margin [5]. In addition, selecting a subset of devices to write/adjust can create sneak-path issues, which can be partially but not entirely dealt with by enhancing nonlinearity of constituent memory devices [6]. However, both of these issues have since been mostly mitigated by the development of highly non-linear selector devices [7], [8] , which allow for efficient reads from dense memory arrays as well as the implementation of vector matrix multiplies (VMMs) for neuromorphic applications [9] . These challenges are somewhat simplified in the neuromorphic inference-only application, where imported weights from another computing system are used to accelerate predictions on known task. Nevertheless, electrical issues such as parasitic voltage drops, parasitic resistances, and increasing current draws may limit the ultimate size and accuracy of inference-only neuromorphic accelerators [10], [11]. In addition, cycle-to-cycle read noise will be an inherent part of such systems, and we aim to show that it can be made manageable. Many machine learning accelerators have been proposed that leverage memristor crossbars like PUMA [12], ISAAC, [13]. Surveys on using memory based technology for accelerators are given by [14] and neuromorphic computing using NVMs by [15]. The key difference between those approaches and ours is the use of Whetstone as our training engine, which utilizes existing machine learning frameworks to generate the spiking neural network. This approach is not tied to any spike encoding scheme.

III. Algorithm-Hardware Co-Design Tool Framework

Traditionally, hardware architectures are an afterthought for algorithm design. However, to maximize performance gains from hardware, both algorithm and hardware architecture need to be optimized. Our goal is to scope and develop a process flow that is a more top-down approach to designing hardware architectures, optimized to spiking algorithms. Our co-design framework coordinates between the algorithm and circuit scales and is critical for efficiently and rapidly producing neuromorphic architectures well-suited for tailoring architectures for specific application domains.

As an example of this approach we combine two powerful tools, Whetstone[3] and CrossSim[4] to study architecture trade-offs for analog ReRAM models using a Whetstone to train the spiking neural network as shown in Fig. 3. This allows for co-design of neural algorithms with hardware, using tools such as Whetstone, which are powerful for hardware-agnostic neural algorithm development, and tools such as CrossSim, which are well-suited for evaluating a specific algorithm/architecture combination.

A. Algorithm Design

Whetsone [3] is a tool for training deep spiking neural networks, i.e. deep neural networks where the activation functions have discrete, binary communication. During the training process, the activation function at each layer of the network is progressively sharpened towards a threshold activation, with limited loss in performance. Hence, whetstone sharpened networks do not require a spike-based coding scheme, thus producing networks comparable in timing and size to conventional artificial neural networks. Whetstone has been demonstrated on a variety of architectures, such as dense and convolutional networks, and tasks, such as image classification, auto-encoders and semantic segmentation. It is currently implemented within the Keras [16] and is widely extendable.

B. Hardware Simulation Software

CrossSim is a crossbar simulation tool that helps model resistive memory crossbars and in future version to target digital memories [4] We plan to use ReRAM as synaptic weights for our spiking neural network since they are extremely dense and efficient. The integration of Whetstone and CrossSim will help us simulate and test our hypothesis for initial use cases.

IV. Integrating Whetstone and CrossSim

As a proof of concept, we have imported Whetstone-computed weights into to a previously published ReRAM device aware crossbar simulation environment, called CrossSim [4] . CrossSim has primarily been used so far to evaluate constraints on adaptive/learning neuromorphic functions [10], and the realistic impact of device variability in the learning context [17], but in principle this environment contains all needed functions to compute realistic inference only estimates given pre-computed weights.

A. MNIST example using ReRAMs

Network Architecture: We use a small densely connected neural network to demonstrate the workflow achievable through our CoDesign process. Specifically, we choose a basic

TABLE I

RESULTS FOR WHETSTONE GENERATED WEIGHT FOR RERAM CROSSBAR

Noise Mode	Synapse Type			
	TaOx:1R	TaOx:2R	ENODE:1R	ENODE:2R
Noiseless	96.44%	64.71%	96.88%	96.89%
Physical Noise, Standard Training	80.12%	84.68%	70.25%	79.76%
Physical Noise, In-Training Noise	93.15%	93.85%	95.15%	94.58%

network topology of Input-Shape (785*) $\rightarrow$ First hidden layer (300 neurons) $\rightarrow$ Output Layer (10 neurons, 1-hot output encoding) which we feel a representative of a vanilla network capable of classifying small, embedded inputs.

After training is complete, all activations are equivalent to step functions at 0.5. We remark that these are fundamentally instantiations of the classic threshold gate. **Suma:** Still true?**Chris:** YesAll networks were trained using batch normalization before each non-linearity, with the batch norm layers removed after training and merged into the weights and biases.To evaluate the role of in-training noise, we trained the networks with additive Gaussian noise at each pre-activation at various standard deviations ranging from 0.0 to 1.0 in increments of 0.1.**William:** What was the amplitude? Standard 1.0? Training under each of the aft-mentioned levels of Gaussian noise was repeated for 30 runs. **Suma:** still included?–seems like results were iffy here **Chris:** the results were not promising enough to use , so I commented the folowing out The simulator environment also contains probabilistic models of device behavior which are used to construct accurate crossbar models. We have considered three cases: 1) Where Whetstone-derived weights are mapped to a perfect crossbar with no ideal effects (results are always within $\approx 0.1\%$ of the floating point results); 2) where unaware weights (computed in a standard way in Whetstone) are mapped to a realistic crossbar adding Gaussian noise to every weight during each VMM , and 3) where weights have been trained in a noise-aware way prior to being imported to the system battling noise during the VMM operations. The results for these three cases are shown in Table I for two different devices: a non-linear Tantalum Oxide ReRAM device considered for VMM applications in [10], [17], [9] and a relatively linear polymeric synapse device [19]. In addition to these cases, we have also highlighted the difference between multiple devices per synapse approach, and the single synapse approach, often referred to as 2R/1R respectively ; as noted in [20], [21], this effect is notably useful in battling device asymmetries as well as noise. In particular, since the logical weight's value is shared by two devices and noise can cancel out in either direction . *Results:* **Chris:** I have re-ordered and eliminated duplicate explanations Table I demonstrates that, given a Whestone style network trained offline without

*MNIST images are 28-by-28 pixels for a flattened dimension of 784; however, we use an additional input neuron to represent the 'bias' similar to[18]. The networks was originally trained with biases, but all biases were converted to the weights of bias neurons.

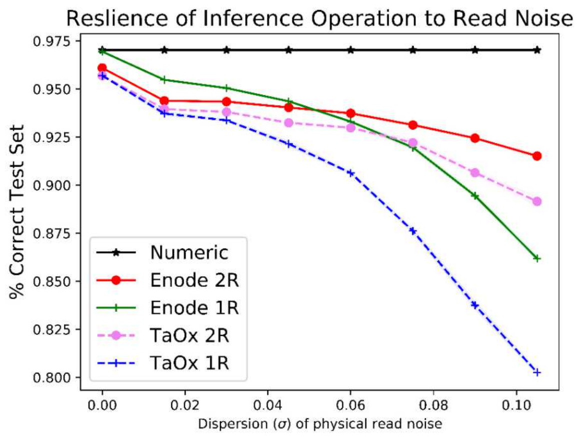

Fig. 2. Resilience of a Crossbar devices trained on Whetstone with Noise : Shows resilience of the improved noise-aware training for CrossBar Devices

any applied noise, the physical application of read noise can have a significant deleterious effect, usually more than 15% as compared to the simulated perfect (noise-less) crossbar performing inference with said devices and synapse styles. However, with the noise-aware style, we can achieve within 1.6% and 2.9% of the perfect inference accuracy. Although we expect the digital to analog and analog to digital peripheral circuitry in our arrays to provide low bounds on physical noise, we have also evaluated the resilience of these different approaches as the level of device noise present during each VMM operation progressively increases to even higher levels. As visible, more linear (e.g. ENODe) synapse physical device and redundant synapse logical models (2R) are protective at very large read noise, in contrast to the other considered choices. In Fig. 2 we demonstrate the resilience of a few different approaches as the level of device noise present during each VMM operation progressively increase. As visible, more linear (e.g. ENODe) synapse physical device and redundant synapse logical models (2R) are protective at very large read noise, in contrast to the other considered choices.

Discussion: Our preliminary results demonstrate the co-integration of Whetstone and CrossSim tools under the *Mosaics* framework. It shows that Whetsone derived frozen weights for a spiking neural network can be ported to a device/physics realistic edge inference tool like CrossSim. We also demonstrate how how training noise and inference noise interact.

V. INTRODUCTION TO *Mosaics*

As illustrated in Fig. 4, for a number of applications (such as deep artificial neural networks (ANNs)), the overall size of an algorithm may be very large, but parts of the necessary computation are repeated. *Mosaics* simply refers to a form of neuromorphic multiplexing, whereby the certain computations (in this example, the neurons, which are often a limiting resource) can be reused for different stages of an algorithm;

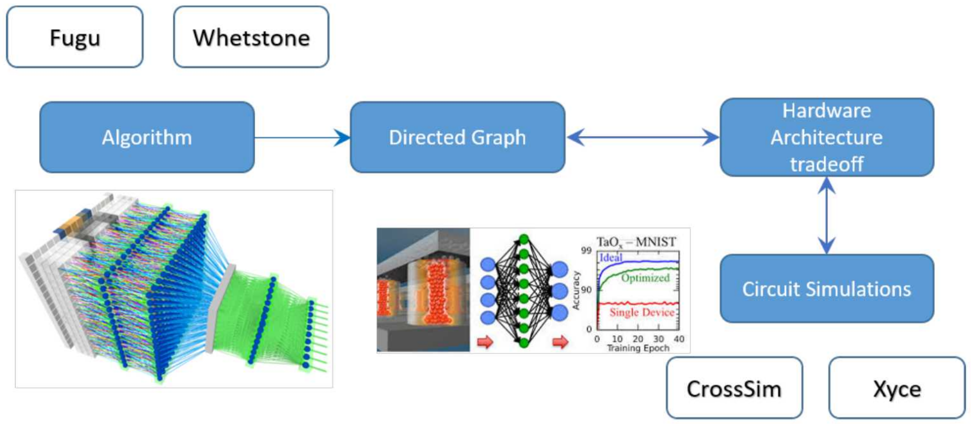

Fig. 3. Proposed Algorithm Hardware CoDesign methodology. Some part of the image reproduced from [3], [4], [22]

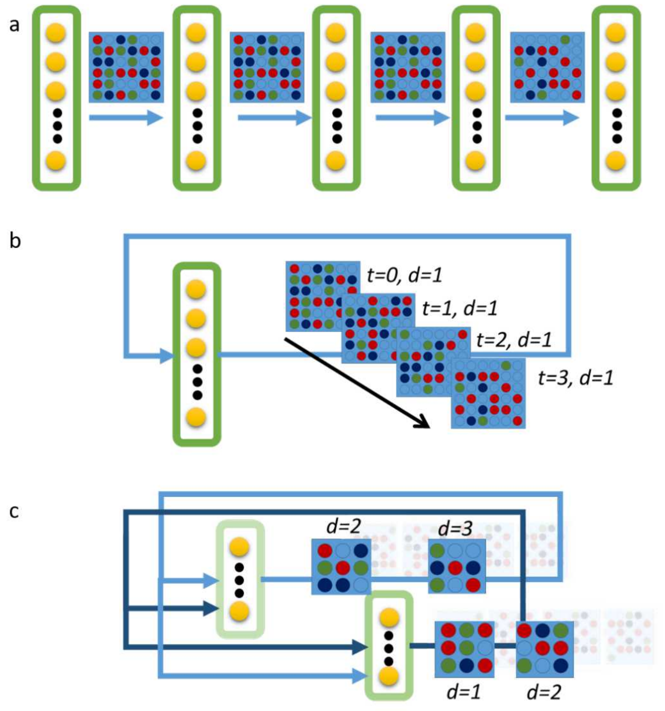

Fig. 4. (**a**) Simple neural network with layers connected by weight matrices. (**b**) Mosaic implementation, with one layer of processing neurons recurrently connected, while progressively using the weight matrices from above. (**c**) Further *Mosaics* implementation, with each layer broken into component sub-layers. This requires that the connection matrices also be partitioned with different delays to permit connections to 'skip' layers to target appropriate node

allowing larger scale algorithms to be implemented on a resource restricted neuromorphic platform.

The *Mosaics* concept that is introduced here offers an alternative that preserves many of the core benefits of specialized hardware, but extends it in a critical way that can generalize a platform's utility. Consider a desired application or algorithm as an extended neural graph, one perhaps that is too large to fit on available hardware. Mosaics are effectively a partitioning of an ideal neural graph into sub-graphs that enable the computation to be performed in isolation on a smaller subset of available computing nodes. This directly has the effect of trading space for time in computation. We expect that this co-design capability will be a critical contribution to advancing application-impactful neuromorphic computing technologies.

We are developing a tool flow that uses existing tools like CrossSim, Xyce [23], Whetstone and Fugu [24] to analyze and validate the *Mosaic* approach.

Fugu is a programming platform developed by Sandia that helps the user go from an algorithm to Spiking networks and target it on to a neuromorphic platform [24]. Fugu enables a user to design their computational flow as blocks and then converts it into a directed graph of spiking neurons. This graph can then be targeted on any neuromorphic hardware. Fugu is still actively being developed with hardware-agnostic backends which will enable it to target different neuromorphic platforms. In addition to CrossSim, we plan to integrate Xyce [23] into our tool framework. Xyce is an open source SPICE-compatible tool that enables high performance circuit simulation. It is capable of solving extremely large-scale circuit problems by supporting large-scale parallel computing platforms [23]. Integrating this tool into our framework enables us to explore other viable analog circuits to interface with our ReRAM model while building large-scale architectures.

VI. Conclusion

In this paper we first introduced a co-design framework to combine the benefits from rapid advances in both high-density analog memories and spiking algorithms to maximize gains from building hardware accelerator architectures. Our preliminary results demonstrate the co-integration of Whetstone and CrossSim tools and showed significant performance gains when the spiking network was trained with crossbar read noise. It shows that Whetsone derived frozen weights for a spiking neural network can be ported to a device physics realistic inference tool like CrossSim. We also demonstrate how how training noise and inference noise interact. Second, we discussed the *Mosaics* concept for resource reuse and how this approach can extract more out of neuromorphic architectures than would be otherwise expected.

References

[1] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[2] M. J. Marinella, S. Agarwal, A. Hsia, I. Richter, R. Jacobs-Gedrim, J. Niroula, S. J. Plimpton, E. Ipek, and C. D. James, "Multiscale co-design analysis of energy, latency, area, and accuracy of a reram analog neural training accelerator," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 86–101, 2018.

[3] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Aimone, "Training deep neural networks for binary communication with the whetstone method," *Nature Machine Intelligence*, vol. 1, no. 2, p. 86, 2019.

[4] *Crossbar Simulator: CrossSim*, 2019, accessed October 2nd, 2019. [Online]. Available: https://cross-sim.sandia.gov

[5] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *2011 International Reliability Physics Symposium*. IEEE, 2011, pp. MY–7.

[6] J. Joshua Yang, M.-X. Zhang, M. D. Pickett, F. Miao, J. Paul Strachan, W.-D. Li, W. Yi, D. A. Ohlberg, B. Joon Choi, W. Wu *et al.*, "Engineering nonlinearity into memristors for passive crossbar applications," *Applied Physics Letters*, vol. 100, no. 11, p. 113501, 2012.

[7] S. G. Kim, J. C. Lee, T. J. Ha, J. H. Lee, J. Y. Lee, Y. T. Park, K. W. Kim, W. K. Ju, Y. S. Ko, H. M. Hwang *et al.*, "Breakthrough of selector technology for cross-point 25-nm reram," in *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2017, pp. 2–1.

[8] J. Zhou, K.-H. Kim, and W. Lu, "Crossbar rram arrays: Selector device requirements during read operation," *IEEE Transactions on Electron Devices*, vol. 61, no. 5, pp. 1369–1376, 2014.

[9] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *Proceedings of the 53rd annual design automation conference*. ACM, 2016, p. 19.

[10] S. Agarwal, S. J. Plimpton, D. R. Hughart, A. H. Hsia, I. Richter, J. A. Cox, C. D. James, and M. J. Marinella, "Resistive memory device requirements for a neural algorithm accelerator," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 929–938.

[11] M. Bavandpour, M. Mahmoodi, H. Nili, F. M. Bayat, M. Prezioso, A. Vincent, D. Strukov, and K. Likharev, "Mixed-signal neuromorphic inference accelerators: Recent results and future prospects," in *2018 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2018, pp. 20–4.

[12] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W.-m. W. Hwu, J. P. Strachan, K. Roy *et al.*, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2019, pp. 715–731.

[13] M. N. Bojnordi and E. Ipek, "Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2016, pp. 1–13.

[14] H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr, "Recent progress in analog memory-based accelerators for deep learning," *Journal of Physics D: Applied Physics*, vol. 51, no. 28, p. 283001, 2018.

[15] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.

[16] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[17] C. H. Bennett, D. Garland, R. B. Jacobs-Gedrim, S. Agarwal, and M. J. Marinella, "Wafer-scale tao x device variability and implications for neuromorphic computing applications," in *2019 IEEE International Reliability Physics Symposium (IRPS)*. IEEE, 2019, pp. 1–4.

[18] C. M. Vineyard, R. Dellana, J. B. Aimone, F. Rothganger, and W. M. Severa, "Low-power deep learning inference using the spinnaker neuromorphic platform," in *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*. ACM, 2019, p. 13.

[19] Y. van de Burgt, E. Lubberman, E. J. Fuller, S. T. Keene, G. C. Faria, S. Agarwal, M. J. Marinella, A. A. Talin, and A. Salleo, "A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing," *Nature materials*, vol. 16, no. 4, p. 414, 2017.

[20] O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, and C. Gamrat, "Visual pattern extraction using energy-efficient "2-pcm synapse" neuromorphic architecture," *IEEE Transactions on Electron Devices*, vol. 59, no. 8, pp. 2206–2214, 2012.

[21] M. Bocquet, T. Hirztlin, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "In-memory and error-immune differential rram implementation of binarized deep neural networks," in *2018 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2018, pp. 20–6.

[22] S. Agarwal, R. B. J. Gedrim, A. H. Hsia, D. R. Hughart, E. J. Fuller, A. A. Talin, C. D. James, S. J. Plimpton, and M. J. Marinella, "Achieving ideal accuracies in analog neuromorphic computing using periodic carry," in *2017 Symposium on VLSI Technology*. IEEE, 2017, pp. T174–T175.

[23] *Xyce: Parallel Electronic Simulation*, 2019, accessed October 6th, 2019. [Online]. Available: https://xyce.sandia.gov/

[24] J. B. Aimone, W. Severa, and C. M. Vineyard, "Composing neural algorithms with fugu," *arXiv preprint arXiv:1905.12130*, 2019.