AFRL-RI-RS-TR-2023-089

MAGNETIC NANOELECTRONICS FOR BRAIN-INSPIRED COMPUTING (MN-BRIC): FROM CIRCUIT MODELS TO FULL SYSTEM ARCHITECTURE SIMULATIONS

RENSSELAER POLYTECHNIC INSTITUTE

MAY 2023

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2023-089 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
JOSEPH E. VANNOSTRAND
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing and Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
MAY 2023	FINAL TECHNICAL REPORT	JUNE 2021	DECEMBER 2022

4. TITLE AND SUBTITLE

MAGNETIC NANOELECTRONICS FOR BRAIN-INSPIRED COMPUTING (MN-BRIC): FROM CIRCUIT MODELS TO FULL SYSTEM ARCHITECTURE SIMULATIONS

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
	FA8750-21-1-1010	62551D/62788F
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
		R34E

6. AUTHOR(S)

Christopher D. Carothers

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Rensselaer Polytechnic Institute
110 8th Street
Troy NY 12180

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Research Laboratory/RITB 525 Brooks Road Rome NY 13441-4505	AFRL/ RI	AFRL-RI-RS-TR-2023-089

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This technical report summarizes the R&D efforts for the AFRL project “Magnetic Nanoelectronics for Brain-Inspired Computing (MN-BRIC): From Circuit Models to Full System Architecture Simulations”. This research project enabled the performance benchmarking of a spintronics hardware platform designed for handling neuromorphic tasks. Spintronics devices that use the spin of electrons as the information state variable have the potential to emulate neuro-synaptic dynamics and can be realized within a compact form-factor, while operating at ultra-low energy-delay point. To explore the benefits of spintronics-based hardware on realistic neuromorphic workloads, we developed a Parallel Discrete-Event Simulation model called Doryta, which is further integrated with a materials-to-systems benchmarking framework. The benchmarking framework allows us to obtain quantitative metrics on the throughput and energy of spintronics-based neuromorphic computing and compare these against standard CMOS-based approaches. Although spintronics hardware offers significant energy and latency advantages, we find that for larger neuromorphic circuits, the performance is limited by the interconnection networks rather than the spintronics-based neurons and synapses. This limitation is overcome by architectural changes to the network.

15. SUBJECT TERMS

Magnetic, antiferromagnetic, neuromorphic, nanoelectronics

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	22

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
JOSEPH E VANNOSTRAND	N/A

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Table of Contents

List of Figures

List of Tables

1.0 Summary

This technical report summarizes the R&D efforts for the AFRL project "*Magnetic Nanoelectronics for Brain-Inspired Computing (MN-BRIC): From Circuit Models to Full System Architecture Simulations*". This research project enabled the performance benchmarking of a spintronics hardware platform designed for handling neuromorphic tasks. Spintronics devices that use the spin of electrons as the information state variable have the potential to emulate neuro-synaptic dynamics and can be realized within a compact form-factor, while operating at ultra-low energy-delay point. To explore the benefits of spintronics-based hardware on realistic neuromorphic workloads, we developed a Parallel Discrete-Event Simulation model called *Doryta*, which is further integrated with a materials-to-systems benchmarking framework. The benchmarking framework allows us to obtain quantitative metrics on the throughput and energy of spintronics-based neuromorphic computing and compare these against standard CMOS-based approaches. Although spintronics hardware offers significant energy and latency advantages, we find that for larger neuromorphic circuits, the performance is limited by the interconnection networks rather than the spintronics-based neurons and synapses. This limitation is overcome by architectural changes to the network.

Through *Doryta* we are also able to show the power of neuromorphic computing by simulating Conway's Game of Life (GoL), thus showing that it's Turing complete. We show that *Doryta* obtains over 300 times speedup using 1,024 CPU cores when tested on a convolutional, sparse, neural architecture. When scaled-up 64 times, to a 200 million neuron model, the simulation ran in 3:42 minutes for a total of 2000 virtual clock steps. Conservative was found to be faster in most cases than optimistic, even when the tie-breaking mechanism, which guarantees deterministic execution, was deactivated. This is attributed to the lockstep synchronization of the model.

2.0 Introduction

Neuromorphic computing is a non-von Neumann approach to computing inspired on the brain with the explicit goal of performing machine learning and related tasks in a highly-efficient manner. For example, as part of the DARPA SyNAPSE program, IBM created an instance of a spiking neuromorphic processor, called TrueNorth, capable of multi-object detection and classification from real-time video input consuming only 63 milliwatts. The chip has 4096 neurosynaptic cores with a total of 1 million spiking neurons and 256 million re-configurable synapses. Another example is Intel's *Loihi,* 2018, a spiking neuromorphic processor capable of performing on-chip learning. Even with these low energy consumption chips in development, there is hope that we have not yet reached the minimum energy limit for neuromorphic computing; according to Hasler and Marr, biological neurons and synapses, if realized truly efficiently in silicon, would be able to compute 1.0e18 multiply-accumulate operations (MAC) per second using only 1 watt of power. To that end, it is important to continue improving the state-of-the-art and further reduce the hardware costs associated with neuromorphic computing.

To seek out improvements at the hardware level, we could make use of spintronic devices for the fabrication of electronic neurons and synapses in a brain-inspired architecture. Spintronics devices, including antiferromagnets and ferromagnets, made using magnetic materials are non-volatile and can mimic the dynamics of biological neurons and synapses in hardware in an energy-efficient and compact form-factor, i.e., occupying around the same space of a single transistor. This energy and area efficiency will potentially open a new artificial intelligence (AI) paradigm endowed with real-time learning, adaptation, and prediction. Such brain-inspired AI hardware can be used in remote, ``edge computing" environments with size, weight, and power constraints.

To facilitate our evaluation of spintronics-based neuromorphic hardware, we have developed a multi-scale modeling and simulation approach where physical hardware costs (i.e., energy, area, and latency) are calculated for key neuromorphic operations including neuron integration, neuron fire, and signal communication. These performance models are then imported into a new neuromorphic parallel simulation model called *Doryta* which enables the energy performance exploration of these devices for neuromorphic applications across a full-scale neural network architecture model.

The key contributions of this work include: (i*) Development* of Doryta, a deterministic, parallel spiking neural network simulation platform that is able to execute real neuromorphic applications in simulation, validated against existing spiking neural network tools; (ii) *Implementation* of Conway's Game of Life as a pure neuromorphic application model for Doryta, thus demonstrating that spiking neural networks are Turing-complete; (iii) *Evaluation* of parallel Doryta simulation performance wherein the Game of Life neuromorphic application model obtains over 300x speedup using 1024 CPU cores. (iv) *In depth analysis* of what makes the conservative approach for synchronization generally faster than optimistic when running a discrete-event simulations such as Game of Life, and the impact of different configuration changes such as size of the model and the activation of the tie-breaker mechanism; (v) *Quantification* of the energy, chip area, and runtime performance of spintronics-based neurons and synapses for neural network inferencing tasks resulting in three to six orders of magnitude improvement in energy-delay product over CMOS designs; (vi) *Exploration* of the effect that interconnect size and wire width have on

performance metrics. (vii) *Development* of user-defined event priorities to work in tandem with the existing event tie-breaking mechanism of the ROSS simulation engine.

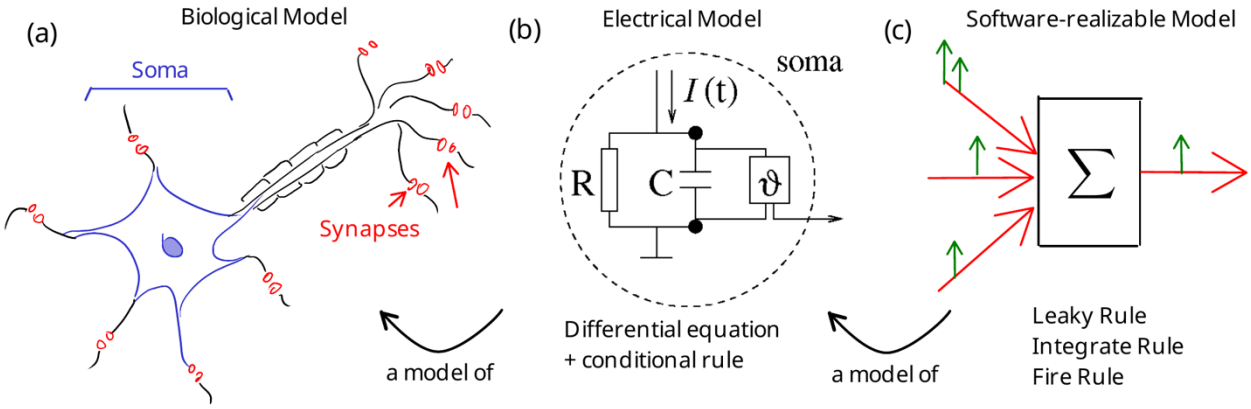

Figure 1. *Source of inspiration for Spiking Neural Networks and abstraction levels. (a) The biological spiking neuron. % and its behavior as function of current. (b) A model (Leaky-Integrate Fire) of the biological spiking neuron. (c) Software-realizable neuron model: neuron (soma) and synapses.*

3.0 Methods, Assumptions and Procedures

3.1 Background

Ubiquitous Artificial Neural Networks (ANNs) are not the only relevant bio-inspired development to come out of studying the brain. Spiking Neural Networks (SNNs), as the name implies, are based on the same foundation as ANNs, the biological spiking neurons. The biological spiking neuron is one of the prevalent neuron types in the brain. It responds to stimuli in the form of spikes. The simplest electrical model of the spiking neuron is the Leaky-Integrate Fire (LIF) Neuron, see **Figure 1**. The neurons in a neural network are interconnected via axons and synapses. We use the term *synapse* to refer to the connection between two neurons at a logical level: from a neuron that fires and sends a spike to a neuron that receives it. At the hardware level, neurons are connected via physical wires or *interconnects*, while the synapse is a circuit element capable of changing the intensity of a signal on a wire. All synapses have two attributes: how much current arrives at the neuron and a delay. For simplicity, we assume that all synapses have the same delay, one clock cycle. With neurons and synapses, we can construct many complex structures, also called neural network (NN) architectures or simply architectures. In fact, we can construct the same architectures defined for ANNs. Although it is not trivial to transform an arbitrary ANN with its weights into an SNN with its parameters, it is possible to implement the architecture on either. A full SNN model is defined by an architecture and three procedures (or operations): the leak operation, the integrate operation, and the fire operation. At the hardware level, we realize these three operations using spintronics-based spiking neurons and synapses, and metallic interconnects.

Spintronics Neurons and Synapses: The field of spintronics is expected to support semiconductor-based microelectronics in `More-than-Moore' and `Beyond Moore' information technologies. In contrast to conventional electronics that deal with the charge of an electron, spintronics utilizes the spin of an electron to manipulate, transmit, store and detect information. Spintronic devices can be fabricated using back-end-of-the-line CMOS processes and, therefore, realized in modern fabrication facilities without much re-tooling. At the heart of a spintronic device is a magnetic material: ferromagnetic (FM) or antiferromagnetic (AFM), which acts as the active component and displays neuro-synaptic dynamics when perturbed by external input (e.g., current pulse or magnetic fields). The spin configurations of FM and AFM materials are distinct, which makes them functionally unique as the building blocks of neuromorphic hardware.

FM-based spintronics nano-devices, such as magnetic tunnel junctions, are commonly used for storage, sensing, logic, interconnections, and as non-linear radio-frequency oscillators. Recently, it was experimentally demonstrated that FM-based nonlinear oscillators could be used to build circuits that embed neural functionality and can perform speech and digit recognition with high accuracy. FM devices have also been used for realizing synaptic behavior in hardware. Furthermore, AFMs display current-tunable spiking dynamics, which could be used to replicate the functionality of spiking neurons in an NN architecture.

Antiferromagnetic (AFM) Spiking Neurons: Due to their unique spin arrangement, AFMs possess no net magnetization. When a spin-polarized electric current is injected into an AFM, it can display spiking dynamics, which are detectable in the form of an output voltage signal. Thus, AFM-based signal generators can emulate spiking neurons in a compact form-factor in hardware.

In order to excite spiking dynamics, an input spin current that exceeds a certain threshold is provided to the AFM. Based on the material parameters and the dimensions, the AFM neuron's spike rate and performance can be efficiently tuned. The neuron spike can be detected in the form of a voltage signal via the anomalous Hall effect (AHE) through a structure or via tunneling magnetoresistance (TMR. The latter has only been theoretically predicted, while AHE has been experimentally demonstrated in a variety of chiral AFMs. Other detection methods like anisotropic magnetoresistance (AMR) and tunneling anisotropic magnetoresistance (TAMR) may also be used in the case of Mn3Sn, although such methods have a weaker output signal compared to TMR and AHE at 300 K.In the case of collinear insulating AFMs, like NiO, a voltage signal may be detected in a non-local spin valve setup or via the inverse spin Hall effect in a bilayer structure of AFM and heavy metal.

Ferromagnetic Non-volatile Synapses: Memristive dynamics based on domain wall (DW) movement can be easily produced by stripe-shaped FM structures. The device response to input current highlights the ability of the device to store real-valued weights with plasticity. Thus, FM materials can act as compact and non-volatile hardware emulators of synapses. The conductance of the device is determined on a training phase. During the training phase, current flows between terminals and the synapse's conductance is set by the magnitude and the duration of the input current which in turn enables movement of domain wall motion with increasing input pulse durations. During inferencing, the output current between the terminals is measured. The output current is given as the product of the voltage across the terminals, and the memristor's conductance. The memristors can be electrically connected in an analog cross-bar fashion such that the net current flowing through the bit line is the weighted sum of the memristors' conductance multiplied by the input voltage.

Parallel Discrete-Event Simulation is an efficient method for modeling the behavior of complex systems with discrete interactions between many simulation entities and is a natural match for the independent, discrete behaviors in spiking neural networks. A PDES simulation is made up of agents or entities known as Logical Processes (LPs) and timestamped events triggered from and to LPs. These LPs are each mapped to the various cores or Processing Elements (PEs) that may exist. In our simulation environment, a single core or MPI process is home of one PE. Given a constant simulation size with some number of LPs, the more PEs we have, then, the fewer LPs will exist within each PE. Thus, the responsibility for simulating the behavior of all LPs in the simulation becomes more distributed as we increase the number of PEs.

Effective parallelization does not come for free. When distributing computation across multiple PEs, there needs to be some mechanism in place for synchronization, ensuring that correct event ordering is always maintained. In general, there are two approaches for addressing this synchronization problem. *Conservative approaches* ensure that events are processed in timestamp order by waiting until it is ``safe" to process each event (i.e., no smaller timestamped events will be received later). *Optimistic approaches* allow events to be processed out of order, but provide a mechanism to detect and erase incorrect event computations. Loosely, each LP has their own clock. The Global Virtual Time (GVT) is the minimum clock time across LPs, anything occurred before is considered to ``have happened" and any redundant information from that time can be forgotten.

For simulation models developed and used in this work, we leverage the Rensselaer Optimistic Simulation System (ROSS). ROSS is a framework for developing parallel discrete-event simulations. ROSS has demonstrated highly scalable, massively parallel event processing capability for both conservative and optimistic synchronization approaches. ROSS' conservative execution is inspired by the YAWNS protocol, utilizing an event creation lookahead window restriction that ensures events cannot be created in a way that causes out-of-order processing. ROSS' optimistic execution is accomplished by implementing the Time Warp protocol which works with virtual time for event time management. ROSS mitigates Time Warp state-saving overheads via *reverse computation*. In this approach, rollback is realized by performing the inverse of the individual operations that were executed in the event computation. This eliminates the need to explicitly store prior LP state, leading to much more efficient memory utilization.

Another useful feature of ROSS is its ability to deterministically break ties between events that occur simultaneously. This feature orders independent simultaneous events in an unbiased, arbitrary ---but reproducible--- order. We build on this feature in this work to incorporate *user-defined priorities* to allow for certain events to always be processed before others in the case of event simultaneity.

3.2 Doryta: Simulating Spiking Neural Networks

Doryta is an architecture-agnostic and deterministic Spiking Neural Network simulator and is written in C as a ROSS model, and thus can run in virtually any system with a C compiler and a compatible MPI library. Doryta's job is not to train a network for a task, i.e., no neuron parameter is altered in the simulation of the network, instead, it is intended to be a reliable, deterministic, and time-aware simulator of SNNs.

Doryta's development is guided by several principles: modularity, reproducibility, determinism, minimal use of third party libraries, and the mantra ``the LIF neuron is king''. Although modularity has its shortcomings ---it tends to produce less efficient code than a monolithic implementation and a high degree of understanding is required to glue all pieces together--- we vouch for modularity to be center stage as it has allowed us to extend Doryta's capabilities with minimal refactoring. Modularity also helped us detect bugs by making it easier to unit-test the code in granular detail.

Built into Doryta is a set of tests that allow the developer to check for the unintended injection of bugs or to verify that Doryta compiles and runs as expected in a new machine or architecture. Furthermore, because Doryta, by leveraging the ROSS PDES engine, is deterministic, tests can quickly determine if what was intended to be a minor change resulted in a different simulation output.

We hope that Doryta serves as a platform for other researchers to play and experiment with. This is only possible if Doryta can be compiled and executed without difficulty. Thus, we stand by our decision to restrict Doryta's development to the use of the minimum number of C capabilities and libraries.

We prefer to lean on the LIF neuron model for its combination of simplicity and capability. Although there are dozens of models for the biological neurons and dozens more models

implemented in libraries, many of these neuron models, however complex, can be boiled down to behavior similar to that of the LIF neuron, i.e., any neuron can be modeled by the application of three simple rules: the leak rule, the integrate rule and the fire rule.

Implementation Details: Doryta is implemented as a ROSS model divided up into multiple modules: driver LPs (neuron LP), layouts, model-loaders, and neuron types. Here, we explain in detail each module and related concepts such as ``Doryta modes'', event types, and delta t-step.

Each LP in Doryta represents a single neuron and is composed of three parts: an ID, a list of synapses (weighted connections to neurons), and, by default, a pointer to the LIF neuron parameters (potential, capacitance, current, resting and reset potential, and threshold). New neuron models can be implemented as C *structs* to replace the default LIF neuron model pointer.

In a continuous simulation, the state of the system is updated one delta-time step “t”. This delta t-step is model dependent and determines the granularity of the simulation. The smaller the time-step, the more precise the simulation, at the cost of a larger computational time. Given the nature of PDES, this time-step in Doryta has to be explicitly encoded as an event. We call this event a *heartbeat event*.

In the LIF neuron model there are three rules: leak, integrate, and fire. Notice that on one hand, in a continuous simulation, all three rules should be executed every delta t time-step, but this is not the case in DES where ``integration'' only occurs when a spike arrives at a neuron and in no other case. On the other hand, leak and fire occur only once even when more than one spike arrives at the neuron within two heartbeats. These two cases are executed by two distinct events: (i) *Heartbeat event*: It applies the rules: leak and fire, in that order. It is scheduled every delta t-step. It is received, processed, and sent by each neuron to itself. There is at most one heartbeat event per neuron at any given point. (ii) *Spike event*: It applies the integration rule (aka, summation) to the neuron, i.e., for the LIF neuron, the spike's current is added up to the input current at time “t”. It can be scheduled at any point in time. Spike events can be created in two ways: loaded when the simulation initializes or created by a neuron. There can be as many spikes per neuron as synapses it connects from.

Under optimistic execution, a PE might need to rollback some events in order to keep in sync with all the other PEs. To allow rollbacks, we save the neuron's state before modifying it.

Due to the possibility that two or more events could occur with the same timestamp (an event tie), we lean on the tie-breaking feature of ROSS to maintain simulation determinism and validity. This, however, will also guarantee that the order of any two independent, simultaneous events is explicitly randomized. In most cases for Doryta, this is acceptable; however, we would like for heartbeat events to always be processed before spike events, should they occur at the same time. This use case is an example of \emph{user-defined event priorities}.

To accomplish this, we extended the tie-breaking data structure of ROSS which enforces lexicographic ordering of events based on the values of the items within the structure. Before, ROSS would compare two events based on their regular timestamp; if there was a tie, then it would compare the two based on their uniquely generated, i.i.d uniform random tie-breaking values. To

implement the user-defined priorities needed for our simulation models, we insert into the structure another value after the regular timestamp but before the random tie-breaker value. This value is defined at event creation depending on the event type; heartbeat events get a higher priority value, spike events get a lower priority value. This allows for the order of simultaneous Doryta events to remain in an unbiased arbitrary order except in the case of comparing a heartbeat and a spike event. In that case, then the heartbeat event will always be processed first.

Unless a neuron receives a spike it will not fire. This is true only when we can guarantee that the neuron does not have positive leak, which would provoke neurons to fire with no input spikes. Therefore, without positive leak, there is no reason to schedule a heartbeat event for every single delta t time-step. We can leverage this fact and run simulations in one of two modes: *Needy mode*: A heartbeat event is scheduled every delta t time-step regardless of neuron input, creating potentially needless events. *Spike-driven mode*: A heartbeat event is scheduled only after a spike is received, preventing the creation of heartbeat events when they are not needed.

To make simulations on both modes semantically the same, two constraints should always be maintained: heartbeats must always be scheduled at the same timestamps whether they are scheduled by another heartbeat or triggered by a spike and applying the leak operation consecutively for "n" delta t time-steps must yield (approximately) the same as computing the analytical solution of the leak operation over that same time larger n-step interval.

4.0 Results and Discussion

4.1 Physical Performance Estimation at the Hardware Level

The performance estimation is accomplished by first computing the performance of basic neurons, synapses and interconnects, and projecting them to the core and the chip levels for the neural networks later implemented. Note that a chip is composed of several cores. The methodology and performance metrics for neuronal spiking, synaptic integration, and data communication via metal interconnects is contained in a companion report lead by Prof. Shaloo Rakheja from UIUC.

4.2 Neuromorphic Applications

Conway's Game of Life (GoL) is an extensively studied cellular automata, a fascinating mathematical construction built out of a grid of cells and a list of rules that determine how the state of the cells change in time. Each cell in the grid is in one of two states: dead or alive, and each cell has eight neighbors. If the grid has a border, then some cells will have less than 8 neighbors. The rules of GoL are simple: a cell stays alive if there are two (2) or three (3) neighboring cells alive, otherwise it dies; and a cell comes to life if it has exactly three (3) alive neighbors, otherwise it stays dead.

It is possible to encode the rules of GoL, and thus simulate GoL, as a multi-layered Neural Network. In Figure 2, we show a 2-layer NN which simulates one step of GoL using as activation function the step function centered on 0.5. Notice that the shape of the network's output (the third layer) is the same as its input. This allows us to connect the output of the last layer as input to the first layer, thus building a recurrent network.

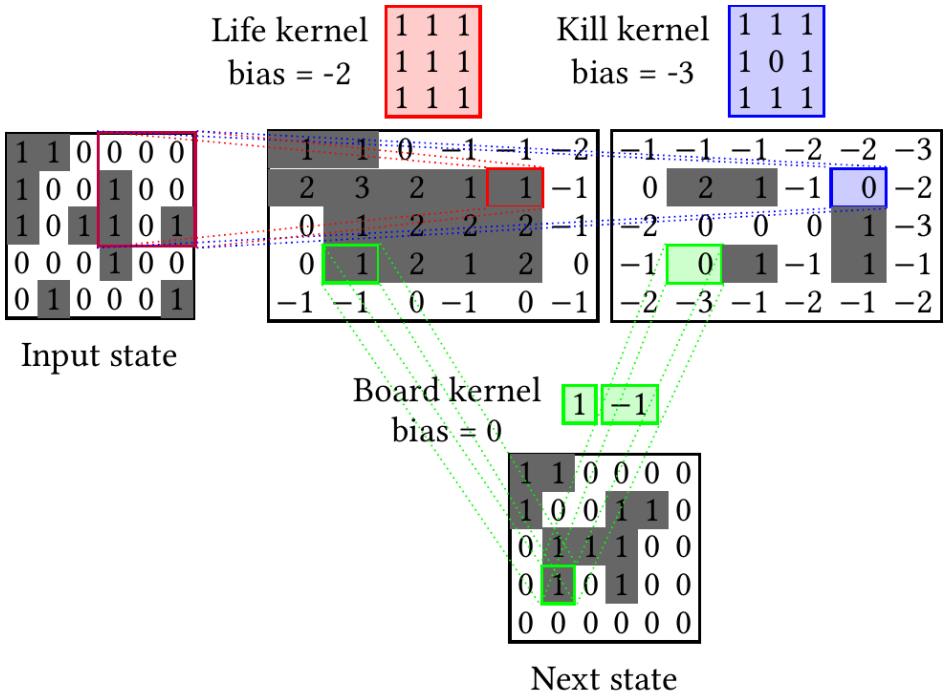

Figure 2. *Conway's Game of Life as a 2-layer convolutional Neural Network. The first layer has a padding of (1) and a kernel with two filters (Life Kernel and Kill Kernel). The second layer has no padding and a kernel of 1x1, two input channels and one filter (the Board Kernel). A neuron fires (in grey) when its summation (plus bias) surpasses (0), i.e., the activation function for both layers is the step function centered on 0.5.*

The first convolution is composed of a kernel of size 3x3 and two filters. We can analyze this kernel by breaking it up into two pieces, one per filter. The first 3x3 kernel corresponds to the Life kernel, which, when applied counts the number of cells alive in groups of 9 cells at the time. The bias for the Life kernel (-2) is added up to these counts resulting in the Life number. It can be seen in Figure 2 that the Life number for the uppermost-rightmost quadrant in the input state is equal to 1 (in red). Note that all numbers are natural numbers. If the Life number is bigger than zero, then either the neuron was dead and at least three neighbors are alive or it is alive and at least two neighbors are alive; in either, case the neuron output should be 1 (the neuron fires and a spike is sent). The second 3x3 kernel corresponds to the Kill kernel, which when applied counts the number of cells alive around a cell. The bias for the Kill kernel (-3) is added up to this count resulting in the Kill number. If the Kill number is bigger than zero, this means that the number of cells alive around a central cell is bigger than 4, which is not a good outcome for the central cell as it cannot be alive with that number of neighbors regardless of its current state (dead or alive).

In Figure 2, the Kill number for the cell-centered in the uppermost-rightmost quadrant is 0 (in blue). The two ``images'' resulting from the application of the first convolution (Life and Kill images) are binary, containing either 1s and 0s or spikes and no-spikes, and they tell us which cells have a chance to be alive, and which must be stopped and killed (respectively). The second convolution (between the second and third layers) is in charge of determining the “life” status of each cell given the output from the Life and Kill images. The Board kernel distinguishes the input from the Life image or Kill image. All values in the Life image are multiplied by 1 and all values

in the Kill kernel are multiplied by -1. The two resulting values, per cell, are added up, giving us the Board number. If the Board number is zero, then the cell is dead, but if it is one, then it is alive. In Figure 2, the green box on the next state image shows the result of adding up 1 …1 and 0 … -1.

A cell in GoL using the 2-layer NN strategy from above can be represented by three neurons, one for the Board kernel and two for the Life and Kill kernels. The simulation takes two clock cycles (delta t-steps) to simulate one time step of GoL. In the first clock cycle, the input state is sent as spikes from the first to the second layer following the Life and Kill kernel weights and the neurons fire if the conditions apply. In the second clock cycle, the neurons that fired on the previous cycle send a spike to the first layer where it is determined whether the cell is alive or dead.

The neuron parameters for GoL are: V_e, V_reset of 0, C of 0.5, R of 1and threshold equal to 0.5 – bias_i, where bias_i is the bias for the neuron i in one of the three possible neuron groups: Board, Life or Kill. The synapses' weights are: 0 if the neuron being connected corresponds to the same cell and the kernel is Kill, -1 if the synapse connects from the group of Kill neurons to the Board neurons, and 1 in any other case. See Figure 3(a) for GoL's crossbar configuration for a grid of size 20x20. The number of input lines for the Board layer is three times that of the size of the grid because it collects the result of the Life and Kill layers as well as the initial state input.

A natural corollary result from simulating GoL as a SNN is that: SNNs are Turing-complete given that GoL has already been shown to be Turing-complete in 1982. Thus, we have shown that the requirements for SNN Turing-completeness are at most: two neuron parameters (threshold and leak), one synaptic parameter (weight) and recurrent connections. Furthermore, it is possible to modify the 2-layer convolutional NN to make use of only non-negative numbers, which makes it possible to set a fixed threshold for all neurons getting rid of one neuron parameter, leaving us with only three requirements for Turing-Completeness: tunable leak, weighted synapses and recursive connections. Prior work made use of the equivalence between mu-recursive functions and Turing Machines to prove that SNNs are Turing-complete. We consider our GoL 2-layer NN to be much a simpler and understandable proof of Turing-completeness for SNNs.

(a)

Layer	Input Lines	Number of Neurons	Synapses per Neuron
Board	1200	400	3
Life	400	400	8.41
Kill	400	400	8.41

(b)

#	Type	Input Lines	Filters	Number of Neurons	Synapses per Neuron
1	Conv	784	1	784	1
2	Conv	784	6	784	22.90
3	Conv	784	6	196	4
4	Conv	1176	16	100	150
5	Conv	100	16	25	4
6	Full	400	-	120	400
7	Full	120	-	84	120
8	Full	84	-	100	84

Figure 3: *Crossbar configurations for: (a) Conway's Game of Life (layer 1 receives spikes from layers 2, 3 and the initial input); (b) LeNet as crossbar connection. Boxes indicate crossbar units; lines represent connections between units and input/output.*

MNIST Classification has become a popular machine learning benchmark that comprises a greyscale image dataset where each image is 28x28 pixels containing a handwritten image. A simple and popular feed forward network architecture used for image classification on MNIST is LetNet, which can be implemented in virtually any neural network framework, including Doryta and Whetstone. Whetstone is written in Python to train ANNs with some restrictions. An ANN model trained with Whetstone can be encoded into memoryless SNNs with little modification. Whetstone's trick is to define an activation function that approximates the step function to act as a threshold in SNNs. We follow the crossbar structure and crossbar parameters as seen in Figure 3(b). The number of classes in MNIST is only 10, not 100, but the output of our network is 10 times larger because Whetstone requires redundant neurons to properly classify an image.

Because models trained in Whetstone use simpler, “memoryless” neurons, we have to decide on the parameters that the spiking neurons should take. The spiking neuron parameters we have selected are: V_e and V_reset of 0, C of 1/256, R of 1 and the V_th (threshold) is equal to the bias of the neuron (plus 0.5 as per details of Whetstone's implementation). The synapses' weights are in the same manner equal to the weights obtained from Whetstone training.

Validating Doryta against Whetstone: After training LeNet in Whetstone, we saved it into a binary file readable by Doryta. The network was loaded in Doryta and then fed 10,000 test images. LeNet's output in Doryta was compared spike per spike against Whetstone's output. No differences

in the output of both Doryta and Whetstone were found. This is remarkable as both Doryta and Whetstone are written in separate languages with completely different underlying libraries. In simulation speed, however, Whetstone takes seconds to infer the class of all 10,000 the images while Doryta takes a couple of minutes. This discrepancy is due to Doryta's nature of simulating individual neurons that forget at an exponential rate, while Whetstone treats each neuron as a summation box of input spikes with no memory of past events. We expect that once transferred to hardware, memory full neurons will operate independently and in parallel of each other at much higher rates than what Doryta can simulate. Inferencing a single image requires only 8 clock cycles (one clock cycle per layer) which in spintronic hardware could take up to only a couple of microseconds.

Table 1: Strong scaling for a simulation of GoL using the tie-breaker mechanism. Randomly initialized GoL grid with size of 1024x1024. Sequential execution time: 5230.72 secs. Conservative mode execution with batch size of 512 and GVT interval of 128. Optimistic simulation restricted to a maximum lookahead of ½ delta T, with a batch size of 32 and 128 KPs per PE.

		Conservative		Optimistic	
Nodes	Cores	Runtime (sec)	Speedup	Runtime (sec)	Speedup
1	2	3084.18	**1.70**	3224.01	1.62
1	4	1397.29	**3.74**	1457.20	3.59
1	8	648.34	**8.07**	679.55	7.70
1	16	315.65	**16.57**	332.94	15.71
1	32	157.65	**33.18**	165.90	31.53
2	64	77.33	**67.64**	81.23	64.39
4	128	40.81	128.16	40.53	**129.06**
8	256	23.55	222.09	21.43	**244.08**
16	512	18.12	288.75	13.98	**374.11**
32	1024	16.37	**319.58**	16.87	310.00

4.3 Experimental Results

Two sets of experiments were run: performance experiments using GoL for benchmarking, and energy estimation experiments using a convolutional neural network (CNN) classifier of MNIST benchmark images. Most experiments in this work were run using up to 32 compute nodes on RPI's AiMOS supercomputer. Each node of AiMOS is composed of two, 20-core IBM Power9 processors clocked at 3.15GHz and 512GB of RAM. Additional experiments were run to determine the cache hit rate using an 20 core Intel Xeon Gold 5218R processor and made use of Intel's Performance Counter Monitor (PCM) tool to collect low-level hardware performance counter statistics.

Strong Scaling Performance: As seen in Table 1, a strong scaling experiment was performed using GoL with a grid of size 1024 X 1024, starting on one core (or MPI rank) and scaled up to 1024 cores on 32 nodes. Note, that only 32 of the 40 cores available on each IBM Power9 compute node were used. The other 8 cores were made available for OS and other system jobs and to reduce the effects of any OS jitter. The parallel simulation ran for 1000 GoL steps starting with the same initial configuration (each cell had a probability of 20% of being alive). A total of 3,145,728 LPs are required for the simulation (3 neurons per cell) and 1.88e10^9 events were processed. Because of Doryta's determinism, all executions generated the same number of committed events, namely 1.88e10^9.

Table 2: Strong scaling for a simulation of GoL with tie-breaker mechanism ***deactivated***. Randomly initialized GoL grid with size of 1024 X 1024. Sequential execution time of 675 seconds. Conservative mode execution with batch size of 512 and GVT interval of 512. Optimistic mode restricted to a maximum lookahead of ½ delta T, with a batch size of 64 and 16 KPs per PE.

		Conservative		Optimistic	
Nodes	Cores	Runtime (sec)	Speedup	Runtime (sec)	Speedup
1	2	687.09	**0.98**	850.91	0.79
1	4	406.35	**1.66**	485.64	1.39
1	8	210.84	**3.20**	250.43	2.69
1	16	109.67	**6.15**	130.02	5.19
1	32	56.92	**11.85**	66.96	10.07
2	64	27.90	**24.17**	33.48	20.15
4	128	15.48	**43.58**	17.36	38.86
8	256	10.25	**65.82**	11.06	61.01
16	512	11.28	**59.77**	11.47	58.82
32	1024	12.10	**55.73**	14.18	47.57

After carefully tuning all ROSS simulation parameters (number of KPs per PE, batch size, and GVT parameters), it was observed that an optimistic event scheduler is nearly as fast as the conservative event scheduler, and in some cases even executed faster.

However, it is observed that Doryta performs much better with the conservative approach by default because of the intrinsic lookahead present in the model. Each neuron fires at the same virtual time (all heartbeats events are scheduled at the same virtual times) and each neuron schedules all spikes events for the same virtual time slot in the future. Because this virtual time slot is encoded as ½ delta T from any heartbeat, the simulation advances at ½ delta T-steps virtual time increments. With the optimistic approach, a PE might be too “optimistic” and might run ahead too far of this time increment, thus it must rollback all events that were processed outside the time increment. The solution to stopping the optimistic scheduler from being “overly optimistic” leverages the Elastic Time approach to constrain the optimism with a maximum lookahead value that is equal to the virtual time increment of ½ delta T. This small constraint is enough to make the optimistic simulation run nearly as fast as conservative as seen in Tables 1 and 2. This is

because it forces the simulation to increment at the same virtual times as the conservative simulation does. Every virtual increment step triggers the GVT operation in both modes. Note that for the constraint to drive the simulation at virtual time increments, no other mechanism should trigger the GVT operation but the lookahead constraint, thus implying that the GVT interval was set to a large value in the simulation. The minimum number of GVT operations to be performed in 1000 GoL steps is 4000 (4 per GoL step). Most conservative simulations utilized 4002 GVT operations while most optimistic 4001 (the discrepancy has to do with the initial event state, which is not synchronized to the virtual time increments).

To quantify the performance impact of tie-breaker mechanism ---which guarantees a deterministic simulation--- we collected parallel simulation performance data for models with and without the tie-breaker enabled as shown in Table 2. If the initial spike events (determined by the user) do not overlap with any heartbeat event, then the result of the simulation will be deterministic, i.e., the order in which tied events are processed does not matter in Doryta, in as much they are the same kind of event. We observe that the simulation slows by a factor of 7 in serial execution. However, the tie-breaker mechanism counts for only a factor of 1.6 when executed in parallel. The smaller difference between the parallel executions (with and without tie-breaker enabled) indicates that most of the runtime is spent on some other ROSS framework components, mainly the network.

Additional performance experiments are included in the ACM TOMACS submission. Additional combined energy performance data will be included in Prof. Shaloo Rakheja's companion report.

4.4 Publications in This Performance Period

1. E. Cruz-Camacho, S. Qian, A. Shukla, N. McGlohon, S. Rakheja and **C. D. Carothers**, *"Evaluating Performance of Spintronics-Based Spiking Neural Network Chips using Parallel Discrete Event Simulation"*, In Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (ACM SIGSIM-PADS '22). Held On-line. June 8–10, 2022.
2. E. Cruz-Camacho, S. Qian, A. Shukla, N. McGlohon, S. Rakheja and **C. D. Carothers**, *"Performance Evaluation of Spintronic-Based Spiking Neural Networks Using Parallel Discrete-Event Simulation", Under Review, Best of ACM SIGSIM-PADS 2023, ACM Transactions on Modeling and Computer Simulation*

5.0 Conclusions

We have presented Doryta, a parallel discrete-event-based, chip-agnostic simulator for neuromorphic applications applied to the energy estimation of novel spintronic-based devices. Due to its modular design and mapping strategy, we've observed that Doryta can be scaled up to well over a 1,000 CPU cores. Doryta simulated a sparse 200 million neuron model for a total of 2000 virtual clock steps in under four minutes using 1,024 CPU cores. Additionally, Doryta is able to reproduce the inference results of another spiking-neural network library, Whetstone, with one key advantage: it takes into account time information and delays. We showed that Turing-complete requirements for neuromorphic computing are at most: one neuron parameter (leak), one synaptic parameter (weight) and recurrent connections. Loading Whetstone's models in Doryta allowed us to determine the workloads that spintronic-based chips would require in terms of basic operations. In an analysis of two spintronic-neuron models, Mn3Sn and NiO, we found that the bulk of the energy consumption of the chips would be driven by the connection between neurons and not the neurons themselves, the interconnect. Compared to CMOS architectures, our analysis indicates that spintronic-based chips have an energy-delay product that is **three to six orders of magnitude smaller** at inferencing a single image using the LeNet architecture.

For future work, we intend to implement further non-ML applications using SNNs such as: digital circuits, RAM, and a fully-fledged computer digital computer; and make Doryta one of the supported simulators by PyNN, a simulator-independent specification framework. On the hardware performance estimation, we plan to incorporate the cost associated with peripheral circuitry and transducers or amplifiers that are needed in a working chip. We plan to look into ferromagnetic based spiking neurons due to their ease of fabrication and characterization in the GHz regime as opposed to THz for spintronics. Finally, since we found that interconnects are the bottlenecks, we see an interesting avenue for research in the improvement of interconnects. We believe that building on Doryta and the techniques in this work will be beneficial to deepening our understanding of the neuromorphic computing devices of the future.

6.0 References

none

7.0 List of Acronyms

ACM: Association for Computing Machinery

AFRL: Air Force Research Laboratory

AiMOS: Artificial Intelligence Multiprocessing Optimized System

ANN: Artificial Neural Network

CMOS: Complementary Metal Oxide Semiconductor

CNN: Convolutional Neural Network

CPU: Central Processing Unit

DARPA: Defense Advanced Research Projects Agency

GoL: Game of Life

KP: Kernel Process

LP: Logical Process

ML: Machine Learning

MNIST: Modified National Institute of Standards and Technology

MPI: Message Passing Interface

OS: Operating System

PCM: Intel's Performance Counter Monitor

PDES: Parallel Discrete Event Simulation

R&D: Research & Devcelopment

RAM: Random Access Memory

RPI: Rensselaer Polytechnic Institute

SNN: Spiking Neural Network

SyNAPSE: Systems of Neuromorphic Adaptive Plastic Scalable Electronics

TOMACS: Transactions on Modeling and Computer Simulation

UIUC: University of Illinois Urbana-Champaign