Exploiting Heterogeneous Parallel Programming for Developing an Educational Neuromorphic Computing Simulator

Faraz Hussain
fhussain@eecs.ucf.edu
Department of Electrical and Computer Science, University of Central Florida (UCF)

Alvaro Velasquez
velasquez@eecs.ucf.edu

Sumit K. Jha
jha@eecs.ucf.edu

ABSTRACT

With the explosive growth of multicore and GPU-based computing, accompanied by the long-predicted demise of Moore’s law, there has been an increasing interest in expanding the breadth and scope of undergraduate instruction in concurrent and distributed computing. We describe our plans for exploiting heterogeneous parallel computing architectures for developing an educational simulator for neuromorphic computing. We also report on our experience as early adopters of the NSF/IEEE-TCP Philipp CIDeur curriculum on Parallel and Distributed computing for Fall 2014 at the University of Central Florida. We include a brief overview of how our courses cover the theoretical and practical aspects of parallel computing, and describe the hardware and software infrastructure used for instruction and research at our institution.

OBJECTIVES

• Develop a neuromorphic computing simulator.
• Incorporate use of the simulator into upper-level undergraduate courses.
• Introduce parallel complexity theory to the graduate-level complexity course taught at UCF.

NEUROMORPHIC COMPUTING AND MEMRISTORS

The significant difficulties in transistor scaling and a concern for energy efficiency have driven the search for new computing technologies in the past decade. Among them, the memristor stands as one of the most promising due to its memory capabilities and non-volatility. It has also opened up new doors to the neuromorphic computing community due to its ability to house memory and computation units on the same chip, a problem which has plagued von Neumann architectures, even in the multicore era.

Neuromorphic computing seeks to mimic the very desirable properties of the most enigmatic, inherently parallel supercomputer: the human brain. In particular, we seek high energy efficiency, robustness to noise, and the ability to implement learning algorithms at the hardware level. The nascent memristor has demonstrated tremendous potential as an enabling technology that is perfectly suited for this purpose.

A memristor consists of a film of width \(D\) with a low-resistance doped region of width \(w\) and a high-resistance undoped region. Consequently, the memristor can be thought of as two resistors in series, with resistances \(R_{up}\) and \(R_{down}\), such that \(R_{up} \ll R_{down}\). A flow of current can cause the doped region to shrink or expand, thus increasing or decreasing the total resistance. With respect to Figure 1, this charge-dependent resistance directly affects \(V_{out}\). Thus, \(V_{out}\) can be seen as a function of \(V_{up}\) and the state of the memristor, similar to how an output neuron is dependent on input neurons and synaptic weights.

METHODOLOGY

UCF has been selected by NVIDIA to be a “2014 CUDA Teaching Center” in order to help promote parallel computing education at UCF; NVIDIA has donated CUDA-capable GPUs as well as instructional material. We plan to leverage these resources to develop a parallelized neuromorphic computing simulator based on memristor crossbars. We will also begin offering core courses that focus on parallel and CUDA programming.

A memristor crossbar consists of two sets of parallel wires, each set perpendicular to the other, with a memristor at each wire junction. Such a crossbar structure, coupled with the charge-dependent conductance of the memristor, can naturally encode a neuromorphic computing architecture. Figure 2 depicts a simple neural network and its memristor crossbar interpretation. Note that for any output \(V_{out}\) in the crossbar, its corresponding value is dependent on the inputs \(V_{up}\), \(1 \leq i \leq m\) and the states of the memristors \(w_{ij}\). Thus, \(V_{out} = f(V_{up}, w_{ij})\), where each memristor acts as a synaptic weight.

CONCLUSION

Over the next year, we will build a simulator for memristor-based neuromorphic computing - a framework primarily instructional purposes. This will add undergraduate students in their understanding of neuromorphic systems and also help interested research students perform experiments to analyze and validate neuromorphic architectures. We will continue our emphasis on PDC instruction for undergraduate and graduate students. In Spring 2014, we will teach a graduate level complexity theory course that will cover the PRAM model and complexity of parallel version of many commonly known algorithms. In addition to discussing time/space complexity, decidability, reducibility and intractability this course will also include a module on randomized algorithms that will introduce students to the complexity class ZPP, RP and R NC. We will document all instructional material and course-related information and will collect student evaluations for each of these courses and report to them to the CEDER center.

REFERENCES