A Heterogeneous Computing System with Memristor-Based Neuromorphic Accelerators

Xiaoxiao Liu, Mengjie Mao, Hai Li, Yiran Chen
Elec. and Comp. Engr.
University of Pittsburgh
Pittsburgh, PA, USA
{xil116, mem231, hal66,yic52}@pitt.edu

Hao Jiang
School of Engineering
San Francisco State Univ.
San Francisco, CA, USA
jianghao@sfsu.edu

J. Joshua Yang
Hewlett Packard Labs
Palo Alto, CA, USA
jianhuay@hp.com

Qing Wu, Mark Barnell
Air Force Research Lab
Information Directorate
Rome, NY, USA
{qing.wu.2, mark.barnell.1}@us.af.mil

***Abstract*—As technology scales, on-chip heterogeneous architecture emerges as a promising solution to combat the power wall of microprocessors. In this work, we propose a heterogeneous computing system with memristor-based neuromorphic computing accelerators (NCAs). In the proposed system, NCA is designed to speed up the artificial neural network (ANN) executions in many high-performance applications by leveraging the extremely efficient mixed-signal computation capability of nanoscale memristor-based crossbar (MBC) arrays. The hierarchical MBC arrays of the NCA can be flexibly configured to different ANN topologies through the help of an analog Network-on-Chip (A-NoC). A general approach which translates the target codes within a program to the corresponding NCA instructions is also developed to facilitate the utilization of the NCA. Our simulation results show that compared to the baseline general purpose processor, the proposed system can achieve on average 18.2X performance speedup and 20.1X energy reduction over nine representative applications. The computation accuracy degradation is constrained within an acceptable range (e.g., 11%), by considering the limited data precision, realistic device variations and analog signal fluctuations.**

Keywords—neuromorphic computing, memristor, crossbar array, analog circuit, network-on-chip

I. INTRODUCTION

Thanks to technology advances, billions of transistors now can be integrated on a single chip. Homogeneous multicore architecture is proposed to fully utilize the silicon area and overcome the frequency up-scaling limit incurred by the exponentially increased wire delay and power consumption [4]. However, the constraints on supply voltage scaling, off-chip communication bandwidth, and the achievable parallelism of applications [14], greatly hinder the increase of the number of CPU cores integrated on a single chip as well as the overall system computational capacity.

In recent years, heterogeneous architecture emerges as a promising technology to conquer the above challenges in multicore system designs [26]. Various forms of off-chip accelerators, e.g., ASIC, FPGA, and GPU, have been well studied [11][13][20] for the tradeoff between computation efficiency and adaptivity: among all the implementations, ASIC demonstrates the highest efficiency [19] while FPGA provides the best functional reconfigurability; GPU offers a balanced solution that often requires special programming models and complex control flows.

In this work, we propose a novel heterogeneous computing system (HCS) with on-chip artificial neural network (ANN) accelerators designated for learning and approximated computations, offering a high-efficient mixed-signal ANN acceleration complementing to the general computation of the CPU. Different from the existing ANN accelerators implemented with digital circuitry, the ANN accelerator in the new system (called neuromorphic computing accelerator, or NCA) is built on nanoscale memristor-based crossbar (MBC) arrays [29]. In the NCA, ANN computations are conducted in analog form while the control signals are kept as digital. Hierarchical MBC arrays can be dynamically reconfigured to adapt to different ANN topologies or even different types of ANN. The data migration among different MBC arrays is realized through an analog Network-on-Chip (NoC) whose routing information is supplied externally. At architecture level, the input data and configuration of the NCA are fed by the CPU while the digitalized output of the NCA is returned to the CPU after the accelerated computation completes. A general approach which translates the target codes within a program to the corresponding NCA instructions is also introduced to facilitate the utilization of the NCA. Compared to the existing implementations of approximated computation and ANN accelerators with digital circuitry, the key differentiations of our work can be summarized as:

1. A novel mixed-signal ANN accelerator is built on the new memristor technology, offering orders of magnitude performance and power efficiency improvement compared to general purpose processors;

2. A hierarchical MBC array structure is proposed in the NCA design, delivering great reconfigurability and adaptability to the different ANN types and topologies;

3. An analog NoC is utilized to transport the data within the NCA, resulting in the improvement of data transferring efficiency by eliminating the AD/DA conversions between different computing components;

4. A generic coordination interface between the NCA and the CPU is designed to ensure the efficient communication between the conventional digital computation and the mixed-signal ANN acceleration.

A set of diverse applications ranging from machine learning, pattern recognition, to image and signal processing are adopted in the evaluation on the proposed HCS. The impacts of coordination interface parameters, e.g., the IO

978-1-4799-6233-4/14/$31.00 ©2014 IEEE

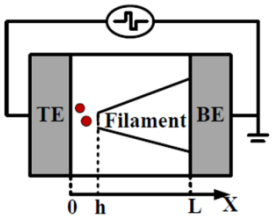
signal precision, the device variations, the signal fluctuations and the AD/DA converter placement on the computation accuracy of the NCA, are also analyzed. Experimental results show that the proposed system achieves 18.2X speedup and 20.1X energy saving on average across all 9 applications, with a reasonable accuracy degradation w.r.t. the CPU-only computation.

The reminder of this paper is organized as follows: Section II gives the preliminary on ANN, memristor and MBC structure; Section III presents an overview of the proposed HCS with NCAs; Section IV describes the design details on the system backend; Section V discusses the experiment setup and results; Section VI gives summaries and future work.

II. Preliminaries

A. Memristor as Synapse

Fig. 1. Metal-oxide memristor.



Memristor is a nonlinear passive two-terminal device of which the resistance is determined by the historical profile of the applied electrical excitations [9]. A memerisor was predicted forty years ago by Prof. Leon Chua [9] and successfully validated by HP labs in 2008 [29]. Fig. 1 shows a metal-oxide memristor model from [34], where a HfO_x layer is sandwiched between two metal electrodes – top electrode (TE) and bottom electrode (BE). During reset process, the memristor switches from low resistance state (LRS) to high resistance state (HRS). The oxygen ions migrate to the electrode/oxide interface and recombine with the oxygen vacancies. A partially ruptured conductive filament region with a high resistance per unit length R_{off} is thus formed on the left of the conductive filament region with a low resistance per unit length R_{on}, as shown in Fig. 1. During set process, the memristor switches from HRS to LRS. The ruptured conductive filament region shrinks accordingly. We define L as the total thickness of the oxide layer and h as the length of the ruptured conductive filament region, respectively. Then the resistance of the memristor R can be calculated by [34]:

$$R = R_{off}h + R_{on}(L - h). \quad (1)$$

The resistance of a memristor can be programmed to any arbitrary value by applying a current/voltage with different pulse width and/or magnitude, and retained if the applied excitation is lower than an "effective" threshold. The similarity between the electrical property of the memristor and the behavior of biological synapse inspires the implementation of memristor-based synapses [10][17][18][22]. Recent research shows that the memristor resistance can be precisely programmed [28][33] to realize an 8-bit precision on a single memristor cell [5]. Moreover, the sensing scheme that can suppress the sneak path [23] in hybrid MBC array/CMOS system has been also developed and validated by silicon [18], demonstrating great potential in neuromorphic computing system designs [31].

B. Memristor-based Crossbar (MBC)

Fig. 2. (a) The structure of the MBC; (b) Neuron logic.

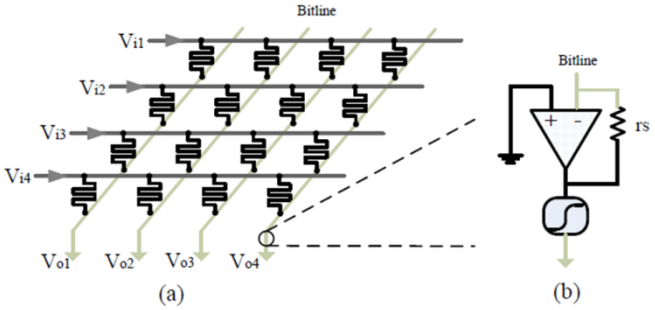
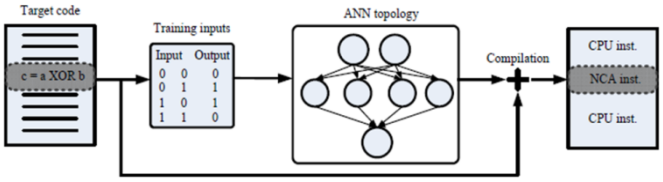


Fig. 2 shows the schematic of a MBC which represents the connections between two layers in a multilayer perception (MLP) neural network. The relationship between the input voltages (V_i) and output voltages (V_o) can be defined as: $\mathbf{V}_o = \mathbf{C}\mathbf{V}_i$, where $\mathbf{C}$ is the connection matrix. Note that the relationship between the connection matrix and the resistances matrix of MBC is not a direct one-to-one mapping due to the existence of sensing circuit at the output of the MBC. The implementation of a N-layer MLP requires N-1 MBC arrays connected in series. In this work, we adopt the MBC programming method in [15] where a write driver design with adaptive tuning circuit [5][33] is used to program memristor cells to particular resistance values. A large volume of ANN computations (weight multiplications) are conducted by the MBC in analog form without any internal control logic. Therefore, MBC structure offers high computation efficiency for ANN applications with extremely low power consumption and short computation delay, compared to any other accelerators built with digital circuitry [15].

III. System Overview

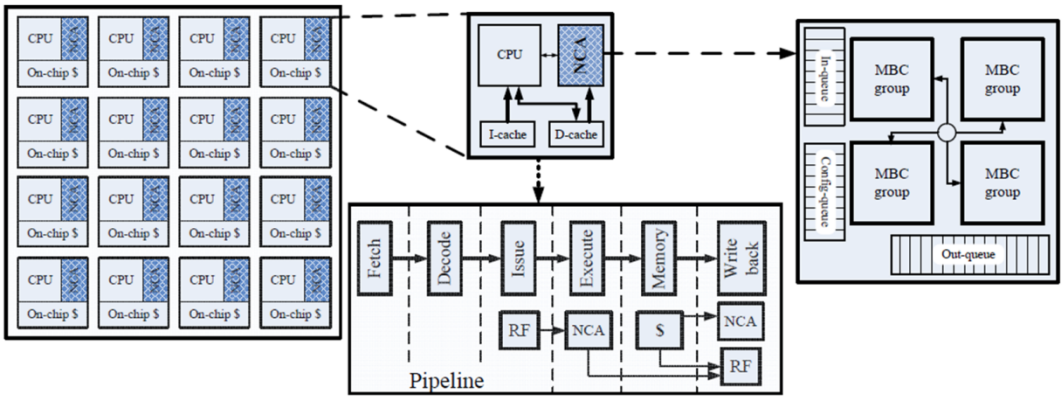
Based on the nature of the implementation, we divided the design of proposed HCS into two integrated components:

Fig. 3. System frontend.



System frontend: The system frontend is composed of all the preparation steps before running ANN computations on the NCA, as illustrated in Fig. 3. The codes that are already or can be implemented with ANN are identified first. Note that here the Boolean function XOR is used just for illustration purpose and the realistic target codes can be much more sophisticated. Based on the characteristics and complexity of the target codes, the topology of ANN, including the number of layers and the number of neurons at each layer etc., is decided and the ANN is trained offline. After that, the trained ANN is mapped to the NCA structure through the reconfiguration logic. During the NCA-aware compilation, the target codes are modified with the annotations of NCA IO instruction generation; then the compiler takes the topology of the trained ANN as the input parameters to generate NCA configuration instructions.

Fig. 4. System backend.



System backend: The system backend denotes the hardware implementation of the heterogeneous architecture, as shown in Fig. 4. 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. The NCA includes 4 MBC groups, each of which consists of 4 MBC arrays that are connected through an analog NoC. As we shall show later, such a structure ensures a high reconfigurability to support different ANN topologies. Compared to the existing ANN accelerators built with digital circuitry, our proposed mixed-signal NCA design highlights three unique features:

- Truly data flow driven execution – The execution of the NCA is evoked by the input data. The elimination of complex control logics significantly simplifies the control of the NCA and improves its utilization efficiency;
- Intrinsic high parallelism – The hierarchical MBC arrays maximize the computation parallelism of matrix-vector multiplications while the basic multiply-add operations are accelerated by the analog circuit of the MBC [15];
- Scalable computation-in-memory model – Compared to conventional ANN acceleration schemes the scale of which is limited by the storage space required by the weighted connections, NCA offers a better scalability by realizing real computation in memory.

In this paper we focus on the implementation of the system backend that is described in Section IV.

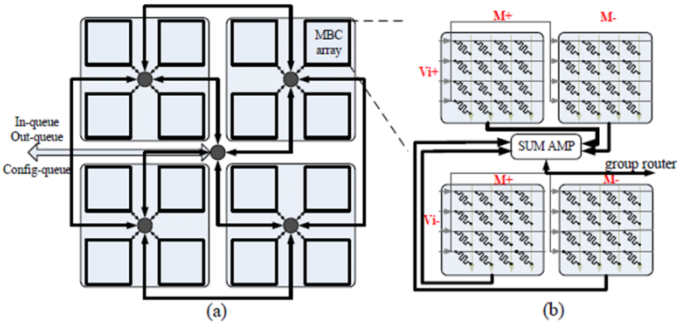
IV. System Backend – Neuromorphic Computing Accelerator (NCA)

The MBC arrays within the NCA are connected hierarchically through an analog NoC (A-NoC). In this section, we first discuss the reconfigurability of the hierarchical MBC arrays and then introduce our design of the A-NoC.

A. Reconfigurable MBC arrays

Fig. 5 depicts the MBC array structure adopted in our NCA design. Four MBC groups are arranged in a metamorphous centralized Mesh (MCMesh) manner, which are connected through a central router. Each MBC group consists of four MBC arrays that are connected through a group router. Each MBC array is designed as a 128×128 crossbar by assuming the numbers of neurons at two consecutive ANN layers are no more than 128. Such an array scale is sufficient to cover nearly 90% of learning applications most of which have no more than 100 neurons at the input layer [30]. Further increasing the MBC array size offers very marginal benefit on the computation performance improvement but incurs exponential circuit reliability degradation [23]. Within each MBC array, there are four MBCs working together to implement the multiplication between signed signals and signed synaptic weights, e.g., negative inputs are multiplied with positive weights by the MBC at the left-bottom corner. Here each connection matrix in the ANN is partitioned into positive and negative parts and mapped to the corresponding MBCs. The input signals are also divided into positive (top) and negative (bottom) lanes by the analog multiplexer.

Fig. 5. The architecture of NCA: (a) Four MBC groups are connected through a center router; four MBC arrays of a MBC group are connected by a group router; (b) One MBC array inside.



Without loss of generality, we use a connection matrix $M_{n\times m}$ as an example to present how to map the connection matrix to the MBC arrays. Here n and m denote the numbers of neurons in the input and output layers of the connection matrix, respectively. If $max(n,m) \le 128$, then the connection matrix can be directly mapped to a 128×128 MBC array; if $256 \ge n > 128$ and $m \le 128$, the connection matrix can be mapped to the two MBC arrays at one column in a MBC group; similarly, if $256 \ge m > 128$ and $n \le 128$, the connection matrix can be mapped to one row of MBC arrays within a MBC group. As we shall show in experiment results, the above mappings can cover most three/four-layer MLP topologies utilized in the simulated

applications though a larger MLP may be partitioned into the tasks within the above size ranges. If the number of partitioned tasks exceeds the availability of the MBC arrays in the NCA, the MLP may be partitioned at software level and computed serially. Multiple target codes may be executed by the same NCA simultaneously as long as the MBC groups/arrays are available. Note that the connection weights of the matrix do not change during NCA computations.

B. Analog Network-on-Chip (A-NoC)

A-NoC is introduced in the NCA to support the MLP mappings and the data migration among the MBC arrays. Since the computation of MBC arrays is in analog form, the adoption of A-NoC can eliminates the costly AD/DA conversions during data transmission. As shown in Fig. 5(a), 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 bi-direction ports connecting with 3 neighbor routers and 4 local MBC 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 MBC array. The output analog signals generated from a MBC 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. Experimental Results

A. Experimental setup

We select 9 representative applications with different computation characteristics to evaluate our proposed heterogeneous computing architecture, as shown in Table I. Four applications – building, gene, mushroom and thyroid, are selected from Proben1 [27] – a collection from UCI machine learning repository [3] tailored for neural network implementation; The Mixed National Institute of Standards and Technology (MNIST) [2] is a widely used data set in learning and recognition algorithm; canny edge detector (CDE) [6] 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 the Princeton Application Repository for Shared-Memory Computers (PARSEC) and its target codes have been implemented by ANN.

We refer to the top 5 applications in Table I 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. Table 2 shows the target codes that can be re-implemented by ANN in each imperative application. 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 CDE 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 CDE. blackscholes from benchNN [8] already provides cross validation training and testing inputs for the target codes.

The adopted ANN topology (except for the details on weighted connections) for each application is also shown in Table I. The optimal ANN topology for each application is selected by taking into account the training time, accuracy and network size. Since our simulation framework faithfully models the memristor device variation and analog signal fluctuation, we adopt the MBC training enhancement technique [24] in the Fast Artificial Neural Network (FANN) library to generate the MBC-aware ANN topology. Here the training error is defined as the average distance between the output of the trained ANN and the target output. Several error metrics are applied to evaluate the reliability of the NCA computations: mean square error (MSE) for learning applications, average relative error (ARE) and root mean square error (RMSE) for imperative applications [12], respectively. Among all the applications, the training errors are always lower than 8%.

We use Cadence to build NCA circuit components, including analog buffer, switch, sum amplifier, and sigmoid circuit, based on 45nm PTM model [7]. The power and timing parameters of different NCA components are characterized with SPICE. We also design ADC/DAC with Cadence and extract their design parameter based on SPICE simulations. The memristor parameters are adopted from [18] and carefully scaled down to 45nm. The NCA design parameters are summarized in Table II.

TABLE I. DETAILS OF THE SELECTED APPLICATIONS.

Application	Type	Target codes	% time spent at target code	Training epoch	Error metric	Training error	MLP topology w/o bias neurons	MBC arrays usage
building	ANN app	-	-	140	MSE	0.71%	14→56→23→3	3 MBC arrays in 1 group
gene	ANN app	-	-	179	MSE	0.09%	120→300→4	6 MBC arrays in 2 groups
mushroom	ANN app	-	-	16	MSE	0.02%	125→32→2	2 MBC arrays in 1 group
thyroid	ANN app	-	-	300	MSE	0.55%	21→32→3	2 MBC arrays in 1 group
MNIST	ANN app	-	-	336	MSE	0.65%	784→250→150→1	14 MBC arrays in 4 groups
blackscholes	Financial app	BlkSchlsEqEuroNoDiv	94.1%	634	ARE	7.14%	6→40→5→1	3 MBC arrays in 1 group
fft	Signal processing	trigonometric function	57.8%	119	ARE	7.03%	1→12→2	2 MBC arrays in 1 group
kmeans	Clustering	centroid calculation	42.3%	427	ARE	8.26%	6→12→1	2 MBC arrays in 1 group
canny edge detector(CED)	Image processing	sobel gradient	79.1%	764	RMSE	4.02%	9→15→5→1	3 MBC arrays in 1 group
		gaussian noise reduction		17			25→14→1	2 MBC arrays in 1 group

TABLE II. NCA PARAMETERS USED BY SIMULATION.

memristor	R_L		R_H		V_{th}	
	200Ω		160KΩ		2V	
neuron & network	V_{dd}	op amp power/delay		setup path in router power/delay		
	1.0V	100μW/0.60ns		0.72μW/0.42ns		
	sigmoid power/delay			MBC power/delay		
	10μW/0.24ns			0.69μW/3ns		
ADC mW/GHz	16-bit	8-bit	7-bit	6-bit	5-bit	4-bit
	-	290.59/1	79.31/1.25	21.76/1.25	10.18/1.5	2.77/1.5
DAC mW/GHz	16-bit	8-bit	7-bit	6-bit	5-bit	4-bit
	96.14/0.75	7.32/1	5.88/1.25	1.72/1.25	0.68/1.5	0.19/1.5

For performance and energy evaluation, we modify MacSim [1], a PIN-based [25] 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 [21] 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 [16], which is modified to model the NCA traffic of each application.

B. Impact of I/O signal precision

The limited precision of the memristor programming and input signals of the MBC array greatly impact the computation accuracy and energy consumption of the NCA. The recent studies [5][28][33] show that the programming precision of a memristor cell can be up to 8-bit. We take this assumption in our simulation by assuming 3-bit for integral part and 5-bit for decimal part. The precision of the NCA IO signals, however, is mainly determined by the capacity of ADC/DAC: As shown in Table II, ADC/DAC with a high resolution is much more power-hungry than that with a low resolution. In the following experiments, only the impact of the IO signal precision of the NCA is evaluated.

Fig. 6. The errors of (a) learning applications and (b) imperative applications under different I/O signal precision levels. The programming precision of memristor is set to 8-bit.

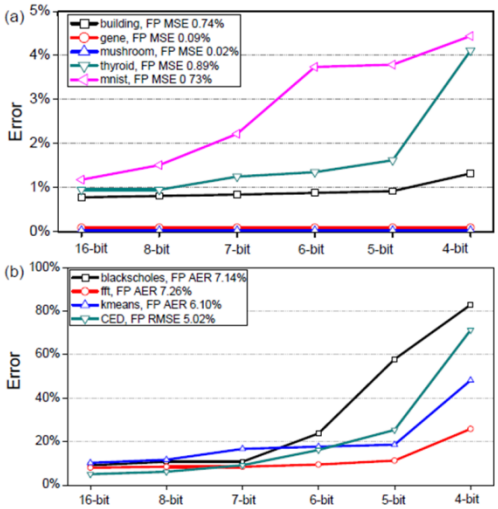


Fig. 6 illustrates the error trends of different applications when IO signal precision varies, including both learning applications and imperative applications. As shown in Fig. 6(a), the results of learning applications (except for MNIST) are quite close to the golden results achieved by floating-point (FP) computation of CPU even under 5-bit IO signal precision, indicating the excellent error tolerance of learning applications. gene and mushroom are immutable to the precision degradation since the inputs/outputs of both applications are represented by 0 and 1. The errors generated by the imperfect hardware are masked by the binary representation of the computation data. Fig. 6(b), however, indicates that imperative applications are more sensitive to the precision degradation of the input signals: if the IO signal precision level degrades to 4-bit, the average error may climb as high as 80%.

Fig. 7. The NCA energy consumption of (a) learning applications and (b) imperative applications at different I/O precision levels. The results are normalized to the one achieved with 4-bit precision level.

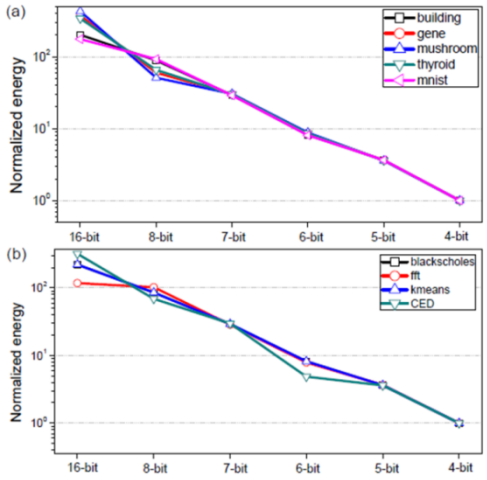


Figure 7(a) and 7(b) show the variations of NCA energy consumption of both learning and imperative applications, respectively, along with the IO signal precision degradation. Here the energy consumption of the NCA at each precision point is normalized to that of 4-bit IO precision. Following the IO signal precision degradation, the energy consumptions of all applications decrease. It implies that ADC/DAC contribute to a significant portion of NCA energy consumption and actually dominate when the IO signal precision is high.

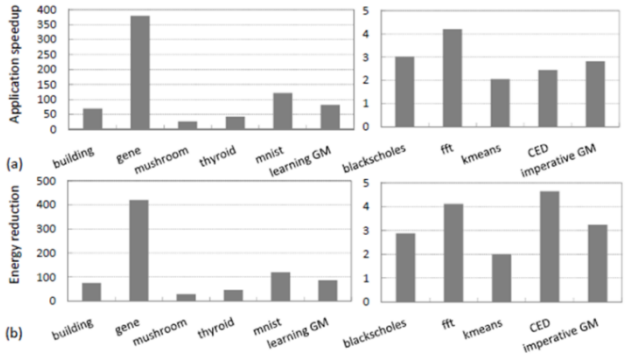
We note that when the IO signal precision reaches a certain level, further raising the precision level only gives us very marginal benefit on computation accuracy but also much high energy overhead. Hence, we are able to find the optimum value of IO precision to balance the computation accuracy and the energy consumption of the NCA. For example, from Fig. 6 and Fig. 7, we found that 8-bit IO precision is sufficient to achieve good computation accuracy close to that of 16-bit IO precision but a much lower energy consumption3. In the following experiments, we fix both IO precision and weight programming precision at 8-bit.

C. Performance and energy evaluations

Fig. 8 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 ~81.2X, as shown in Fig. 8(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 ~2.82X. 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 Figure 15 (b). On average, 86.8X energy reduction is achieved in learning applications while the one achieved in imperative applications is 3.23X, compared to the pure CPU running. Across all applications, the proposed HCS achieves 18.2X performance speedup and 20.1X energy reduction w.r.t. the conventional CPU without ANN acceleration.

Fig. 8. (a) Application speedups and (b) Energy reductions with NCA over CPU execution.



VI. SUMMARIES

In this work, 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 HCS can achieve on average 18.2X performance speedup and 20.1X 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: 1) the high-throughput of the mixed-signal NCA computation; 2) the excellent reconfigurability of the hierarchical memristor crossbar array structure in the NCA; 3) the low data transmission overhead on the analog NoC in the NCA.

ACKNOWLEDGEMENT AND DISCLAIMER

Received and approved for public release by AFRL on 06/18/2014, case number 88ABW-2014-3012. Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

REFERENCES

[1] "Macsim," http://code.google.com/p/macsim/.
[2] "The mnist database of handwritten digits," http://yann.lecun.com/exdb/mnist/.
[3] "Uci machine learning repository," http://archive.ics.uci.edu/ml/.
[4] V. Agarwal et al., "Clock rate versus ipc: the end of the road for conventional microarchitectures," in ISCA, 2000, pp. 248–259.
[5] F. Alibart et al., "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," Nanotechnology, vol. 23, no. 7, 2012.
[6] J. Canny, "A computational approach to edge detection," Pattern Anal- ysis and Machine Intelligence, IEEE Transactions on, no. 6, pp. 679– 698, 1986.
[7] Y. Cao and et.al., "New paradigm of predictive mosfet and interconnect modeling for early circuit design," in CICC, 2000, pp. 201–204, http://www-device.eecs.berkeley.edu/ ptm/.
[8] T. Chen et al., "Benchnn: On the broad potential application scope of hardware neural network accelerators," in IISWC, 2012, pp. 36–45.
[9] L. O. Chua, "Memristor-the missing circuit element," IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507–519, 1971.
[10] L. O. Chua and S.-M. Kang, "Memristive devices and systems," Proceedings of the IEEE, vol. 64, no. 2, pp. 209–223, 1976.
[11] E. S. Chung et al., "Single-chip heterogeneous computing: Does the future include custom logic, fpgas, and gpgpus?" in MICRO, 2010, pp. 225–236.
[12] H. Esmaeilzadeh et al., "Neural acceleration for general-purpose approximate programs," in MICRO, 2012, pp. 449–460.
[13] R. Hameed et al., "Understanding sources of inefficiency in general-purpose chips," in ISCA, 2010, pp. 37–47.
[14] M. D. Hill and M. R. Marty, "Amdahl's law in the multicore era," IEEE Computer, vol. 41, no. 7, pp. 33–38, 2008.
[15] M. Hu et al., "Hardware realization of bsb recall function using memristor crossbar arrays," in DAC, 2012, pp. 498–503.
[16] N. Jian et al., "A detailed and flexible cycle-accurate network-on-chip simulator," To appear in ISPASS, 2013.
[17] S. H. Jo et al., "Nanoscale memristor device as synapse in neuromorphic systems," Nano letters, vol. 10, no. 4, pp. 1297–1301, 2010.
[18] K.-H. Kim et al., "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," Nano letters, vol. 12, no. 1, pp. 389–395, 2011.
[19] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," in FPGA, 2006, pp. 21–30.
[20] V. W. Lee et al., "Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu," in ISCA, 2010, pp. 451–460.
[21] S. Li et al., "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in MICRO, 2009, pp. 469–480.
[22] K. K. Likharev, "Crossnets: Neuromorphic hybrid cmos/nanoelectronic networks," Science of Advanced Materials, vol. 3, no. 3, pp. 322–331, 2011.
[23] E. Linn et al., "Complementary resistive switches for passive nanocrossbar memories," Nature materials, vol. 9, pp. 403–406, 2010.
[24] B. Liu et al., "Digital assisted noise eliminating training for memristor crossbar based analog neuromorphic computing engine," in DAC, 2013.
[25] C.-K. Luk et al., "Pin: building customized program analysis tools with dynamic instrumentation," in PLDI, 2005, pp. 190–200.
[26] A. Lukefahr et al., "Composite cores: Pushing heterogeneity into a core," in MICRO, 2012, pp. 317–328.
[27] L. Prechelt, "Proben1-a set of neural network benchmark problems and benchmarking rules," University of Karlsruhe, Tech. Rep., 1994.
[28] S. Shin, K. Kim, and S.-M. Kang, "Memristor applications for programmable analog ics," IEEE Transactions on Nanotechnology, vol. 10, no. 2, pp. 266–274, 2011.
[29] D. B. Strukov et al., "The missing memristor found," Nature, vol. 453, pp. 80–83, 2008.
[30] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in ISCA, 2012, pp. 356–367.
[31] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," Nature nanotechnology, vol. 8, no. 1, pp. 13–24, 2012.
[32] S. Yehia et al., "Reconciling specialization and flexibility through compound circuits," in HPCA, 2009, pp. 277–288.
[33] W. Yi et al., "Feedback write scheme for memristive switching devices," Applied Physics A, vol. 102, no. 4, pp. 973–982, 2011.
[34] S. Yu, Y. Wu, and H.-S. P. Wong, "Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory," Applied Physics Letters, vol. 98, no. 10, 2011.