The Memristive Boltzmann Machines

The proposed memristive Boltzmann machine is a massively parallel, memory-centric hardware accelerator based on recently developed resistive RAM (RRAM) technology. The proposed accelerator exploits the electrical properties of RRAM to realize in situ, fine-grained parallel computation within memory arrays, thereby eliminating the need for exchanging data between the memory cells and computational units.

Mahdi Nazm Bojnordi
University of Utah

Engin Ipek
University of Rochester

Combinatorial optimization is a branch of discrete mathematics that is concerned with finding the optimum element of a finite or countably infinite set. An enormous number of critical problems in science and engineering can be cast within the combinatorial optimization framework, including classical problems such as traveling salesman, integer linear programming, knapsack, bin packing, and scheduling problems, as well as numerous optimization problems in machine learning and data mining. Because many of these problems are NP-hard, heuristic algorithms are commonly used to find approximate solutions for even moderately sized problem instances.

Simulated annealing is one of the most commonly used optimization algorithms. On many types of NP-hard problems, simulated annealing achieves better results than other heuristics; however, its convergence may be slow. This problem was first addressed by reformulating simulated annealing within the context of a massively parallel computational model called the Boltzmann machine.[1] The Boltzmann machine is amenable to a massively parallel implementation in either software or hardware.[2] With the growing interest in deep learning models that rely on Boltzmann machines for training (such as deep belief networks), the importance of high-performance Boltzmann machine implementations is increasing. Regrettably, the required all-to-all communication among the processing units limits these recent efforts' performance.

The memristive Boltzmann machine is a massively parallel, memory-centric hardware accelerator for the Boltzmann machine based on recently developed resistive RAM (RRAM) technology. RRAM is a memristive, nonvolatile memory technology that provides Flash-like density and DRAM-like read speed. The accelerator exploits the electrical properties of the bitlines and wordlines in a conventional single-level cell (SLC) RRAM array to realize in situ, fine-grained parallel computation, which eliminates the need for exchanging data among the memory arrays and computational units. The proposed hardware platform connects to a general-purpose system via the DDRx interface and can be selectively integrated with systems that run optimization workloads.

Published by the IEEE Computer Society

Computation within Memristive Arrays

The key idea behind the proposed memory-centric accelerator is to exploit the electrical properties of the storage cells and the interconnections among those cells to compute the dot product—the fundamental building block of the Boltzmann machine—in situ within the memory arrays. This novel capability of the proposed memristive arrays eliminates unnecessary latency, bandwidth, and energy overheads associated with streaming the data out of the memory arrays during computation.

The Boltzmann Machine

The Boltzmann machine, proposed by Geoffrey Hinton and colleagues in 1983,[2] is a well-known example of a stochastic neural network that can learn internal representations and solve combinatorial optimization problems. The Boltzmann machine is a fully connected network comprising two-state units. It employs simulated annealing for transitioning between the possible network states. The units flip their states on the basis of the current state of their neighbors and the corresponding edge weights to maximize a global consensus function, which is equivalent to minimizing the network energy.

Many combinatorial optimization problems, as well as machine learning tasks, can be mapped directly onto a Boltzmann machine by choosing the appropriate edge weights and the initial state of the units within the network. As a result of this mapping, each possible state of the network represents a candidate solution to the optimization problem, and minimizing the network energy becomes equivalent to solving the optimization problem. The energy minimization process is typically performed either by adjusting the edge weights (learning) or recomputing the unit states (searching and classifying). This process is repeated until convergence is reached. The solution to an optimization problem can be found by reading—and appropriately interpreting—the network's final state. For example, Figure 1 depicts the mapping from an example graph with five vertices to a Boltzmann machine with five nodes. The Boltzmann machine is used to solve a Max-Cut problem. Given an undirected graph G with N nodes whose connection weights

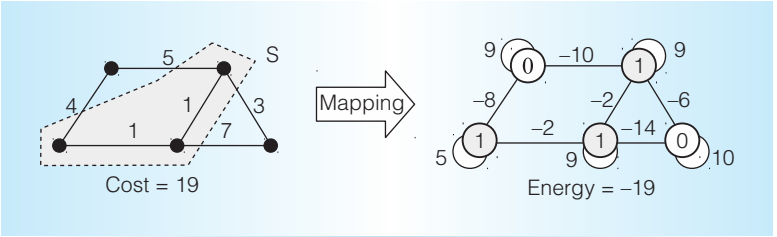

Figure 1. Mapping a Max-Cut problem to the Boltzmann machine model. An example five-vertex undirected graph is mapped and partitioned using a five-node Boltzmann machine.

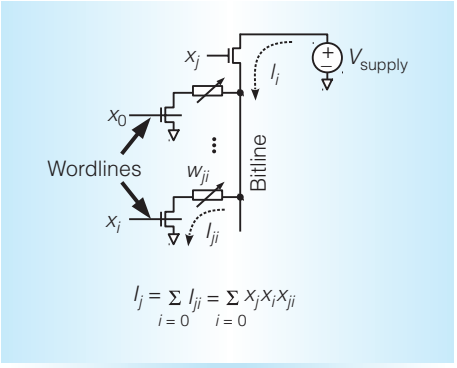

Figure 2. The key concept of in situ computation within memristive arrays. Current summation within every bitline is used to compute the result of a dot product.

(d_{ij}) are represented by a symmetric weight matrix, the maximum cut problem is to find a subset $S \subset \{1, \ldots, N\}$ of the nodes that maximizes $\sum_{i,j} d_{ij}$, in which $i \in S$ and $j \notin S$. To solve the problem on a Boltzmann machine, a one-to-one mapping is established between the graph G and a Boltzmann machine with N processing units. The Boltzmann machine is configured as $w_{jj} = \sum_i d_{ji}$ and $w_{ji} = -2d_{ji}$. When the machine reaches its lowest energy, ($E(x) = -19$), the state variables represent the optimum solution, in which a value of 1 at unit i indicates that the corresponding graphical node belongs to S.

In Situ Computation

The critical computation that the Boltzmann machine performs consists of multiplying a weight matrix W by a state vector $\mathbf{x}$. Every entry of the symmetric matrix W (w_{ji}) records the weight between two units (j and i); every entry of the vector $\mathbf{x}$($\mathbf{x}_i$) stores the state of a single unit (i). Figure 2 depicts the fundamental concept behind the design of the

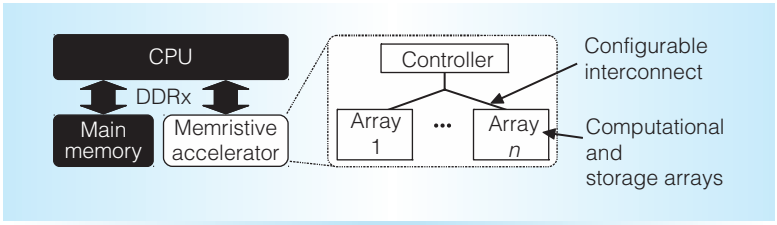

Figure 3. System overview. The proposed accelerator can be selectively integrated in general-purpose computer systems.

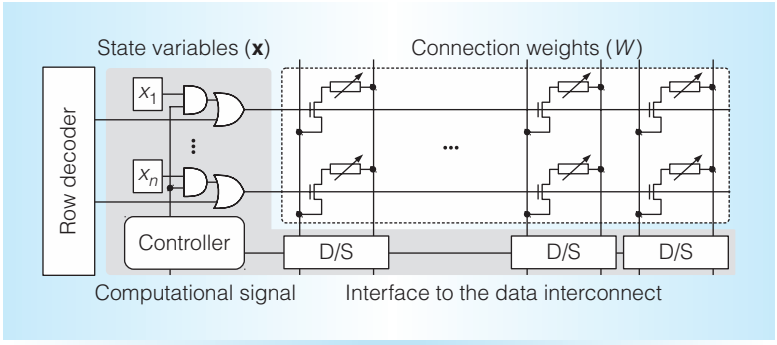

Figure 4. The proposed array structure. The conventional one-transistor, one-memristor (1T-1R) array structure is employed to build the proposed accelerator.

memristive Boltzmann machine. The weights and the state variables are represented using memristors and transistors, respectively. A constant voltage supply (V_{supply}) is connected to parallel memristors through a shared vertical bitline. The total current pulled from the voltage source represents the result of the computation. This current (I_j) is set to zero when x_j is OFF; otherwise, the current is equal to the sum of the currents pulled by the individual cells connected to the bitline.

System Overview

Figure 3 shows an example of the proposed accelerator that resides on the memory bus and interfaces to a general-purpose computer system via the DDRx interface. This modular organization permits the system designers to selectively integrate the accelerator in systems that execute combinatorial optimization and machine learning workloads. The memristive Boltzmann machine comprises a hierarchy of data arrays connected via a configurable interconnection network. A controller implements the interface between the accelerator and the processor. The data arrays can store the weights (w_{ji}) and the state variables (x_i); it is possible to compute the product of weights and state variables in situ within the data arrays. The interconnection network permits the accelerator to retrieve and sum these partial products to compute the final result.

Fundamental Building Blocks

The fundamental building blocks of the proposed memristive Boltzmann machine are storage elements, a current summation circuit, a reduction unit, and a consensus unit. The design of these hardware primitives must strike a careful balance among multiple goals: high memory density, low energy consumption, and in situ, fine-grained parallel computation.

Storage Elements

As Figure 4 shows, the proposed accelerator employs the conventional one-transistor, one-memristor (1T-1R) array to store the connection weights (the matrix W). The relevant state variables (the vector $\mathbf{x}$) are kept close to the data arrays holding the weights. The memristive 1T-1R array is used for both storing the weights and computing the dot product between these weights and the state variables.

Current Summation Circuit

The result of a dot product computation is obtained by measuring the aggregate current pulled by the memory cells connected to a common bitline. Computing the sum of the bit products requires measuring the total amount of current per column and merging the partial results into a single sum of products. This is accomplished by local column sense amplifiers and a bit summation tree at the periphery of the data arrays.

Reduction Unit

To enable the processing of large matrices using multiple data arrays, an efficient data reduction unit is employed. The reduction units are used to build a reduction network, which sums the partial results as they are transferred from the data arrays to the controller. Large matrix columns are partitioned and stored in multiple data arrays, in which the partial sums are individually computed. The reduction network merges the partial

results into a single sum. Multiple such networks are used to process the weight columns in parallel. The reduction tree comprises a hierarchy of bit-serial adders to strike a balance between throughput and area efficiency.

Figure 5 shows the proposed reduction mechanism. The column is partitioned into four segments, each of which is processed separately to produce a total of four partial results. The partial results are collected by a reduction network comprising three bimodal reduction elements. Each element is configured using a local latch that operates in one of two modes: forwarding or reduction. Each reduction unit employs a full adder to compute the sum of the two inputs when operating in the reduction mode. In the forwarding mode, the unit is used for transferring the content of one input upstream to the root.

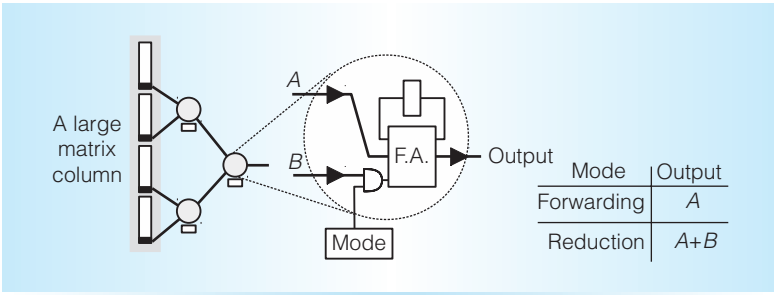

Figure 5. The proposed reduction element. The reduction element can operate in forwarding or reduction mode.

Consensus Unit

The Boltzmann machine relies on a sigmoidal activation function, which plays a key role in both the model's optimization and machine learning applications. A precise implementation of the sigmoid function, however, would introduce unnecessary energy and performance overheads. The proposed memristive accelerator employs an approximation unit using logic gates and lookup tables to implement the consensus function. As Figure 6 shows, the table contains 64 precomputed sample points of the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$, in which x varies between –4 and 4. The samples are evenly distributed on the x-axis. Six bits of a given fixed-point value are used to index the lookup table and retrieve a sample value. The most significant bits of the input data are ANDed and NORed to decide whether the input value is outside the domain [–4, 4]; if so, the sign bit is extended to implement $f(x) = 0$ or $f(x) = 1$; otherwise, the retrieved sample is chosen as the outcome.

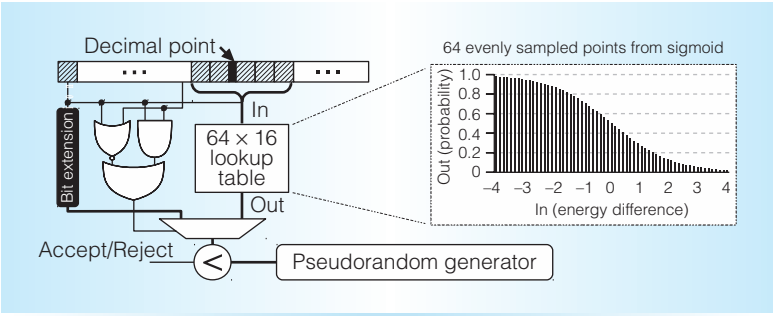

Figure 6. The proposed unit for the activation function. A 64-entry lookup table is used for approximating the sigmoid function.

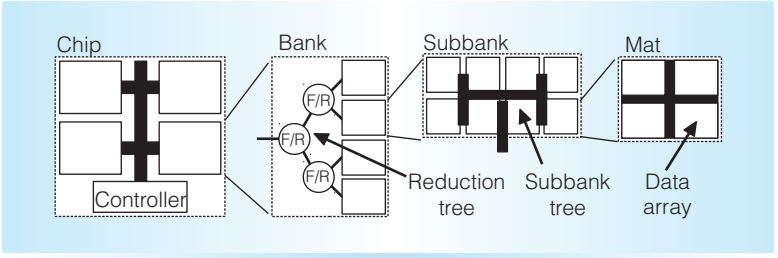

Figure 7. Hierarchical organization of a chip. A chip controller is employed to manage the multiple independent banks.

System Architecture

The proposed architecture for the memristive Boltzmann machine comprises multiple banks and a controller (see Figure 7). The banks operate independently and serve memory and computation requests in parallel. For example, column 0 can be multiplied by the vector **x** at bank 0 while any location of bank 1 is being read. Within each bank, a set of sub-banks is connected to a shared interconnection tree. The bank interconnect is equipped with reduction units to contribute to the dot product computation. In the reduction mode, all sub-banks actively produce the partial results, while the reduction tree selectively merges the results from a subset of the sub-banks. This capability is useful for computing the large matrix columns partitioned across multiple sub-banks. Each sub-bank comprises multiple mats, each of which is composed of a controller and multiple data arrays. The sub-bank tree transfers the data bits between the mats

and the bank tree in a bit-parallel fashion, thereby increasing the parallelism.

Data Organization

To amortize the peripheral circuitry's cost, the data array's columns and rows are time shared. Each sense amplifier is shared by four bitlines. The array is vertically partitioned along the bitlines into 16 stripes, multiples of which can be enabled per array computation. This allows the software to keep a balance between the accuracy of the computation and the performance for a given application by quantizing more bit products into a fixed number of bits.

On-Chip Control

The proposed hardware can accelerate optimization and deep learning tasks by appropriately configuring the on-chip controller. The controller configures the reduction trees, maps the data to the internal resources, orchestrates the data movement among the banks, performs annealing or training tasks, and interfaces to the external bus.

DIMM Organization

To solve large-scale optimization and machine learning problems whose state spaces do not fit within a single chip, we can interconnect multiple accelerators on a DIMM. Each DIMM is equipped with control registers, data buffers, and a controller. This controller receives DDRx commands, data, and address bits from the external interface and orchestrates computation among all of the chips on the DIMM.

Software Support

To make the proposed accelerator visible to software, we memory map its address range to a portion of the physical address space. A small fraction of the address space within every chip is mapped to an internal RAM array and is used to implement the data buffers and configuration parameters. Software configures the on-chip data layout and initiates the optimization by writing to a memory mapped control register.

Evaluation Highlights

We modify the SESC simulator[3] to model a baseline eight-core out-of-order processor. The memristive Boltzmann machine is interfaced to a single-core system via a single DDR3-1600 channel. We develop an RRAM-based processing-in-memory (PIM) baseline. The weights are stored within data arrays that are equipped with integer and binary multipliers to perform the dot products. The proposed consensus units, optimization and training controllers, and mapping algorithms are employed to accelerate the annealing and training processes. When compared to existing computer systems and GPU-based accelerators, the PIM baseline can achieve significantly higher performance and energy efficiency because it eliminates the unnecessary data movement on the memory bus, exploits data parallelism throughout the chip, and transfers the data across the chip using energy-efficient reduction trees. The PIM baseline is optimized so that it occupies the same area as that of the memristive accelerator.

Area, Delay, and Power Breakdown

We model the data array, sensing circuits, drivers, local array controller, and interconnect elements using Spice predictive technology models[4] of *n*-channel and *p*-channel metal-oxide semiconductor transistors at 22 nm. The full adders, latches, and control logic are synthesized using FreePDK[5] at 45 nm. We first scale the results to 22 nm using scaling parameters reported in prior work,[6] and then scale them using the fan-out of 4 (FO4) parameters for *International Technology Roadmap for Semiconductors* low-standby-power (LSTP) devices to model the impact of using a memory process on peripheral and global circuitry.[7,8] We use McPAT[9] to estimate the processor power.

Figure 8 shows a breakdown of the computational energy, leakage power, computational latency, and die area among different hardware components. The sense amplifiers and interconnects are the major contributors to the dynamic energy (41 and 36 percent, respectively). The leakage is caused mainly by the current summation circuits (40 percent) and other logic (59 percent), which includes the charge pumps, write drivers, and controllers. The computation latency, however, is due mainly to the interconnects (49 percent), the wordlines, and the bitlines (32 percent).

Notably, only a fraction of the memory arrays must be active during a computational operation. A subset of the mats within each bank performs current sensing of the bitlines; the partial results are then serially streamed to the controller on the interconnect wires. The experiments indicate that a fully utilized accelerator integrated circuit (IC) consumes 1.3 W, which is below the peak power rating of a standard DDR3 chip (1.4 W).

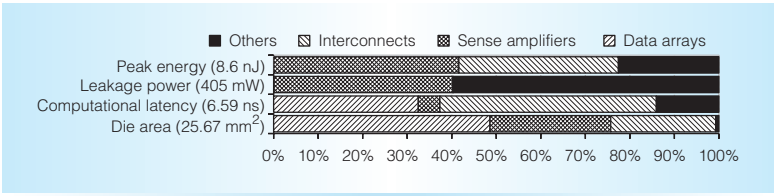

Figure 8. Area, delay, and power breakdown. Peak energy, leakage power, computational latency, and die area are estimated at the 22-nm technology node.

Performance

Figure 9 shows the performance on the proposed accelerator, the PIM architecture, the multicore system running the multithreaded kernel, and the single-core system running the semidefinite programing (SDP) and MaxWalkSAT kernels. The results are normalized to the single-threaded kernel running on a single core. The results indicate that the single-threaded kernel (Boltzmann machine) is faster than the baselines (SDP and MaxWalkSAT heuristics) by an average of 38 percent. The average performance gain for the multithreaded kernel is limited to 6 percent, owing to significant state update overheads. PIM outperforms the single-threaded kernel by 9.31 times. The memristive accelerator outperforms all of the baselines (57.75 times speedup over the single-threaded kernel and 6.19 times over PIM). Moreover, the proposed accelerator performs the deep learning tasks 68.79 times faster than the single-threaded kernel and 6.89 times faster than PIM.

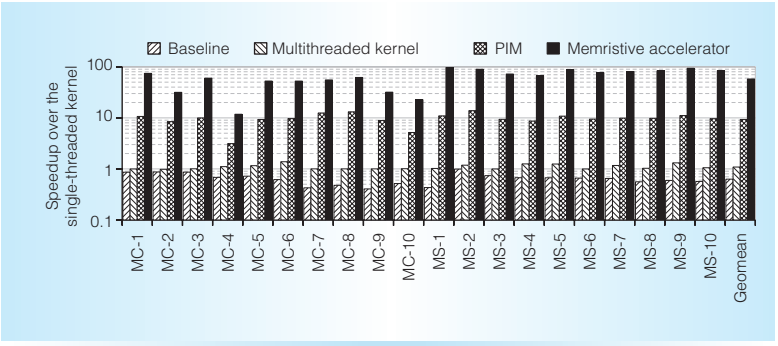

Figure 9. Performance on optimization. Speedup of various system configurations over the single-threaded kernel.

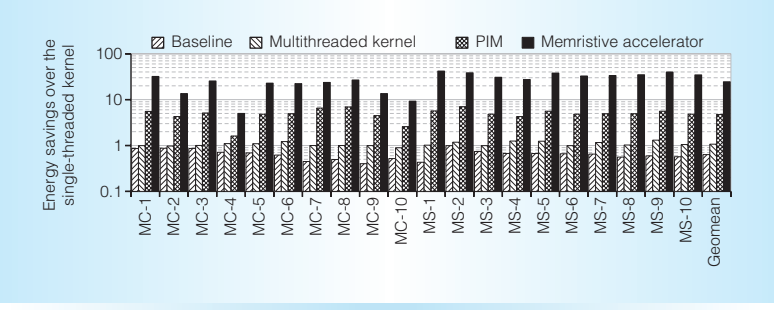

Figure 10. Energy savings on optimization. Energy savings of various system configurations over the single-threaded kernel.

Energy

Figure 10 shows the energy savings as compared to PIM, the multithreaded kernel, SDP, and MaxWalkSAT. On average, energy is reduced by 25 times as compared to the single-threaded kernel implementation, which is 5.2 times better than PIM. For the deep learning tasks, the system energy is improved by 63 times, which is 5.3 times better than the energy consumption of PIM.

Sensitivity to Process Variations

Memristor parameters can deviate from their nominal values, owing to process variations caused by line edge roughness, oxide thickness fluctuation, and random discrete doping. These parameter deviations result in cycle-to-cycle and device-to-device variabilities. We evaluate the impact of cycle-to-cycle variation on the computation's outcome by considering a bit error rate of 10^{-5} in all of the simulations, along the lines of the analysis provided in prior work.[10] The proposed accelerator successfully tolerates such errors, with less than a 1-percent change in the outcome as compared to a perfect software implementation.

The resistance of RRAM cells can fluctuate because of the device-to-device variation, which can impact the outcome of a column summation—that is, a partial dot product.

We use the geometric model of memristance variation proposed by Miao Hu and colleagues[11] to conduct Monte Carlo simulations for 1 million columns, each comprising 32 cells. The experiment yields two distributions for low resistance (R_{LO}) and high resistance (R_{HI}) samples that are then approximated by normal distributions with respective standard deviations of 2.16 and 2.94 percent (similar to the prior work by Hu and colleagues). We then find a bit pattern that results in the largest summation error for each column. We observe up to 2.6×10^{-6} deviation in the column conductance, which can result in up to 1 bit error per summation. Subsequent simulation results indicate that the accelerator can tolerate this error, with less than a 2 percent change in the outcome quality.

Finite Switching Endurance

RRAM cells exhibit finite switching endurance ranging from 1e6 to 1e12 writes. We evaluate the impact of finite endurance on an accelerator module's lifetime. Because wear is induced only by the updating of the weights stored in memristors, we track the number of times that each weight is written. The edge weights are written once in optimization problems and multiple times in deep learning workloads. (Updating the state variables, stored in static CMOS latches, does not induce wear on RRAM.) We track the total number of updates per second to estimate the lifetime of an eight-chip DIMM. Assuming endurance parameters of 1e6 and 1e8 writes, the respective module lifetimes are 3.7 and 376 years for optimization and 1.5 and 151 years for deep learning.

Data movement between memory cells and processor cores is the primary contributor to power dissipation in computer systems. A recent report by the US Department of Energy identifies the power consumed in moving data between the memory and the processor as one of the 10 most significant challenges in the exascale computing era.[12] The same report indicates that by 2020, the energy cost of moving data across the memory hierarchy will be orders of magnitude higher than the cost of performing a double-precision floating-point operation.

Emerging large-scale applications such as combinatorial optimization and deep learning tasks are even more influenced by memory bandwidth and power problems. In these applications, massive datasets have to be iteratively accessed by the processor cores to achieve a desirable output quality, which consumes excessive memory bandwidth and system energy. To address this problem, numerous software and hardware optimizations using GPUs, clusters based on message passing interface (MPI), field-programmable gate arrays, and application-specific integrated circuits have been proposed in the literature. These proposals focus on energy-efficient computing with reduced data movement among the processor cores and memory arrays. These proposals' performance and energy efficiency are limited by read accesses that are necessary to move the operands from the memory arrays to the processing units. A memory subsystem that allows for in situ computation within its data arrays could address these limitations by eliminating the need to move raw data between the memory arrays and the processor cores.

Designing a platform capable of performing in situ computation is a significant challenge. In addition to storage cells, extra circuits are required to perform analog computation within the memory cells, which decreases memory density and area efficiency. Moreover, power dissipation and area consumption of the required components for signal conversion between analog and digital domains could become serious limiting factors. Hence, it is critical to strike a careful balance between the accelerator's performance and complexity.

The memristive Boltzmann machine is the first memory-centric accelerator that addresses these challenges. It provides a new framework for designing memory-centric accelerators. Large scale combinatorial optimization problems and deep learning tasks are mapped onto a memory-centric, non-Von Neumann computing substrate and solved in situ within the memory cells, with orders of magnitude greater performance and energy efficiency than contemporary supercomputers. Unlike PIM-based accelerators, the proposed accelerator enables computation within conventional data arrays to achieve the

energy-efficient and massively parallel processing required for the Boltzmann machine model.

We expect the proposed memory-centric accelerator to set off a new line of research on in situ approaches to accelerate large-scale problems such as combinatorial optimization and deep learning tasks and to significantly increase the performance and energy efficiency of future computer systems. MICRO

Acknowledgments

This work was supported in part by NSF grant CCF-1533762.

References

1. E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, 1989.
2. S.E. Fahlman, G.E. Hinton, and T.J. Sejnowski, "Massively Parallel Architectures for AI: NETL, Thistle, and Boltzmann Machines," *Proc. Assoc. Advancement of AI* (AAAI), 1983, pp. 109–113.
3. J. Renau et al., "SESC Simulator," Jan. 2005; http://sesc.sourceforge.net.
4. W. Zhao and Y. Cao, "New Generation of Predictive Technology Model for Sub-45nm Design Exploration," *Proc. Int'l Symp. Quality Electronic Design*, 2006, pp. 585–590.
5. "FreePDK 45nm Open-Access Based PDK for the 45nm Technology Node," 29 May 2014; www.eda.ncsu.edu/wiki/FreePDK.
6. M.N. Bojnordi and E. Ipek, "Pardis: A Programmable Memory Controller for the DDRX Interfacing Standards," *Proc. 39th Ann. Int'l Symp. Computer Architecture* (ISCA), 2012 pp. 13–24.
7. N.K. Choudhary et al., "Fabscalar: Composing Synthesizable RTL Designs of Arbitrary Cores Within a Canonical Superscalar Template," *Proc. 38th Ann. Int'l Symp. Computer Architecture*, 2011, pp. 11–22.
8. S. Thoziyoor et al., "A Comprehensive Memory Modeling Tool and Its Application to the Design and Analysis of Future Memory Hierarchies," *Proc. 35th Int'l Symp. Computer Architecture* (ISCA), 2008, pp. 51–62.
9. S. Li et al., "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," *Proc. 36th Int'l Symp. Computer Architecture* (ISCA), 2009, pp. 468–480.
10. D. Niu et al., "Impact of Process Variations on Emerging Memristor," *Proc. 47th ACM/IEEE Design Automation Conf.* (DAC), 2010, pp. 877–882.
11. M. Hu et al., "Geometry Variations Analysis of Tio 2 Thin-Film and Spintronic Memristors," *Proc. 16th Asia and South Pacific Design Automation Conf.*, 2011, pp. 25–30.
12. *The Top Ten Exascale Research Challenges*, tech. report, Advanced Scientific Computing Advisory Committee Subcommittee, Dept. of Energy, 2014.

Mahdi Nazm Bojnordi is an assistant professor in the School of Computing at the University of Utah. His research focuses on computer architecture, with an emphasis on energy-efficient computing. Nazm Bojnordi received a PhD in electrical engineering from the University of Rochester. Contact him at bojnordi@cs.utah.edu.

Engin Ipek is an associate professor in the Department of Electrical and Computer Engineering and the Department of Computer Science at the University of Rochester. His research interests include energy-efficient architectures, high-performance memory systems, and the application of emerging technologies to computer systems. Ipek received a PhD in electrical and computer engineering from Cornell University. He has received the 2014 IEEE Computer Society TCCA Young Computer Architect Award, two *IEEE Micro* Top Picks awards, and an NSF CAREER award. Contact him at ipek@ece.rochester.edu.