A CAD System for Exploring Neuromorphic Computing with Emerging Technologies

James S. Plank, Garrett S. Rose, Mark E. Dean and Catherine D. Schuman

March, 2017

42nd Annual GOMACTech Conference

https://www.gomactech.net/2017/index.html

The online home for this paper may be found at: http://neuromorphic.eecs.utk.edu
A CAD System for Exploring Neuromorphic Computing with Emerging Technologies

James S. Plank\(^1\), Garrett S. Rose\(^1\), Mark E. Dean\(^1\) and Catherine D. Schuman\(^2\)

\(^1\)Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, TN, USA 37996

\(^2\)Computational Data Analytics
Oak Ridge National Laboratory
Oak Ridge, TN, USA 37831

Contact Author Email: jplank@utk.edu

Abstract: Emerging technologies are attractive as replacement parts for conventional digital electronics as we move past Moore's Law in computing. However, any emerging technology must be accompanied by a battery of software infrastructure, both to "program" the technology, and to incorporate it into critical applications. Herein, we detail our research in software infrastructure for leveraging emerging technologies in neuromorphic computing systems.

Keywords: CAD; beyond Moore's law; neuromorphic computing, emerging devices; spiking neuromorphic networks; evolutionary optimization.

1. Introduction

Neuromorphic computing systems have re-risen to prominence as alternatives to standard Von Neumann computing systems for several key reasons. They feature smaller, more tightly coupled processing and memory cores that promise low power consumption and massive degrees of parallelism. They attack application areas in classification, control and security that are difficult for more traditional computing systems. In particular, the recent successes in convolutional neural networks (i.e., "Deep Learning") as image classification engines have dominated technology headlines, spawning scores of startups and academic research projects to leverage the success. Finally, neuromorphic computing systems feature the ability to leverage alternative technologies as core components.

At the University of Tennessee and Oak Ridge National Laboratory, we have a vertical research program centered around spiking neuromorphic computing systems. Spiking systems are different from conventional neural networks, as they model collections of neurons and synapses that accumulate and transmit electrical charge over time, as in real biological systems. We have applied them to control and classification applications that feature a temporal component. The temporal component has proved difficult for Deep Learning systems, whose programming technique, based on back propagation, is not conducive to the ever-changing nature of temporal applications.

The research program's vertical nature is illustrated in Figure 1. We call it a CAD system, because it enables the design of complex hardware solutions to neuromorphic problems. The top of the picture are the two components of the project that have external visibility: the applications and how they are programmed on the underlying neuromorphic system. We will discuss our applications below. Our current programming technique is based on a custom mapping of application inputs and outputs to electrical charge events, and evolutionary optimization (EO) of neural networks that are specific to the applications. The EO is based on training, starting with a seeded population of networks, which each application rates using a custom fitness function. As the EO progresses, it refines and improves the networks based on

\[ \text{Applications} \quad \quad \text{Programming} \]

\[ \begin{align*}
\text{Software support for neural networks,} \\
\text{programming and models.} \\
\end{align*} \]

\[ \begin{align*}
\text{Model} \\
\text{Simulation} \\
\text{Devices} \\
\end{align*} \]

\[ \begin{align*}
\text{Model} \\
\text{Simulation} \\
\text{Devices} \\
\end{align*} \]

Figure 1. Vertical organization of our CAD system for spiking neuromorphic computing systems. Components in white boxes are part of the software infrastructure.
the fitness function, and upon success, the EO emits a network that applies to the application successfully. EO is a powerful programming technique, because it is generic to the application and to the specific neuromorphic model, but it can be limited in how fast it converges to discover successful networks. We are actively pursuing research on improving EO by employing massive parallelism and custom devices, and by honing the optimization itself.

Both applications and EO program to a general model of spiking neuromorphic system, supported by a layer of software shown beneath the applications and programming. This layer is the “glue” between the external components and the various neuromorphic models. At the bottom are the neuromorphic models themselves, which typically have multiple implementations: one or more software simulators, the neuromorphic models themselves, which typically have multiple implementations: one or more software simulators, used for development and verification, and the physical devices themselves.

2. Neuromorphic Computing Models
At present, our CAD system works with three different neuromorphic computing models:

1. **NIDA** (Neuroscience-Inspired Dynamic Architecture) [1] is an application of the RISC philosophy to spiking neuromorphic computing models, featuring simple, analog neurons laid out in 3D space. Synapse delays equal their lengths in the 3D space. The only features in NIDA beyond this simple description are refractory periods for neurons, and long-term potentiation and depression of synapse weights. NIDA is implemented in simulation only.

2. **DANNA** (Dynamic Adaptive Neural Network Array) is a model designed specifically for digital implementation. It features a programmable 2D array of elements, where elements are either neurons or synapses. Neuron thresholds, synapse delays, and synapse weights are programmable, and elements may only connect to their 16 nearest neighbors. Unlike NIDA, all components of DANNA are discrete. We have implemented DANNA successfully in simulation and on FPGA’s [2,3]. A VLSI implementation of DANNA, featuring very low power consumption, has been designed, but not yet fabricated [4]. The simulator has been implemented to be very fast, with a small memory footprint. Recent timings on a commodity workstation have demonstrated speeds that exceed IBM’s TrueNorth spiking neuromorphic hardware [3]. Further, we have recently enriched the simulator with CUDA code that enables banks of GPUs to execute multiple instances of DANNA in parallel.

3. **mrDANNA** (Memristive DANNA) is a model designed so that the neurons and synapses may be implemented by an emerging analog device called the *memristor*. Like DANNA, mrDANNA is based on a 2D grid of elements with constraints on connectivity, but like NIDA, they are analog in nature. mrDANNA is currently implemented in simulation. We discuss physical implementation using memristors in section 4 below.

To work within our CAD system, a model must implement an interface that includes the following components:

- A **network**, which is a layout of neurons and synapses, including input neurons that accept charge, and output synapses, whose firing events may be observed externally to the network. While we assume that networks are composed of neurons and synapses, we make no assumptions about the properties of these components or how they connect. That is up to the individual model.
- A **device**, on which a network may be loaded, and then run, receiving input charge events, and emitting output firing events.
- **Network operations**, such as random network generation, mutation and crossover, which facilitate programming by evolutionary optimization.

By implementing these operations, a model may then support applications that are programmed to the general system, by using evolutionary optimization to develop networks, specific to the model, that “solve” the application.

3. Applications
Our canonical application is a classic from control theory -- balancing a pole on a cart, which is on a horizontal track. The cart and pole's positions are applied periodically as inputs to the neuromorphic system, which decides whether to apply force on the cart to the left or the right. We have developed the application to our general model, and have successfully applied EO to design networks that keep the pole balanced in all three neuromorphic models described above. The interested reader is encouraged to view narrated demonstrations of the pole balancing application at [http://neuromorphic.eecs.utk.edu/pages/research-demos/](http://neuromorphic.eecs.utk.edu/pages/research-demos/).

A second working application is an adaptation of the cell phone game “Flappy Bird,” for which networks have been developed that keep the bird flying indefinitely. Other applications under development within our CAD framework are:

- A 3-dimensional navigation application whose goal is to steer a drone to a target, avoiding moving obstacles.
• An application to keep a helicopter hovering in brownout conditions.
• An application to perform real-time phoneme recognition on audio feeds.
• A general classification application to make classification decisions on data sets by using pre-classified data for training.
• A suite of micro-applications, such as bit arithmetic, Galois field arithmetic, counting, selection and multiplexing, whose goal is to improve evolutionary optimization, and to facilitate the composition of networks to achieve more complex goals.

We reiterate that the important feature of application development in our CAD system is that the application software is developed for the general spiking model. The evolutionary optimization can then be used to build solution networks for any of the implemented models without changing any of the application software, abstracting away the details of a particular model from the application developer.

4. Emerging Technologies
As mentioned above, we have developed a neuromorphic computing model called mrDANNA, which is based on DANNA, but designed specifically for implementation by memristors. We have completed the circuit designs, multiple levels of simulators, and are collaborating with researchers from SUNY Nanotech to perform chip design and fabrication for testing this emerging technology [5]. This work (and other DANNA research) is funded by the Air Force Research Laboratory (see acknowledgements).

We are commencing a second, multi-disciplinary project to explore biomimetic membranes as a neuromorphic computing fabric [6]. We call the fabric bmDANNA, and like mrDANNA, it will use DANNA as a basic neuromorphic computing structure, but adapt its features to leverage those of the emerging devices. As with mrDANNA, we will design the model, then implement a simulator to embed within our CAD structure to leverage the applications and programming tools that we have already developed. This is work funded by the National Science Foundation (1631472).

5. Contrasting Approaches
We briefly examine a few contrasting approaches to developing neuromorphic computing technology. A popular approach is to evaluate technology on benchmarks from biology, with the thought that mimicking physical systems will yield benefits [7]. While this approach may yield benefits in the future, we have rejected it in favor of an application-driven approach, where success is measured by how well an application is served by the neuromorphic structure, rather than how close the structure matches biology.

A second approach is to map the networks of one neuromorphic model to a second. This is the approach that researchers have used to program IBM’s TrueNorth hardware, developing Deep Learning convolutional networks, and then translating them to TrueNorth’s spiking structure [8,9]. This was the approach that we initially tried with mrDANNA, translating classification networks developed for NIDA [1]. While this approach does feature expediency, and it leverages previous work, it can only succeed to the degree that the new neuromorphic model shares commonality with the old one.

In contrast, our approach based on evolutionary optimization programs the application directly for the model, thereby making fewer assumptions about the model, and also leveraging potentially unique functionalities of each model. This works, because the models themselves implement the major operations of evolutionary optimization, namely mutation and crossover. The downside of evolutionary optimization is that when an application is complex enough, the optimization is unable to converge upon a solution. As mentioned above, we are performing research to improve optimization, and thereby increase the applicability of our approach.

6. Conclusion
We have developed a CAD framework for application development and programming based on evolutionary optimization, and then crafting solutions that are specific to each model, which in turn has been developed to be specific to the conventional or emerging technology that implements it. We have used this framework for control applications on the models NIDA, DANNA and mrDANNA. The last of these is based on an emerging technology, the memristor.

Our continued research agenda includes richer applications, including aircraft control and speech recognition, improved evolutionary optimization techniques for programming, tailoring evolutionary optimization to supercomputing hardware, continuing device-level implementations of DANNA and mrDANNA, and developing newer models, such as bmDANNA, for other emerging technologies.

Acknowledgements
This research was supported in part by an AFRL Information Directorate grant to the University of Tennessee at Knoxville (FA8750-16-1-0065), by the National Science Foundation under grant 1631472, and 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.

References
In International Joint Conference on Neural Networks, Vancouver, July 2016.


