Emerging Memristor Technology Enabled Next Generation Cortical Processor

(Invited Paper)

Hai (Helen) Li, Miao Hu, Xiaoxiao Liu, Mengjie Mao
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA 15261, USA
Email: {hal66, mih73, xil116, mem231}@pitt.edu

Chuandong Li and Shukai Duan
School of Electronics and Information Engineering
Southwest University
Chongqing 400715, China
Email: {cdli, duansk}@swu.edu.cn

***Abstract*—The explosion of "big data" applications imposes severe challenges of data processing speed and scalability on traditional computer systems. However, the performance of von Neumann machine is greatly hindered by the increasing performance gap between CPU and memory ("known as memory wall"), motivating the active research on new or alternative computing architecture. As one important instance, neuromorphic computing systems have gained considerable attentions. Neuromorphic computing systems refer to the computing architecture inspired by the working mechanism of human brains. The human neocortex system naturally possesses a massively parallel architecture with closely coupled memory and computing as well as the unique analog domain operations. By imitating such structure, neuromorphic computing system is anticipated to be superior to the conventional computer systems in image recognition and natural language understanding. Among all the possible solutions, cortical processor has gained significant attention. The cortical-like hierarchical model conducts data processing by using spatial and temporal evolution of the data representation to form relationships. The straightforward hardware realization of such massively parallel algorithms inspired by cortical models, however, commonly consumes a large volume of memory and computing resources, incurring high design complexity and hardware cost. Here, we suggest realizing the cortical processor by combining the flexibility of conventional architecture in computation and the efficiency of the emerging memristor technology. Computing accelerator is introduced to accelerate neuromorphic computations with ultra-low energy consumption. The computation and data exchange are carefully coordinated and supported by a hierarchical network-on-chip across digital and analog domains.**

I. Introduction

The explosion of "big data" applications imposes severe challenges of data processing speed and scalability on traditional computer systems. However, the performance of von Neumann machine is greatly hindered by the increasing performance gap between CPU and memory, which is well known as *memory wall*. Such a situation motivates the active research on new or alternative computing architecture. As one important instance, neuromorphic computing systems have gained considerable attentions.

Neuromorphic computing systems refer to the computing architecture inspired by the working mechanism of human brains. The human neocortex system naturally possesses a massively parallel architecture with closely coupled memory and computing as well as the unique analog domain operation [1]. The simple unified building blocks *(i.e.*, neurons) follow integrate-and-fire mechanisms, leading to an ultra-high computing performance beyond 100 TFLOPs (*Trillion FLoating-point Operations Per Second*) and a power consumption of mere 20 Watt. By imitating such structure, neuromorphic computing system is anticipated to be superior to the conventional computer systems in image recognition and natural language understanding. As the most resource-consuming part in neuromorphic algorithms [2], matrix operations are normally processed by hardware accelerators like CPU/GPU/FPGA [3] or VLSI circuits [4–6]. The straightforward hardware realization of neural networks, however, commonly consumes a large volume of memory and computing resources, incurring high design complexity and hardware cost [7, 8].

The structural similarity makes the reconfigurable array conceptually efficient for matrix operations [9–12] and inspired many researches on the corresponding circuit designs, *i.e.*, the arrays of flash transistor [13, 14] or DRAM capacitor [15]. However, the computation capacity and scalability of these designs are generally limited by the large cell footprint. Recently, the discovery of memristor device triggered a revolution in neuromorphic computing system design: synaptic behavior is easily mimicked by the historical recording property of the memristor while the crossbar array structure offers the highest integration density in 2/3-D design [16–18].

Here, we suggest realizing the cortical processor by combining the flexibility of conventional architecture in computation and the efficiency of the emerging memristor technology. Computing accelerator is introduced to accelerate neuromorphic computations with ultra-low energy consumption. The computation and data exchange are carefully coordinated and supported by a hierarchical network-on-chip across digital and analog domains.

II. Background

A. Neural Network and Neuromorphic Systems

Fig. 1 illustrates a simple example of a neural network, in which two groups of neurons are connected by a set of synapses. We define $a_{i,j}$ as the synaptic strength of the synapse connecting the j^{th} neuron in the input group and the i^{th} neuron in the output one. The relationship of the activity patterns **F**

978-1-4799-3378-5/14/$31.00 ©2014 IEEE

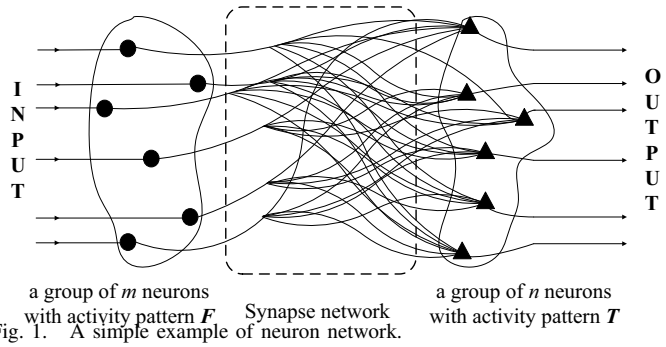

Fig. 1. A simple example of neuron network.

of input neurons and **T** of output neurons can be described in a matrix form:

$$\mathbf{T} = \mathbf{AF}, \tag{1}$$

where matrix **A**, denoted as a connection matrix, consists of the synaptic strengths between the two neuron groups. The matrix-vector multiplication of Eq. (1) is a common operation in neural networks to model functionally associated neurons.

Neuromorphic systems are based on algorithms and techniques inspired by the principles of information processing in biological nervous systems. These systems are believed to have high robustness and adaptability for many cognitive applications. For instance, an object recognition software was developed to detect features with scale and position tolerance [19]. and a hardware accelerator has been developed for it to speed up the image processing at lower level [20].

Some works on neuromorphic hardware have been published recently. A representative example is the digital neurosynaptic core from IBM [21], which described a 1024×256 crossbar memory imitating 218 synapses. This is the first large-scale hardware that emulates a neurological system. However, it has considerably large size because each memory node on the crossbar is an SRAM cell. Only binary synapses are implemented in the prototype system while the crossbar is only a storage element. The data still need to be moved to adders or multipliers for calculation. This system has been used for character image detection. But how to apply it for larger scale problem is not clear. In [22], a hardware design is proposed for spiky neuron emulation where only single neuron is discussed. Some other works simulated the structure of the brain for medical analysis: a cortex system with one billion neurons and 10 trillion synapses has been simulated on IBM Blue Gene supercomputer [23]. Although the model captures the detailed physics of the neural system, how to evolve it to have the intelligence to solve real-life problem is still unknown.

B. Memristor and Crossbar Array

Many materials have demonstrated memristive behavior in theory and/or by experiments under different mechanisms. In 2008, HP Lab demonstrated the first memristive device, in which the memristive effect was achieved by moving the doping front within a TiO_2 thin-film [24].

Fig. 2 illustrates the conceptual view of the TiO_2 thin-film memristor and the corresponding variable resistor model, which is equivalent to two serially-connected resistors. Here, R_L and R_H respectively denote the *low resistance state* (LRS)

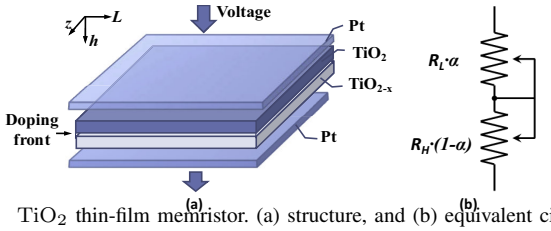

Fig. 2. TiO_2 thin-film memristor. (a) structure, and (b) equivalent circuit.

and the *high resistance state* (HRS). The overall memristance can be expressed as:

$$M(p) = p \cdot R_H + (1-p) \cdot R_L, \tag{2}$$

where p ($0 \le p \le 1$) is the relative doping front position, which is the ratio of doping front position over the total thickness of the TiO_2 thin-film. The velocity of doping front movement $v(t)$, driven by the voltage applied across the memristor $V(t)$, can be expressed as:

$$v(t) = \frac{dp(t)}{dt} = \mu_v \cdot \frac{R_L}{h^2} \cdot \frac{V(t)}{M(p)}, \tag{3}$$

where μ_v is the equivalent mobility of dopants, h is the total thickness of the thin film, and $M(p)$ is the total memristance when the relative doping front position is p. In general, a certain energy (or threshold voltage) is required to enable the state change in a memristive device. When the electrical excitation through a memristor is greater than the threshold voltage, i.e., $V(t) > V_{th}$, the memristance changes (in training). Otherwise, a memristor behaves like a resistor.

Crossbar array illustrated in Fig. 3 is a typical structure of memristor based memories. It employs a memristor device at each intersection of horizontal and vertical metal wires without any selectors [25, 26]. The memristor crossbar array is naturally attractive for implementation of connection matrix in neural networks because it can provide a large number of signal connections within a small footprint and conduct the weighted combination of input signals [27].

III. Matrix vs. Crossbar Array

A. Mapping a Connection Matrix to a Crossbar Array

Let's use the N-by-N memristor crossbar array illustrated in Fig. 3 to demonstrate its matrix computation functionality. Here, we apply a set of input voltages $\mathbf{V_I^T} = [V_{I,1}, V_{I,2}, , V_{I,N}]$ on the *word-lines* (WL) of the array, and collect the current through each *bit-line* (BL) by measuring the voltage across a sensing resistor. The same sensing resistors are used on

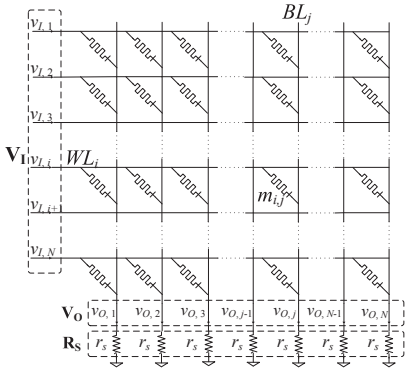

Fig. 3. A memristor crossbar array.

all BLs with resistance r_s, or conductance $g_s = 1/r_s$. The output voltage vector $\mathbf{V_O}^T = [V_{O,1}, V_{O,2}, , V_{O,N}]$. Assume the memristor sitting on the connection between WL_i and BL_j has a memristance of $m_{i,j}$. The corresponding conductance $g_{i,j} = 1/m_{i,j}$. Then, the relation between the input and output voltages can be represented by:

$$\mathbf{V_O} = \mathbf{C}\mathbf{V_I}. \tag{4}$$

Here, matrix $\mathbf{C}$ can be represented by the memristors' conductance and the load resistors as:

$$\mathbf{C} = \mathbf{D}\mathbf{G}^T = \mathrm{diag}(d_1, \ldots, d_N) \begin{bmatrix} g_{1,1} & \cdots & g_{1,N} \\ \vdots & \ddots & \vdots \\ g_{N,1} & \cdots & g_{N,N} \end{bmatrix}, \tag{5}$$

where $d_i = 1/(g_s + \Sigma_{i=1}^{N} g_{i,j})$. To differentiate the mathematical connection matrix $\mathbf{A}$ in neural network, we use $\mathbf{C}$ to describe the physical relation between $\mathbf{V_I}$ and $\mathbf{V_O}$. Thus, all the terms in $\mathbf{C}$ must be positive values. Please note that some non-iterative neuromorphic hardware uses the output currents $\mathbf{I_O}$ as output signals.

B. A Fast Approximation Function

Eq. (4) indicates that a trained memristor crossbar array can be used to construct the positive matrix $\mathbf{C}$, and transfer the input vector $\mathbf{V_I}$ to the output vector $\mathbf{V_O}$. However, $\mathbf{C}$ is not a direct one-to-one mapping of conductance matrix $\mathbf{G}$ as indicated in Eq. (5). Though a numerical iteration method can be used to obtain the exact mathematical solution of $\mathbf{G}$, it is too complex and hence impractical when frequent updates are needed.

For simplification, assume $g_{i,j} \in G$ satisfies $g_{min} \leq g_{i,j} \leq g_{max}$, where g_{min} and g_{max} respectively represent the minimum and the maximum conductance of all the memristors in the crossbar array. Thus, a simpler and faster approximation solution to the mapping problem is defined as:

$$g_{j,i} = c_{i,j} \cdot (g_{max} - g_{min}) + g_{min}. \tag{6}$$

A decayed version of $\mathbf{C}$, which is $\hat{\mathbf{C}}$ can be approximately mapped to the conductance matrix $\mathbf{G}$ of the memristive array. Plugging Eq. (6) into Eq. (5), we have:

$$\hat{c}_{i,j} = \frac{c_{i,j} \cdot (g_{max} - g_{min}) + g_{min}}{g_s + (g_{max} - g_{min} \cdot \Sigma_{j=1}^{N} + N \cdot g_{min})}. \tag{7}$$

Note that many memristive materials, such as TiO_2, demonstrate a large g_{max}/g_{min} ratio [24]. Thus, a memristor at the high resistance state under a low voltage excitation can be regarded as an insulator, that is, $g_{min} \approx 0$. Many neuromorphic systems demonstrate small $\Sigma_{j=1}^{N} c_{i,j}$ [10, 12]. As such, the term $\Sigma_{j=1}^{N} c_{i,j}$ can be further reduced by increasing the ratio g_s/g_{max}. As a result, the impact of $\Sigma_{j=1}^{N} c_{i,j}$ can be ignored. These two facts indicate that Eq. (7) can be further simplified as:

$$\hat{c}_{i,j} = c_{i,j} \cdot g_{max}/g_s. \tag{8}$$

In summary, with the proposed fast approximation function Eq. (6), the memristor crossbar array performs as a decayed matrix $\hat{\mathbf{C}}$ between the input and output voltage signals.

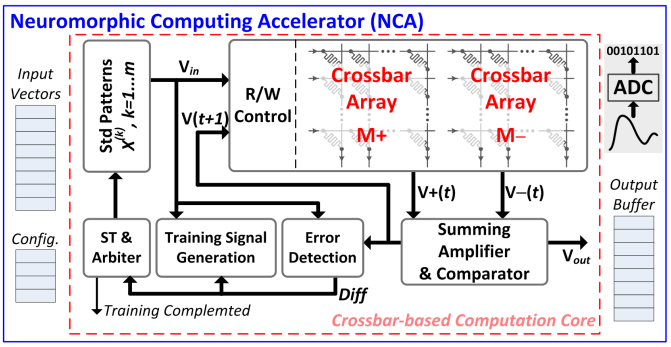

Fig. 4. The NCA architecture.

IV. Memristor Enabled Neuromorphic System

Therefore, we propose memristor crossbar-based *neuromorphic computing accelerator* (NCA) for matrix computations, which can be regarded as a *processing element* (PE) in an *analog Network-on-Chip* (A-NoC) systems.

A. Neuromorphic Computing Accelerator

Fig. 4 is an overview of the proposed NCA architecture. Crossbar arrays conduct matrix-vector multiplications in normal operation. Because the device resistance is always larger than zero, two crossbar arrays **M+** and **M–** are required to represent the positive and negative terms, respectively; Summing amplifiers conduct vector computations at the outputs of the crossbars, such as scaling and summation. The voltage signal generated by the summing amplifiers $\mathbf{V}_{(t+1)}$ is either sent out of the computing module, or fed back to the inputs of crossbar arrays if more iteration is needed.

The modified extend Delta rule [12] is used for on-chip programming of crossbar arrays. Delta rule is used to find the weights that minimize the squared error between a target output pattern and the input prototype patterns. Since fully implementing Delta rule demands high computation, we propose to simplify the weight updating function to

$$\Delta w_{i,j} = \alpha \cdot \mathrm{sgn}[V'_{(t+1),j} - V_{(t+1),j}] \cdot \mathrm{sgn}[V_{(t),i}]. \tag{9}$$

Here, $V'_{(t+1),j}$ and $V_{(t+1),j}$ are the target value and the observed value at j^{th} entry of output vector $\mathbf{V}_{(t+1)}$, respectively; $V_{(t),i}$ is the i^{th} input value; and α is the learning rate. The changing rate of the weights for each training step is controlled by the learning rate α, which can be adjusted in the circuit implementation for the best training performance. This method ensures the weight change is in the same direction as that in the original Delta rule algorithm by paying much less design cost. The according hardware realization requires those components in normal operation for $\mathbf{V}_{(t+1)}$) generation, the error detection block to detect weight change directions, and the training signal generation block to integrate them together.

I/O interface of the memristor crossbar-based NCA includes input/output buffers and configuration queue, which carries the information required for crossbar array programming etc. Since the default operating data type for the NCA is analog, the input/output buffers are able to retain analog data, *e.g.*, by using the arbitrary state of the memristor devices. As we shall show in the following sub-section, the data communication among the different NCAs will be managed by

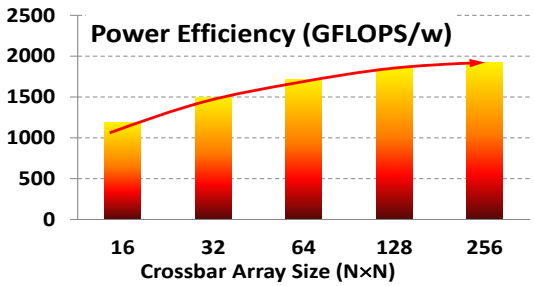

Fig. 5. Power Efficiency of NCA.

a novel analog while the analog-digital converters only exist at the interface between the NCA array and the conventional pipeline.

The proposed NCA is much faster and more energy efficient than the conventional computing architecture. The major component of the operation latency of an NCA comes from the interconnects of crossbar arrays and peripheral circuits. For instance, to complete one iteration in a *brain-in-a-state-box* (BSB) recall operation for a 256-entry vector, the computation latency of an NCA should not exceed 100ns even we conservatively assume the delay of an summing amplifier is 50ns (including setup and computation). As a comparison, the delay of the Boolean mapping is 750ns by assuming using 256 4-bit multipliers with 2ns latency and 256 4-bit CSLA adders with 1ns latency. Fig. 5 shows the estimated power efficiency of memristor-based NCA at 65nm technology node. The NCA design with a small crossbar array, *i.e.*, 16×16, already obtains 1200 GFLOPS/W. Increasing the crossbar size to 256×256 results in a higher computation parallelism, further boosting the power efficiency close to 2 TFLOPS/W.

B. A Heterogeneous Computing Platform

Fig. 6 demonstrates the system implementation of the heterogeneous architecture. Every general purpose processor is augmented with one NCA. The operations of the NCA are triggered at different pipeline stages (*e.g.*, execute or write back) depending on the NCA instruction type. Within a NCA, three FIFO queues buffer the input/output data and the configurations of the NCA, respectively.

Since the computation of the NCA is conducted in analog form, the dequeued/enqueued data from/to the In-queue/Out-queue must be converted to analog/digital signals by DAC/ADC, respectively. Such a structure ensures a high reconfigurability to support different *artificial neural network* (ANN) topologies. Compared to the existing ANN accelerators built with digital circuitry, our proposed mixed-signal NCA design offers three unique features, including truly data flow driven execution, intrinsic high prallelis, and scalable computation-in-memory model.

Note that *analog Network-on-Chip* (A-NoC) is introduced to support the *multilayer perception* (MLP) mappings and the data migration among the memristor crossbar arrays. Since the computation of memristor crossbar arrays is in analog form, the adoption of A-NoC can eliminates the costly AD/DA conversions during data transmission. In the A-NoC, two types of routers – central router and group router, are in charge of inter-group and intra-group data communication, respectively. Each group router has digital control logic and up to 7 analog

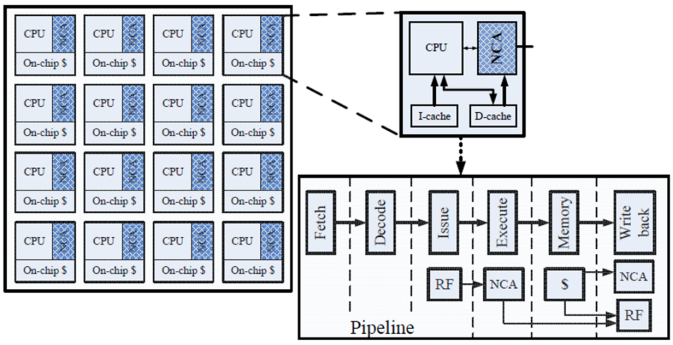

Fig. 6. An overview of the heterogeneous architecture.

bi-direction ports connecting with 3 neighbor routers and 4 local memristor crossbar arrays. The central router shares the same architecture but only connects with the CPU and other group routers. This centralized architecture maximizes the number of parties that each router communicates with, minimizes the effective communication distance as well as the hop count, and simplifies the control complexity.

During the operation of the NCA, the input analog signals first go to the central router after passing through the DAC and then are sent to the group router. The group router then directs the signals to the destined memristor crossbar array. The output analog signals generated from a memristor crossbar array will be sent to the local group router first, and then are routed to the next destination. The final output signals are always sent to the central router and converted to digital signals by the ADC before being buffered in the Out-queue.

V. Evaluation

A. Experimental setup

We select nine representative applications with different computation characteristics to evaluate our proposed heterogeneous computing architecture. Four applications (`building`, `gene`, `mushroom` and `thyroid`) are selected from Proben1 [28], a collection from UCI machine learning repository [29] tailored for neural network implementation; `MNIST` [30] is a widely used data set in learning and recognition algorithm; *canny edge detector* (`CED`) [31] is an image edge detector commonly used in image processing and computer vision; `kmeans` is a popular clustering algorithm in data mining; `fft` is a signal processing algorithm; `blackscholes` is a financial application originally from PARSEC and its target codes have been implemented by ANN.

We refer to the first five applications as learning applications and the rest as imperative applications. Learning applications naturally come with training and testing inputs so that no special efforts are required to prepare them for ANN implementation. Imperative applications are programmed by imperative language where only some codes can be re-implemented by ANN. The application inputs for `fft` and `kmeans` are generated randomly and their target codes are evoked many times during executions. We first run these two applications once with instrumented codes to collect the training inputs. We run `CED` five times with totally different input images to collect training inputs for the target codes, and choose a new image as test inputs. Note that there are two pieces of target codes in `CED`. `blackscholes` from

TABLE I. NCA PARAMETERS USED IN SIMULATIONS

Memristor	R_L 200Ω		R_H $160K\Omega$		V_{th} $2V$
Neuron & Network power/delay ($\mu W/ns$)	V_{dd} $1.0V$	op-amp $100/0.60$		setup path in router $0.72/0.42$	
	sigmoid $10/0.24$			crossbar $0.69/3$	
ADC (mW/GHz)	8-bit 290.59/1	7-bit 79.31/1.25	6-bit 21.76/1.25	5-bit 10.18/1.5	4-bit 2.77/1.5
DAC (mW/GHz)	8-bit 7.32/1	7-bit 5.88/1.25	6-bit 1.72/1.25	5-bit 0.68/1.5	4-bit 0.19/1.5

benchNN [32] already provides cross validation training and testing inputs for the target codes.

We use Cadence to build NCA circuit components, including analog buffer, switch, sum amplifier, and sigmoid circuit, based on $45nm$ PTM model [33]. The power and timing parameters of different NCA components are characterized with SPICE. We also design ADC/DAC (*analog digital converter/digital analog converter*) with Cadence and extract their design parameter based on SPICE simulations. The memristor parameters are adopted from [34] and carefully scaled down to $45nm$. The NCA design parameters are summarized in TABLE I.

For performance and energy evaluation, we modify MacSim [35], a PIN-based [36] cycle-level x86 simulator, by adding a cycle-accurate NCA module to conduct the architecture level experiments. To generate the NCA instructions of a target code, we pack the whole target code into one function. During trace generation, the modified PIN tool generates the simulation trace by replacing the function with the corresponding NCA instructions according to the selected ANN topology in specific applications.

We use McPAT [37] to evaluate the energy consumption of the CPU core. We generate a detail log of NCA utilization during the execution and the circuit synthesis result of the NCA components is used to estimate the energy consumptions of the NCA. The estimated results are validated by the results generated from booksim simulator [38], which is modified to model the NCA traffic of each application.

B. Performance and Energy

Fig. 7 shows the performance speedups and energy reductions of the proposed heterogeneous architecture, which are normalized to the same CPU architecture without NCA acceleration (*i.e.*, learning applications are running exclusively on the CPU by using FANN library to simulate ANN topology). The *geometric mean speedup* (GMS) of learning applications achieved is $\sim 81.2\times$, as shown in Fig. 7(a). Note that the CPU still conducts some tasks, *e.g.*, the post data processing after the NCA in learning applications. The GMS of imperative applications, however, is only $\sim 2.82\times$. The reason for such small speedup is because the target codes in the selected imperative applications have already been deeply optimized to achieve a high ILP (*instruction level parallelism*) for CPU running.

The corresponding energy savings of all applications are shown in Fig. 7(b). On average, $86.8\times$ energy reduction is achieved in learning applications while the one achieved in imperative applications is $3.23\times$, compared to the pure CPU

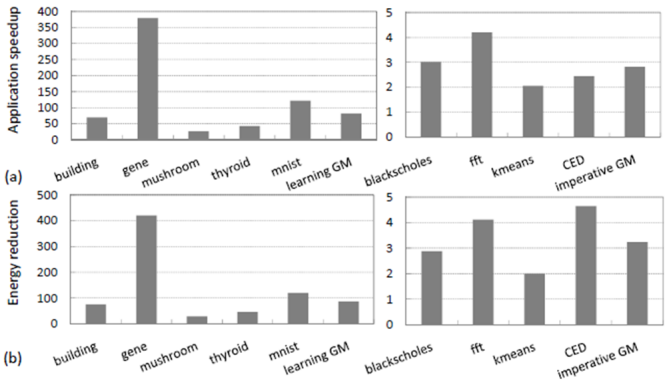

Fig. 7. (a) Application speedups and (b) Energy reductions with NCA over CPU execution..

running. Across all applications, the proposed HCS achieves $18.2\times$ performance speedup and $20.1\times$ energy reduction w.r.t. the conventional CPU without ANN acceleration.

VI. CONCLUSION

In this work, we attempt to combine the flexibility of conventional architecture in computation and the efficiency of the emerging memristor technology. As such, the realization of cortical models that commonly consumes a large volume of memory and computing resources becomes possible. More specific, we propose a heterogeneous computing system, which contains a memristor-based neuromorphic computing accelerator (NCA) tightly coupled to general purpose processor. Compared to conventional general purpose processor, the proposed heterogeneous computing system can achieve on average $18.2\times$ performance speedup and $20.1\times$ energy reduction over the simulated 9 learning and imperative applications. The computation accuracy degradation incurred by the mixed-signal ANN acceleration in the NCA is constrained within an acceptable range. The high computation and energy efficiency of the proposed system mainly come from the high-throughput of the mixed-signal NCA computation, the excellent reconfigurability of the hierarchical memristor crossbar array structure, and the low data transmission overhead on the analog NoC.

ACKNOWLEDGMENT AND DISCLAIMER

This work was supported in part by DARPA D13AP00042, NSF CCF-1337198, and CNS-1342566. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, NSF, or their contractors.

REFERENCES

[1] H. Moravec, "When will Computer Hardware Match the Human Brain?" vol. 1, no. 1, p. 10.

[2] M. Wang, B. Yan, J. Hu, and P. Li, "Simulation of Large Neuronal Networks with Biophysically Accurate Models on Graphics Processors," in *the 2011 International Joint Conference on Neural Networks (IJCNN)*, pp. 3184–3193.

[3] H. Shayani, P. Bentley, and A. Tyrrell, "Hardware Implementation of A Bio-plausible Neuron Model for Evolution and Growth of Spiking Neural Networks on FPGA," in *NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 236–243.

[4] K. Maezawa, T. Akeyoshi, and T. Mizutani, "Functions and Applications of Monostable-Bistable Transition Logic Elements (Mobile) Having Multiple-Input Terminals," vol. 41, pp. 148–154.

[5] V. Beiu, J. Quintana, and M. Avedillo, "VLSI Implementations of Threshold Logic-a Comprehen-sive Survey," vol. 14, pp. 1217âĂŞ–1243.

[6] G. Rose, R. Pino, and Q. Wu, "A Low-Power Memristive Neuromorphic Circuit Utilizing a Glob-al/Local Training Mechanism," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, pp. 817–819.

[7] J. Partzsch and R. Schuffny, "Analyzing the Scaling of Connectivity in Neuromorphic Hardware and in Models of Neural Networks," vol. 22, pp. 919–935.

[8] "Google Builds Artificial Brain Which Can Recognize A Cat," June 2012.

[9] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware Realization of BSB Recall Function Using Memristor Crossbar Arrays," in *49th Annual Design Automation Conference*, 2012, pp. 498–503.

[10] M. Hu, H. Li, Q. Wu, G. Rose, and Y. Chen, "Memristor crossbar based hardware realization of bsb recall function," in *the 2012 International Joint Conference on Neural Networks (IJCNN)*, June 2012, pp. 1–7.

[11] M. Hu, H. Li, Y. Chen, Q. Wu, and G. Rose, "Bsb training scheme implementation on memristor-based circuit," in *2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, April 2013, pp. 80–87.

[12] M. Hu, H. Li, Y. Chen, Q. Wu, G. Rose, and R. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–1, 2014.

[13] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nW/MHz, 128x32 Current mode Programmable Analog Vector-matrix Multiplier with Over Two Decades of Linearity," in *Proceedings of the IEEE 2004 Custom Integrated Circuits Conference*, pp. 651–654.

[14] C. Schlottmann and P. Hasler, "A Highly Dense, Low Power, Programmable Analog Vector-Matrix Multiplier: The FPAA Implementation," vol. 1, no. 3, pp. 403–411.

[15] R. Genov, S. Chakrabartty, and G. Cauwenberghs, "Silicon Support Vector Machine with OnLine Learning," vol. 17, no. 3, pp. 385–404.

[16] C. Cheng, "Novel Ultra-low power RRAM with good endurance and retention," in *2012 Symposi-um on VLSI Technology (VLSIT*, pp. 85–86.

[17] J. Park, N. Lee, M. Hasan, S. Jung, H. Choi, J. Lee, M. Jo, W. Lee, S. Park, S. Kim, Y. H. Jang, Y. Lee, M. Sung, D. Kil, Y. Hwang, S. Chung, S. Hong, J. Roh, and H. Hwang, "Effect of oxygen migra-tion and interface engineering on resistance switching behavior of reactive metal/polycrystalline $r_{0.7}Ca_{0.3}MnO_3$ device for nonvolatile memory applications," in *2009 IEEE International Electron Devices Meeting (IEDM)*, pp. 1–4.

[18] B. Gao, L. F. Liu, B. Sum, X. Liu, R. O. Han, J. F. Kang, and B. Yu, "A unified physical model of switching behavior in oxide-based RRAM," in *2008 Symposium on VLSI Technology (VLSIT)*, pp. 100–101.

[19] T. Serre, L. wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust Object Recognition with Cortex-like Mechanisms," vol. 29, no. 3, pp. 411–426.

[20] A. A. Maashri, M. DeBole, M. Cotter, N. Chandramoorthy, Y. Xiao, V. Narayanan, and C. Chakrabarti, "Accelerating neuromorphic vision algorithms for recognition," in *Proceeding of Design Automa-tion Conference*, pp. 579–584.

[21] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A Digital Neu-rosynaptic Core Using Embedded Crossbar Memory with 45pJ per Spike in 45nm," in *Proceeding of IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4.

[22] A. Jimenez-Fernandez, A. Linares-Barranco, R. Paz-Vicente, G. Jimenez, and A. Civit, "Building Blocks for Spikes Signals Processing," in *Proceeding of International Joint Conference on Neural Networks (IJCNN)*.

[23] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: corti-cal simulations with 109 neurons, 1013 synapses," in *Proceedings of the Conference on High Perfor-mance Computing Networking, Storage and Analysis*, p. 63.

[24] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The Missing Memristor Found," *Nature*, vol. 453, pp. 80–83, 2008.

[25] Y.-C. Chen, H. Li, W. Zhang, and R. Pino, "The 3D Stacking Bipolar RRAM for High Density," vol. 11, no. 5, pp. 948–956.

[26] ——, "3D-HIM: A 3-Dimensional High-Density Interleaved Memory for Bipolar RRAM Design," in *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pp. 59–64.

[27] Y.-C. Chen, H. Li, and W. Zhang, "A RRAM-based Memory System and Applications," in *The Non-Volatile Memories Workshop (NVMW)*.

[28] L. Prechelt, "Proben1-a set of neural network benchmark problems and benchmarking rules."

[29] "UCI machine learning repository," http://archive.ics.uci.edu/ml/.

[30] "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[31] J. Canny, "A computational approach to edge detection," no. 6, pp. 679–698.

[32] T. Chen, Y. Chen, M. Duranton, Q. Guo, A. Hashmi, M. Lipasti, A. Nere, S. Qiu, M. Sebag, and O. Temam, "BenchNN: On the broad potential application scope of hardware neural network accelerators," in *IEEE International Symposium on Workload Characterization (IISWC)*, pp. 36–45.

[33] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation," in *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*, 2000, pp. 201–204, http://www-device.eecs.berkeley.edu/ptm/.

[34] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," *Nano Letters*, vol. 12, no. 1, pp. 389–395, 2012.

[35] "Macsim," http://code.google.com/p/macsim/.

[36] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "pin: Building customized program analysis tools with dynamic instrumentation."

[37] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "(mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures."

[38] N. Jiang, D. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. Shaw, J. Kim, and W. Dally, "A detailed and flexible cycle-accurate Network-on-Chip simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2013, pp. 86–96.