Memristor-based Approximated Computation

Boxun Li[1], Yi Shan[1], Miao Hu[2], Yu Wang[1], Yiran Chen[2], Huazhong Yang[1]
[1]Dept. of E.E., TNList, Tsinghua University, Beijing, China
[2]Dept. of E.C.E., University of Pittsburgh, Pittsburgh, USA
[1] Email: yu-wang@mail.tsinghua.edu.cn

***Abstract*—The cessation of Moore's Law has limited further improvements in power efficiency. In recent years, the physical realization of the memristor has demonstrated a promising solution to ultra-integrated hardware realization of neural networks, which can be leveraged for better performance and power efficiency gains. In this work, we introduce a power efficient framework for approximated computations by taking advantage of the memristor-based multilayer neural networks. A programmable memristor approximated computation unit (Memristor ACU) is introduced first to accelerate approximated computation and a memristor-based approximated computation framework with scalability is proposed on top of the Memristor ACU. We also introduce a parameter configuration algorithm of the Memristor ACU and a feedback state tuning circuit to program the Memristor ACU effectively. Our simulation results show that the maximum error of the Memristor ACU for 6 common complex functions is only 1.87% while the state tuning circuit can achieve 12-bit precision. The implementation of HMAX model atop our proposed memristor-based approximated computation framework demonstrates $22\times$ power efficiency improvements than its pure digital implementation counterpart.**

***Index Terms*—memristor, approximated computation, power efficiency, neuromorphic**

I. INTRODUCTION

Power efficiency is a major concern in computing systems. The limited battery capacity urges power efficiency of hundreds of giga floating point operation per second per watt (GFLOPS/W) for mobile embedded systems to achieve a desirable portability and performance [1]. However, the highest power efficiency of contemporary CPU or GPU systems is only $\sim$ 20 GFLOPS/W, which is expected not to substantially improve in the predictably scaled technology node [2]. Therefore, researchers are looking for an alternative computing architecture for the conventional digital computing systems to achieve performance and power efficiency gains [3].

Brain is such an power efficient platform with incredible computational capability. Compared to biological systems, nowadays programmable computing systems are more than 6 orders of magnitude less efficient [4]. This result drives more and more people to work on neuromorphic computational paradigms beyond digital logic by replicating the brain's extraordinary computational abilities. However, the contemporary neuromorphic system's research is mostly confined to algorithm field, while the hardware implementations are still built upon the traditional CPU/GPU/FPGA systems, which usually consumes a large volume of memory and computing resources [5], [6]. In general, the algorithm enhancement with conventional CMOS implementation only alleviates the power efficiency issue but not fundamentally resolve it.

In recent years, the device technology innovations have enabled new computational paradigms beyond Von-Neumann architectures, which not only provide a promising hardware solution to neuromorphic system but also help drastically close the gap of power efficiency between computing systems and the brain. The memristor is one of those promising devices. The memristor is able to support a large number of signal connections within a small footprint by taking the advantage of the ultra-integration density [7]. And most importantly, the nonvolatile feature that the state of the memristor could be tuned by the current passing through itself makes the memristor a potential, perhaps even the best, device to realize neuromorphic computing systems with picojoule level energy consumption [8], [9].

Our objective is to use memristors to design a power efficient neuromorphic framework for approximated computation with both programmability and computation generality. The framework is inspired by the theory that a multilayer neural network can work as a universal approximator, and the widely observation of approximated computation in a series of applications, ranging from signal processing, pattern recognition to computer vision and nature language processing. Both the neuromorphic computing architecture and the tolerance of inexact computation can be leveraged for substantial performance and power efficiency gains [9], [10].

To realize this goal, the following challenges must be overcome: Firstly, a power efficient Memristor ACU which can complete general algebraic calculus is demanded to accomplish different computing task. Secondly, Memristor ACUs must be organized effectively to form a computation framework in order to accomplish different processing tasks. Thirdly, there needs a parameter configuration algorithm to work out the effective parameters of the Memristor ACUs according to the function requirement. Finally, the memristor cannot be effectively programmed and thus a state tuning scheme both efficient and accuracy for memristors is demanded.

In this work, we for the first time propose a complete framework for memristor-based approximated computation. The contributions of this paper are:

1) We propose a power efficient memristor-based approximated computation framework. The framework is integrated with our programmable Memristor ACUs. Simulation results show that our Memristor ACU offers less than 1.87% error for 6 common complex functions and works reliably under a noisy condition;
2) We provide a parameter configuration algorithm to convert the weights of the neural network to the appropriate resistance of the memristors in Memristor ACUs. The algorithm can provide both the maximum dynamic range and the validity of the memristance by taking the impact of fabrication variation into consideration;

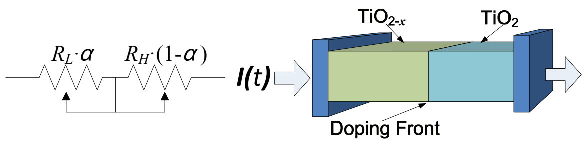

Fig. 1. Physical model of HP memristor

3) We propose a feedback circuit to program the memristor accurately. The circuit can provide a linear relationship between the input voltage and the final value of the memristor. The accuracy increases with the frequency of the reference voltage and an analog storage with 12-bit precision is realized under 2MHz;

The rest of our paper is organized as follows: Section 2 provides background information and related work. Section 3 proposes the details of memristor-based approximated computation framework. Section 4 depicts the detailed program method of the framework including the parameter configuration algorithm and the high-precision state tuning circuit for memristors. Section 5 presents a case study based on HMAX model. Section 6 concludes this work and shows some future directions.

II. Background Knowledge and Related Work

A. Memristor

Memristor was first physically realized by HP Labs in 2008 [11]. Fig. 1 shows the physical model of the HP memristor [12], which is a two-layer thin film of TiO_2. One layer consists of intact TiO_2 while the other layer lacks of a small amount of oxygen (TiO_{2-x}). These two layers demonstrate different conductivity and the overall resistance is the sum of the two layers. When a current is applied to the device, the boundary of layers (doping front) will move so as to change the resistance of the memristor.

B. Universal Approximators by Multilayer Feedforward Networks

It has been proven that a universal approximator can be implemented by a multilayer feedforward network with only one hidden layer and sigmoid activation function [13], [14]. Table I gives the maximum errors of the approximations of six common functions by this approximated method based on MATLAB simulation. The mean square errors (MSE) of approximations are less than 10^{-7} after the back propagation training algorithm completes. Theoretically, the networks accuracy will increase with the size of the network. However, it's usually more difficult to train a network of a bigger size [15]. A larger network may easily fall into local minima, instead of the global one, and thus sometimes provides a worse result as shown in Table I. Such a precision level (MSE $\approx 10^{-7}$) is sufficient for most approximated computation, e.g., the HMAX model included in our case study.

TABLE I
Maximum Errors of Network Approximators

Function	Nodes in the Hidden Layer					
	0	5	10	15	20	25
$x_1 \cdot x_2 \cdot x_3$	22.79	1.10	0.68	0.28	0.34	0.27
x^{-1}	9.53	0.25	0.20	0.14	0.10	0.05
$sin(x)$	10.9	0.05	0.07	0.05	0.07	0.06
$log(x)$	7.89	0.21	0.13	0.14	0.12	0.14
$exp(-x^2)$	20.27	0.04	0.03	0.05	0.03	0.04
$\sqrt{x}$	13.76	1.87	1.19	1.43	0.35	0.49

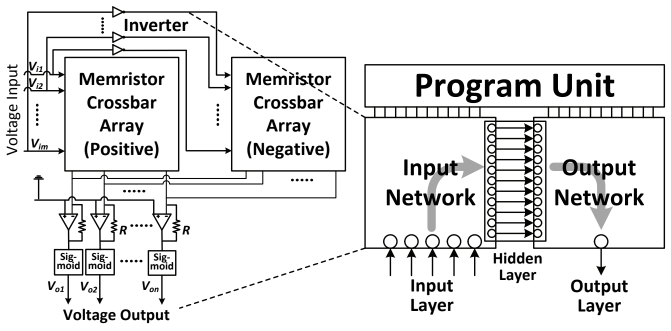

Fig. 2. Memristor ACU

III. Memristor-Based Approximated Computation Framework

A. Memristor ACU

Fig. 2 shows the overview of the Memristor ACU. The Memristor ACU is based on a memristor-based hardware implementation of a multilayer network (with one hidden layer) to work as a universal approximator. The mechanism is as follows. The basic operation of feedforward network is matrix-vector multiplication and can be mapped to the memristor crossbar array illustrated in Fig. 3. The output of the crossbar array can be expressed as:

$$V_{oj} = R \cdot \sum_k (V_{ik} \cdot g_{kj}), g_{kj} = \frac{1}{M_{kj}} \tag{1}$$

where V_{ik} and V_{oj} are the input and output voltage of each row and column. k and j represent the index numbers of input and output voltage, respectively. And they are also the index numbers of the rows and columns of the crossbar array separately. M_{kj} and g_{kj} represent the resistance and admittance of each memristor in the crossbar array, respectively; R is the resistor at the end of the crossbar array.

Op Amps are used to enhance the output accuracy and isolate the two layers. All Op Amps work as current feedback amplifiers so that the function of such a crossbar array can be expressed as a vector-matrix multiplication.

Since both R and M can only be positive, two crossbars are needed to represent the positive and negative weights of a network, respectively [9]. Each pair of the corresponding inputs to the two crossbar arrays is complimentary, e.g., one is the reversion of the other one to guarantee the opposite polarities and the same amplitude of the input voltages. The

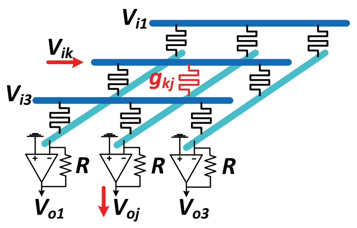

Fig. 3. Memristor Crossbar Array

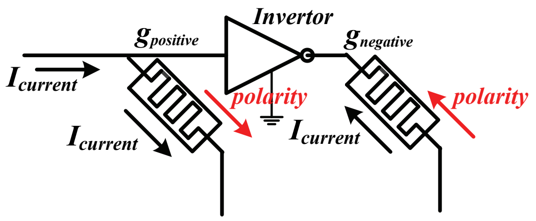

Fig. 4. Memristor Pairing

practical weights of the network can be expressed as:

$$w_{kj} = R \cdot (g_{kj(postive)} - g_{kj(negative)}), g_{kj} = \frac{1}{M_{kj}} \qquad (2)$$

which is the difference between the conductances of the two paired memristors in the positive and negative crossbar arrays. In addition, the sigmoid activation function can be generated by the circuit similar to that in [16] and a complete double-layer feedforward network unit with one weight matirx is accomplished.

We note that the polarities of the terminals of the memristors in two crossbars are set to the opposite direction in order to minimize the deviations of the memristor states caused by the accumulation of the current passing through the memristors. As the input pair is complimentary, the polarities of the programming currents passing through the paired memristors in two crossbar arrays are opposite, leading to the same resistance changing rate and direction of the two memristors. As a result, the deviations of the two memristors will cancel each other. We refer to this technique as memristor pairing and its shown in Fig. 4.

Finally, by combining two double-layer networks together, a multi-layer feedforward network unit (with one hidden layer) is realized. As shown in Section 2.2, the network can work as a universal approximator and thus a memristor approximated computation unit is accomplished.

The maximum errors of network approximators shown in Table I are obtained under ideal conditions without any noise. However, as pointed out in [9], the fluctuations on the input signal and the parametric variations of the memristor devices will have influence on the accuracy of the Memristor ACU. In order to evaluate the robustness of our proposed Memristor ACU under different noise conditions, we conduct Monte-Carlo simulations for the functions listed in Table I. The

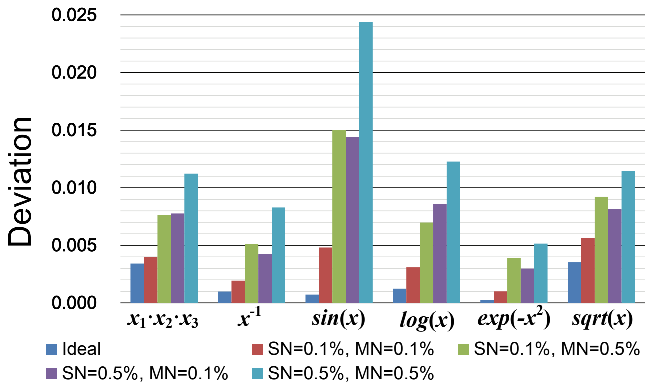

Fig. 5. Maximum errors of the approximators under different noise conditions. SN: input signal noise ratio. MN: The standard deviation of the memristor resistance.

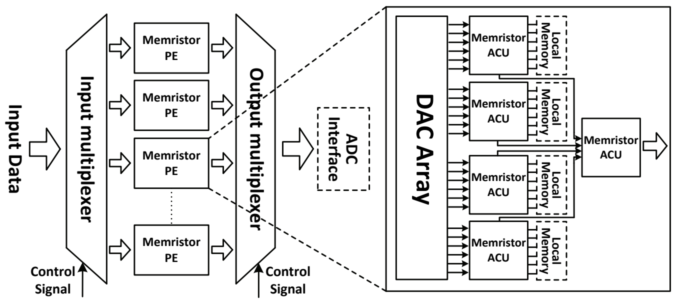

Fig. 6. Memristor-based Approximated Computation Architecture

number of the nodes in the hidden layer is set to 20. Total 5 different noise conditions are tested with different input signal noise ratios and memristor resistance deviations. The MSE of each approximator is trained to less than 10^{-7} before the test starts. Each Monte-Carlo simulation includes 15,000 runs. The simulation results in Fig. 5 shows that our Memristor ACU is immune to certain magnitude of noise and the immunity varies with the category of the function: For example, $\sin(x)$ and $\sqrt{x}$ are more sensitive to the noise while $\exp(-x^2)$ and x^{-1} can tolerate the noise with larger magnitude. Therefore, during the application of the Memristor ACUs, the noise condition must be well controlled below the corresponding level for specific functions.

B. Memristor-based Approximated Computation Framework

The overview of the proposed memristor approximated computation framework is shown in Fig. 6. The building blocks of the framework are the memristor processing elements (Memristor PE). Each Memristor PE consists of several Memristor ACU to accomplish algebraic calculus. Each Memristor PE is also equipped with its own digital-to-analog converters to generate analog signal for processing and hold it for a period of time until the Memristor ACU finishes computation. In addition, the Memristor PE may also have several local memory, e.g., analog data stored in form of the resistance of memristors, according to the functional requirement. Atop of that, all the Memristor PEs are organized by two multiplexers in Round-Robin algorithm. To be specific, in the processing stage, the data will be injected into the platform sequentially and the input multiplexers will deliver the data into the relevant Memristor PE for approximated computation. The data get to the Memristor PE in digital format and the DAC in each Memristor PE help convert the data into analog signals and hold them. Each Memristor PE may work under low frequency but all the Memristor PEs work in parallel to achieve a high performance. Finally, the output data will be transmitted out from the Memristor PE by output multiplexer for further processing, e.g., the data can be converted back into digital format by a high performance ADC.

The framework is scalable and the user can configure it according to individual demand. For example, for a power efficiency required task, it's better to choose low power Op Amps to form the Memristor ACUs and each Memristor PE may work under low frequency. On the other hand, high speed Op Amps and ADCs and hierarchical task allocation architecture will be a better solution to the problems requiring high-performance platforms.

IV. Program Memristor Approximated Computation Unit

A. Parameter Configuration

As shown in Section 3, the memristor approximated computation unit is based on the memristor-based hardware implementation of a multilayer network (with one hidden layer) to work as a universal approximator. Hence, the parameters of the network must be determined for specific function. We build a library for several common functions to reduce the complexity of configuring the parameters of the network. We also provide several mature training algorithms, such as stochastic gradient descent and L-BFGS, for neural networks to calculate the parameters of unprepared functions [15].

Once the parameters of the network are determined, the parameters need to be mapped effectively to the appropriate resistance of the memristors in the crossbar arrays. Improperly converting the array weights to the resistance of the memristors may result in the following problems: 1) the converted parameter is beyond the actual range of the device; 2) the dynamic range of the parameters is too small so that the memristor state may easily saturate; and 3) the weights of the network are so high that the summation of the output voltage exceeds the working range of the Op Amps. In this section, we propose a parameter configuration algorithm to convert the weights of the neural network to the appropriate resistance of the memristor to prevent improper parameter conversion.

The feasible range of the weight of the network can be expressed by the function of the memristor parameters as:

$$-R\cdot(g_{on}-g_{off})\le w\le R\cdot(g_{on}-g_{off}) \tag{3}$$

where:

$$g_{on}=\frac{1}{M_{\min}},\quad g_{off}=\frac{1}{M_{\max}} \tag{4}$$

where M_{min} and M_{max} represent the global maximum and minimum resistance of the memristor crossbar arrays, respectively.

In order to extend the dynamic range and minimize the impact of fabrication variation, we adjust g_{on} and g_{off} to:

$$g'_{on}=\frac{1}{\eta\cdot\Delta_{\min}+M_{\min}},\quad g'_{off}=\frac{1}{M_{\max}-\eta\cdot\Delta_{\max}} \tag{5}$$

where Δ_{min} and Δ_{max} represent the maximum deviation for M_{min} and M_{max} induced by the fabrication variation of the crossbar array, respectively. η is a scale coefficient which is set to 1.1$\sim$1.5 in our design to achieve a safety margin. The appropriate resistance of the memristors, g_{pos} and g_{neg}, can be achieved by solving the following cost function:

$$(g_{pos},g_{neg})=\arg\min\{|g_{pos}-g'_{mid}|+|g_{neg}-g'_{mid}|\} \tag{6}$$

where:

$$g'_{mid}=\frac{g'_{on}+g'_{off}}{2} \tag{7}$$

The constraint condition of Eq. 6 is:

$$\begin{cases} R\cdot(g_{pos}-g_{neg})=w\\ g'_{on}\le g_{pos}\\ g_{neg}\le g'_{off}\end{cases} \tag{8}$$

where w is the weight of the network. The minimum of the risk is achieved when:

$$\begin{cases} g_{pos}=g'_{mid}+\frac{w}{2R}\\ g_{neg}=g'_{mid}-\frac{w}{2R}\end{cases},w\le R\cdot(g'_{on}-g'_{off}) \tag{9}$$

These are the appropriate resistance of the memristors with the minimum risk of saturate and the impact of fabrication variation to prevent improper parameter conversion.

B. State Tuning Circuit with Feedback

Once the parameters of the network are determined, the memristors in the crossbar array need to be programmed to the specific states. This requires a state tuning circuits. Unfortunately, there still lacks an approach that fully satisfy the mobile system requirements: the main categories of memristor state tuning circuits are either using multiple voltage pulses to step-by-step tune the memristor, which limits the tuning accuracy (e.g., $\sim$ 5-bit precision in [17]) or using several resistors as reference, which incurs a high hardware cost such as a large number of reference resistors (for example, an 8-bit precision state tuning circuit needs about 256 resistors as reference [18]). Therefore, we introduce a feedback circuit with minor hardware cost to program the memristor accurately and guarantee the accuracy of the network parameters.

Fig. 7 illustrates our proposed memristor state tuning circuit with feedback. OP1 is a current feedback amplifier. Assuming the initial negative input of OP1 is grounded, the output voltage of OP1 $V_{OP1}(t)$ is:

$$\mathrm{V}_{OP1}(t)=-I_{R_1}(t)\cdot M(t)=-\frac{M(t)}{R_1}\times\mathrm{V}_{\sin}(t) \tag{10}$$

where $M(t)$ represents the resistance of the memristor. $V_{sin}(t)$ is the sinusoidal reference signal with small amplitude connected to the negative input of OP1. The frequency of $V_{sin}(t)$ should be sufficiently high so that the memristor state is not influenced significantly over periods. Diode D_1 and capacitor C form a peak detector and record the peak output voltage of OP1:

$$\mathrm{V}_C(t)=\frac{M(t)}{R_1}\times\mathrm{V}_{\sin(\max)}-\mathrm{V}_{D_{1(th)}} \tag{11}$$

where $V_C(t)$ is the output voltage of the peak detector and $V_{D_1}(t)$ is the voltage drop across D_1. In addition, C can get discharged through R4 so that $V_C(t)$ can track the peak of $V_{OP1}(t)$ even if the peak voltage climbs down. OP2 is also a current feedback amplifier. When $R_2=R_4$ and $R_3=R_5$,

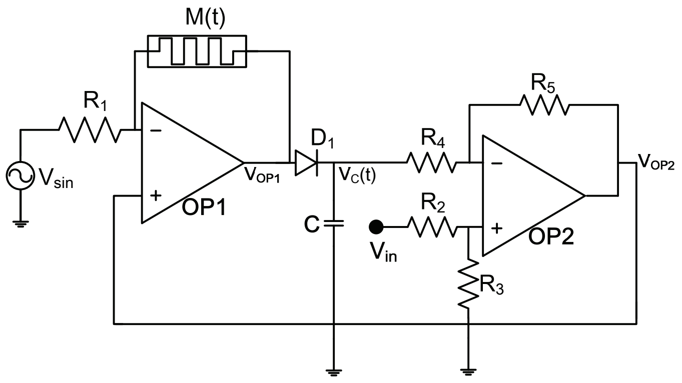

Fig. 7. Memristor State Tuning Circuit

TABLE II
PARAMETERS OF THE MEMRISTOR

R_L (Ω)	R_H (Ω)	μ_v ($m^2 \cdot s^{-1} \cdot V^{-1}$)	L (nm)	R_1 (Ω)	$V_{read(max)}$ (V)
100	16000	10^{-14}	1	1000	1

TABLE III
MAXIMUM ERRORS OF MEMRISTOR STATE TUNING (Ω)

$M(t)$	Frequency of V_{sin} (100kHz)					
	1.0	2.0	5.0	10.0	20.0	50.0
$1k\Omega$	32.267	15.984	6.206	3.142	1.586	0.687
$2k\Omega$	24.040	14.413	5.949	2.782	1.363	0.664
$5k\Omega$	22.162	13.037	5.789	2.314	1.078	0.428
$10k\Omega$	20.843	11.758	4.442	2.771	1.224	0.489

OP2 functions as a subtractor and the output voltage of OP2 ($V_{OP2}(t)$) can be expressed as:

$$\mathrm{V}_{OP2}(t) = \frac{R_3}{R_4} \times (\mathrm{V}_{in} - \mathrm{V}_C(t)) \tag{12}$$

where V_{in} is the input voltage that determines the target state of the memristor. The output voltage of OP2 will be sent back into the positive input of OP1 as the DC bias of $V_{OP1}(t)$. This feedback bias will tune the memristor state until $V_C(t) = V_{in}$. According to Eq. 11 and 12, the final state of the memristor can be expressed as:

$$M_{final} = \frac{R_1}{\mathrm{V}_{sin(\max)}} \times (\mathrm{V}_{in} + \mathrm{V}_{D_{1(th)}}) \tag{13}$$

Eq. 13 demonstrates a linear relationship between the input voltage and the final resistance (memristance) of the memristor. The negative feedback between $M(t)$ and V_{OP2} helps to maintain the memristor state with high accuracy.

Finally, we conduct SPICE simulation to evaluate the proposed state tuning circuit. The memristor parameters adopted in the simulation are summarized in Table II. Table III shows the maximum errors of memristor resistance from the target states under different reference voltage frequency. The tuning accuracy is around $\pm 0.01\%$ (f=2MHz), indicating a 12-bit multilevel digital storage.

In addition, compared with other state tuning circuits described earlier in this section. Our design provides about 128 times higher accuracy than the first method [17] and saves around 1000 times number of resistors under the same accuracy condition and tuning speed compared with the second method [18]. In conclusion, our circuit provides both high precision and small circuit size at the same time without increasing the tuning time cost.

V. A CASE STUDY: HMAX

A. HMAX Model

In order to evaluate the performance of our proposed mixed-signal accelerator, we conduct a case study on HMAX model. HMAX is a trainable model for general object recognition in complex environment by Serre, Wolf, and Poggio [19]. The model consumes more than 95% amount of computation to perform pattern matching by calculating the distance between the prototypes and units as:

$$D(X,P) = \exp\{-\frac{\sum_{o=1}^{12}\sum_{w=1}^{n}\sum_{h=1}^{n}(x_{h,w,o} - p_{h,w,o})^2}{\alpha \cdot n^2 \cdot 12}\} \tag{14}$$

where $x_{h,w,o}$ represents the pixel in each possible position X and $p_{h,w,o}$ represents the pixel of each pattern. $n \in 4, 8, 16$ is the size of the pattern. α is scale parameter to adjust feature matching results. During the normal operation, there are hundreds of such patterns. The amount of computation is too huge to realize real-time video processing on conventional CPUs while the computation accuracy requirement is not strict (around 10 $\sim$ 12 bit [20]). In our case study, we apply the proposed memristor-based approximated computation framework to conduct the distance calculations to promote both processing performance and power efficiency. The overview of the proposed framework are the same as Fig. 6.

As shown in Fig. 6, each memristor processing element consists of four 6-input Memristor ACU for Gaussian calculators and one for 4-input multiplication. Therefore, each Memristor PE can calculate a 24-input distance in Eq. 14 per clock cycle. Additionally, each Memristor PE is equipped with 4 local analog memory based on memristors to store the pattern. The main concern in this design is power efficiency and thus we choose low power amplifiers to construct the Memristor PEs. The working mechanism of the HMAX framework is the same as Section 3.2 and the detailed parameters of the PE are illustrated in Table IV.

TABLE IV
PARAMETERS OF MEMRISTOR ACU IN HMAX

Average memristance (Ω)	10000
Average input voltage (V)	0.5
Amount of input in each analog ALU	24
Amount of output in each analog ALU	1
Amount of D/A converters in each analog ALU	24
Amount of A/D converter in mixed-signal framework	1
Bit-length of data	10
Power of each memristor (μW)	1.25
Power of each sigmoid circuit (μW)	10
Power of each D/A converter (μW)	0.4
Power of each A/D converter (μW)	15
Amount of memristors in each analog PE	1270
Amount of Op Amps in each analog PE	223
Amount of sigmoid circuits in each analog PE	95
Amount of D/A converter in each analog PE	240
Delay for A/D converters and input data (ns)	5
Delay for each analog PE (ns)	50

B. Performance of the Framework

We use 1,000 images (350 of cars and 650 of the other categories) from PASCAL Challenge 2011 database [21] to evaluate the performance of the HMAX system on the digital and the memristor-based approximated computation framework. Each image is of 320×240 pixels with complex background. The HMAX model contains 300 patterns of car images which remain the same on each platform. A correct result indicates both a right judgment on the classification of the object and a successful detection on the object location. The results of correct rate are shown in Table. V. As we can observe, the correct rate degradation is only 2.1% on the ideal

TABLE V
PERFORMANCE OF THE MEMRISTOR-BASED HMAX

Platform	Memristor-based Framework					CPU
Signal Noise Rate (%)	0	0.1	0.1	0.5	0.5	-
Device Noise Rate (%)	0	0.1	0.5	0.1	0.5	
Accuracy (%)	78.1	75.8	69.2	65.7	60.1	80.2

TABLE VI
POWER EFFICIENCY OF THE MEMRISTOR-BASED HMAX

DAC (mW)	ADC (mW)	Analog (mW)	Total (mW)	Flops /cycle	Frequency (MHz)	Efficiency (GFlops/W)
96	15	248.38	359.38	740	200	411.83

TABLE VII
POWER EFFICIENCY COMPARISON WITH DIFFERENT PLATFORMS (FPGA, GPU, CPUS IN [20])

Parameters	Proposed	FPGA	GPU	CPUs
Size of input image	320×240	256×256		
HMAX orientations	12			
HMAX scale	12			
HMAX prototypes	300	5120		
Average size of prototypes	8			
Cycles need for Eq.14	32	-		
Amount of Eq.14 /frame	5455×300	-		
Frequency (MHz)	200	-		
Power (W)	0.359	40	144	116
Unified fps/W	10.627	0.483	0.091	0.023
Speed Up	-	22.00	116.78	462.04

memristor-based approximated computation framework w.r.t. the CPU platform. Such a small degradation may be easily compensated by increasing the amount of patterns [19]. When taking the noise into consideration, the correct rates further drops to up to 10 percent less, which are still good enough for normal applications. To satisfy a high accuracy application, the noise level needs to be suppressed.

C. Power Efficiency of the Memristor-based Framework

The power efficiency evaluation of the memristor-based HMAX accelerator is given in Table VI while the detailed comparisons to other platforms are given in Table VII. Our simulation results show that the power efficiency of memristor-based approximated computation framework is higher than 400 GFLOPS/W. Compared to other platforms like FPGA, GPU and CPU in [20], memristor-based HMAX achieved a performance up to 10.6 fps/W, which is $> 20\times$ higher than its digital counterparts.

VI. CONCLUSION

In this work, we propose a approximated computation framework based on memristor technology for power-efficient approximated computation. We first introduced a memristor approximated computation framework by integrating our programmable Memristor ACU. Our experiment results of HMAX application show that the memristor-based framework is able to achieve $> 20\times$ power efficiency with slightly degraded correct rate. We also introduce the parameter configuration algorithm of the Memristor ACU along with a high precision feedback state tuning circuit.

However, there're still many challenges remaining in this memristor-based approximated computation framework. For example, the interfaces between digital and analog systems, such as ADCs, are always the key consideration of mixed-signal computing systems. Thus, how to design an automatic design flow to configure the interfaces efficiently is one of the challenges. In addition, we note that the accuracy of Memristor ACUs will slowly degrade when the operation time increases. The reason is that the parameters of the memristors drift gradually from the initial states. This observation leads to an open problem of the optimized refresh scheme for memristor-based computation. Finally, the performance of the framework will be limited by the input and output multiplexers, especially when the number of Memristor PEs becomes higher. Therefore, how to design an efficient interconnect architecture for the Memristor PEs will be another interesting problem.

ACKNOWLEDGMENT

This work was supported by 973 project 2013CB329000, National Science and Technology Major Project (2011ZX03003-003-01, 2013ZX03003013-003) and National Natural Science Foundation of China (No.61261160501, 61028006), and Tsinghua University Initiative Scientific Research Program.

REFERENCES

[1] DARPA. Power efficiency revolution for embedded computing technologies.

[2] NVIDIA TESLA K-SERIES DATASHEET. Kepler family product overview, 2012.

[3] Esmaeilzadeh et al. Dark silicon and the end of multicore scaling. In *ISCA*, pages 365–376. IEEE, 2011.

[4] Duygu Kuzum, Rakesh GD Jeyasingh, et al. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano letters*, 12(5):2179–2186, 2011.

[5] Jeffrey Dean et al. Large scale distributed deep networks. In *Neural Information Processing Systems*, 2012.

[6] Ngiam et al. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[7] Sung Hyun Jo, Ting Chang, et al. Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters*, 10(4):1297–1301, 2010.

[8] Jianxing Wang et al. A practical low-power memristor-based analog neural branch predictor. In *Proceedings of ISLPED 2013*.

[9] Miao Hu, Hai Li, et al. Hardware realization of bsb recall function using memristor crossbar arrays. In *DAC*, pages 498–503, 2012.

[10] Esmaeilzadeh et al. Neural acceleration for general-purpose approximate programs. In *MICRO*, pages 449–460, 2012.

[11] Dmitri B Strukov, Gregory S Snider, et al. The missing memristor found. *Nature*, 453(7191):80–83, 2008.

[12] Pino et al. Statistical memristor modeling and case study in neuromorphic computing. In *DAC*, pages 585–590. ACM, 2012.

[13] K. Hornik, M. Stinchcombe, et al. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[14] Yoshifusa Ito. Approximation capability of layered neural networks with sigmoid units on two layers. *Neural Computation*, 6(6):1233–1243, 1994.

[15] Laurene V Fausett. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall Englewood Cliffs, 1994.

[16] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi. Analog implementation of a novel resistive-type sigmoidal neuron. *TVLSI*, 20(4):750 –754, april 2012.

[17] Sangho Shin, Kyungmin Kim, and Sung-Mo Kang. Memristor applications for programmable analog ics. *Nanotechnology, IEEE Transactions on*, 10(2):266 –274, march 2011.

[18] Wei Yi, Frederick Perner, et al. Feedback write scheme for memristive switching devices. *Applied Physics A*, 102:973–982, 2011.

[19] Jim Mutch and David G. Lowe. Object class recognition and localization using sparse features with limited receptive fields. *Int. J. Comput. Vision*, 80(1):45–57, October 2008.

[20] Ahmed Al Maashri, Michael Debole, et al. Accelerating neuromorphic vision algorithms for recognition. In *DAC*, DAC '12, pages 579–584, New York, NY, USA, 2012. ACM.

[21] Mark Everingham et al. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.