Neuromorphic Computing: A Post-Moore’s Law Complementary Architecture

Catherine D. Schuman, Member, IEEE, J. Douglas Birdwell, Fellow, IEEE, Mark E. Dean, Fellow, IEEE, James S. Plank, Member, IEEE, and Garrett S. Rose, Member, IEEE

Abstract—We describe our approach to post-Moore’s law computing with three neuromorphic computing models that share a RISC philosophy, featuring simple components combined with a flexible and programmable structure. We envision these to be leveraged as co-processors, or as data filters to provide in situ data analysis in supercomputing environments.

1 INTRODUCTION

The post-Moore’s law era is upon us; it is vital that the community examine complementary, non-von Neumann architectures. A confluence of events beyond the end of Moore’s law indicate that new architectures should be researched, implemented, and deployed. These events include the end of Dennard scaling, the prevalence of the von Neumann bottleneck, and the continued difficulties of standard parallel programming. We advocate neuromorphic computing systems as complementary non-von Neumann architectures to augment existing systems. The key properties of neuromorphic systems that we are exploiting are (1) collocated computation and memory, (2) two basic components: neurons and synapses, (3) simple communication between components, (4) inherent parallelism in computation, and (5) spiking or discrete-event systems with low power requirements. We advocate the use of neuromorphic computing as a complementary architecture because the collocated memory and simple communication avoid von Neumann bottlenecks and are not as affected by slower communication speeds; the basic components and the inherent parallelism of computation help overcome the issues associated with parallel programming and synchronization, and innovations in device design help to overcome the issues with the end of Moore’s law and Dennard scaling.

2 APPROACH

Our approach to neuromorphic computing includes software and hardware components, as well as simulators for testing. We are exploring implementations using off-the-shelf components, such as FPGAs and GPUs, as well as custom implementations in VLSI and emerging device technologies such as memristors. Our goal has been to produce a common programming technology, and architectures with shared characteristics. All of our systems share two characteristics: very simple neurons and synapses with two parameters each (one parameter is a notion of memory while the other affects temporal components) and programmable/structure (in the number and placement of neurons and synapses). Each architecture also includes a primitive notion of on-line learning, inspired by long-term potentiation and long-term depression in biological systems. Our three major architectures (Figure 1) are:

- NIDA: A 3D model that is implemented in simulation only, from which other implementations are derived [9].
- DANNA: An 2D model designed specifically for digital hardware, currently implemented in FPGAs and in early-stage development for VLSI [4].
- mrDANNA: An implementation of NIDA using memristors as synapses.

![Fig. 1. Our neuromorphic models: NIDA (simulation only), DANNA (FPGA, future VLSI), and mrDANNA (leveraging memristors).](image)

We are developing simulation software and customized training implementations for each architecture. The training is based on evolutionary optimization, which defines the model structure and parameters for each application. In addition, we are developing system software for each implementation. Our neuromorphic approach differs from other neuromorphic computing approaches by taking a RISC approach, choosing very simple component functionality (in the neuron and synapse), but allowing for flexibility in the way those components are combined (in the programmability of structure available in each
of our neuromorphic architectures). We compare our architectures (NIDA/DANNA/mrDANNA) with five other major neuromorphic computing efforts in Table 1. In this table, we quantify programmable structure as whether the architecture allows for varying numbers and placements of neurons and synapses in the array and component complexity in terms of the number of parameters for neurons and synapses (listed in the table as number of neuron parameters / number of synapse parameters). Our programmability has an advantage for applications in supercomputing because it allows flexibility in parameter settings and in the way the components are connected and laid out in the system. Because of this flexibility, the user has a greater degree of freedom to exploit the properties of the neuromorphic architecture, potentially widening the scope of applicability of neuromorphic systems. The simplicity of our components helps alleviate the burden of actually programming the device to perform tasks. Our evolutionary optimization programming method can determine parameters and/or structure, allowing the user as much input in the training process as they desire.

2.1 Maturity of the Architecture

The most mature of all of our implementations is the FPGA implementation, with a working prototype. The mrDANNA implementation is in simulation, and the DANNA VLSI implementation is in early development. Supporting software for each system is in progress. DANNA simulation software is complete, and a version that uses GPUs is in development. Circuit-level simulations using SPICE have been completed for small mrDANNA networks, but a custom software simulation implementation is still in development.

2.2 Supercomputing Applications

We have identified two areas in supercomputing through which neuromorphic computers may be useful. One area is as a co-processor and the other is as an in situ data analysis processor. The co-processor may be used in a similar way as a GPU; in particular, certain applications or functionality within an application may be accelerated using the neuromorphic co-processor. Because neuromorphic systems in general and ours in particular can process spatiotemporal information, a neuromorphic system can also be used to pre-process data as it’s being generated by the supercomputer (for example, data from a large-scale science simulation). We have used our neuromorphic systems for simple classification [8], anomaly detection [7], and control [3] tasks. However, we do not expect to achieve state-of-the-art results on machine learning tasks such as image classification using our approach; deep learning techniques such as convolutional neural networks will likely still achieve better results. It may be possible to run a version of a convolutional neural network on our architectures, but we would still expect a performance drop as a result of the mapping process. This is why we advocate the use of neuromorphic systems as pre-processors for the data. We believe it will be possible to use a neuromorphic system as a filter, to identify interesting or anomalous results in the data generated, which can then be further processed by an off-chip, more intensive machine learning technique.

2.3 Risks

The investment in custom chips (DANNA VLSI and mrDANNA) will be significant; however, our focus on off-the-shelf commodities, such as FPGAs, minimizes the risk of investment, as the FPGA can be used in other ways (beyond being programmed as a DANNA). However, power efficiency and scalability is severely decreased by limiting to FPGAs. We recommend deployment in stages: small-scale deployment using FPGA implementations, in order to understand the characteristics and capabilities of the system on desired applications, followed by larger-scale deployment using either VLSI DANNA or mrDANNA as appropriate for the application. In addition to capital investment, significant time investment will be required to identify the proper applications and to develop programs or code for the neuromorphic devices. Especially for the co-processor example, the development of usability tools (such as compilers or compiler directives) to ease the transition between existing programming techniques and programming for neuromorphic computers will be a necessary step to alleviate some of the risk associated with adopting neuromorphic computers.

3 Conclusion

Neuromorphic computing is a promising complementary architecture for the post-Moore’s law era of computing. With its key properties of collocated computation and memory, RISC-like components (neurons and synapses), simple communication, and temporal processing ability, it has the potential to re-shape the way we think about what is possible through computing. Our neuromorphic architectures are implementations with these characteristics that are in various stages of development. We plan to produce functional prototypes of each of these architectures, that will be made available to the community for research purposes. We believe our architectures are particularly amenable for supercomputing applications because of their programmability.

<table>
<thead>
<tr>
<th>Programming Structure</th>
<th>Component Complexity</th>
<th>On-Chip Learning</th>
<th>Materials/Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>DANNA/mrDANNA</td>
<td>Neurons and synapses</td>
<td>2 / 2</td>
<td>Yes</td>
</tr>
<tr>
<td>Darwin [10]</td>
<td>Neurons and synapses</td>
<td>2 / 2</td>
<td>No</td>
</tr>
<tr>
<td>TrueNorth [9]</td>
<td>Fixed (Synapses on/off)</td>
<td>10 / 3</td>
<td>No</td>
</tr>
<tr>
<td>BrainScale [2]</td>
<td>Neurons and synapses</td>
<td>Variable (Multiple models)</td>
<td>Yes</td>
</tr>
<tr>
<td>SpiNNaker [6]</td>
<td>Neurons and synapses</td>
<td>Variable (Multiple models)</td>
<td>No</td>
</tr>
<tr>
<td>Neurogrid [1]</td>
<td>Fixed (Synapses on/off)</td>
<td>29 / 8 + additional</td>
<td>No</td>
</tr>
</tbody>
</table>
REFERENCES


