A versatile neuromorphic system based on simple neuron model

Cite as: AIP Advances **9**, 015324 (2019); https://doi.org/10.1063/1.5052609
Submitted: 20 August 2018 • Accepted: 16 January 2019 • Published Online: 24 January 2019

C. M. Zhang, G. C. Qiao, S. G. Hu, et al.

A versatile neuromorphic system based on simple neuron model

Cite as: AIP Advances 9, 015324 (2019); doi: 10.1063/1.5052609
Submitted: 20 August 2018 • Accepted: 16 January 2019 •
Published Online: 24 January 2019

C. M. Zhang, G. C. Qiao, S. G. Hu,[a] J. J. Wang, Z. W. Liu, Y. A. Liu, Q. Yu, and Y. Liu

AFFILIATIONS

School of Electronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, P. R. China

[a]Corresponding author: sghu@uestc.edu.cn

ABSTRACT

Brain-inspired neuromorphic computing has attracted much attention for its advanced computing concept. However, the massive hardware cost in fully-connected architectures makes it challenging to build a large-scale neuromorphic system. In this work, we report a compact, programmable, versatile, and scalable neuromorphic architecture. To demonstrate the concept of the neuromorphic architecture, a neuromorphic system consisting of four cores is implemented on an FPGA platform. On the one hand, the neuromorphic system is extremely compact and hardware-saving. The computing block based on a simple digital leaky Integrate-and-Fire (LIF) model only costs 69 logic elements (LEs); only one physical neuron is implemented in each core, and it can be reused as hundreds of virtual neurons by time-division-multiplexing; only four 9-bit synaptic weights are assigned to each neuron, which effectively alleviates the hardware explosion in fully-connected architecture. On the other hand, the neuromorphic system is programmable and versatile, and can perform different neural network computing. The neuromorphic system mapped with a three-layer feedforward network successfully recognizes the MNIST handwritten digits with an accuracy of 96.26%, and it also effectively realizes different convolution operations which are basic computing operations in convolutional neural networks. Last but not least, each neuromorphic core has its own router module, making it convenient to scale up.

I. INTRODUCTION

The human brain, consisting of 10^{10}-10^{11} neurons, is one of the most complex systems known at present.[1] Although Von-Neumann computers have developed rapidly according to Moore's law in the past half-century, they work fundamentally different from the human brain, making them low efficiency, high power consumption, and poor performance in implementing the brain behaviors.[2] Recently, neuromorphic computing system inspired by the brain has received much attention.[3] Since the emerging of neuromorphic computing concept, analog CMOS technology has been widely used to develop neuromorphic cores, such as the Neurogrid,[4] the BrainScaleS,[5] and the ROLLS.[6] Analog neurons are compact, but are poor in programmability and are challenging to multiplex.[7] In comparison, the digital CMOS technology shows high multiplex capability, programmability and scalability, and thus neuromorphic platforms based on digital CMOS/FPGA technologies have received much attention.[8] In 2014, IBM reported the TrueNorth neuromorphic core based on 28-nm digital CMOS technology, which employs a stochastic leaky Integrate-and-Fire neuron as the basic building block.[9,10] D. Ma et al. reported the Darwin neuromorphic chip based on 180-nm digital CMOS technology,[11] which consists of a RISC CPU and a neural-network core based on the LIF neuron model.

Compared with CMOS technology, FPGA is a competitive candidate for implementing digital neuromorphic platforms because of its short design cycle, low cost, high flexibility, and excellent stability. In 2011, A. Cassidy proposed an FPGA-based array of LIF neurons,[12] and an FPGA-based large-scale neuromorphic architecture.[13] In 2015, D. Wang reported an FPGA

based multicore neuromorphic system with a scalable routing network.[14] In 2017, E. I. Guerra-Hernandez reported an FPGA-based neuromorphic locomotion system for three different legged robots.[15] Y. Qi et al. reported an FPGA-based multicore neuromorphic system for ECG and edge detection application.[16] However, for a single neuron may cost massive hardware resource, and the hardware cost of synapses increases drastically with the number of neurons in a fully-connected architecture, it is full of challenge to implement a neuromorphic system whose scale is comparable to that of the human brain.

In this work, a compact, programmable, versatile, and scalable digital neuromorphic platform is proposed and implemented on an FPGA platform. Firstly, the neuromorphic core is hugely compact: 1) the basic building block is constructed based on a simple digital LIF neuron model, which only costs 69 logic elements (LEs); 2) only one programmable neuron is physically implemented in a neuromorphic core, while it is able to be reused as 256 LIF neurons, which theoretically reduces the hardware cost of neurons by 255 times; 3) each neuron shares four different synaptic weights, which effectively avoids the quasi-square growth of synapse hardware cost with neuron number in fully-connected architectures. Secondly, the neuromorphic system is programmable and versatile. Critical properties of the building block, such as threshold, leaky, and reset modes, are designed to be configurable, allowing the neuromorphic system to be able to map various types of neural network. As examples, fully-connected multilayer spiking neural network and different convolution operations are successfully implemented on the neuromorphic platform, indicating the feasibility of the way to build the neuromorphic system. Thirdly, a programmable router module is packaged in the compact neuromorphic core, making the neuromorphic system easy to be scaled up. This work may provide a useful reference for building extremely large-scale neuromorphic systems.

II. SYSTEM DESIGN

The neuron, as the fundamental computing element, fundamentally determines the functionality and scalability of the neuromorphic system. Among the various neuron models (i.e., the MP model,[17] the Hodgkin-Huxley model,[18] and the Izhikevich model[19]), the leaky integrate-and-fire (LIF) model[20] is widely used in computational neuroscience and neuromorphic computing, because of its highly biological plausibility and excellent computing efficiency. In order to save hardware cost, and to improve computational efficiency and large-scale scalability, a simple digital LIF neuron model is used in this neuromorphic system, whose membrane potential is governed by:

$$V_j(t) = V_j(t-1) + \sum_i^n (O_{i,j} \times w_{i,j}) + L_j \tag{1}$$

$$V_j = \begin{cases} V_j(t) & |V_j| < H_j \\ R_j & |V_j| \geq H_j \end{cases} \tag{2}$$

where j and i are the indexes of post- and pre-synaptic neurons, respectively; $w_{i,j}$ is the synaptic weight; $V_j(t)$ and $V_j(t-1)$ are the membrane voltage in the t^{th} and $(t-1)^{th}$ timestep, respectively; $O_{i,j}$ denotes whether the i^{th} pre-synaptic neuron fires a spike in the $(t-1)^{th}$ timestep; L_j denotes the leakage voltage; H_j is the firing threshold; and R_j is the reset voltage. If the membrane potential is greater than or equal to H_j, the neuron fires a spike and resets its membrane potential to the R_j.

In the present work, a 4-core neuromorphic system based on the simple LIF neuron model was implemented on a Terasic DE2-115 FPGA board. Figure 1(a) shows the overall architecture of the neuromorphic system, which consists of a System Controller, a serial peripheral interface (SPI) Controller, and a two-dimensional Core Array. The SPI Controller module acts as the data exchange channel between the system and the host FPGA. The System Controller is used to set up the required

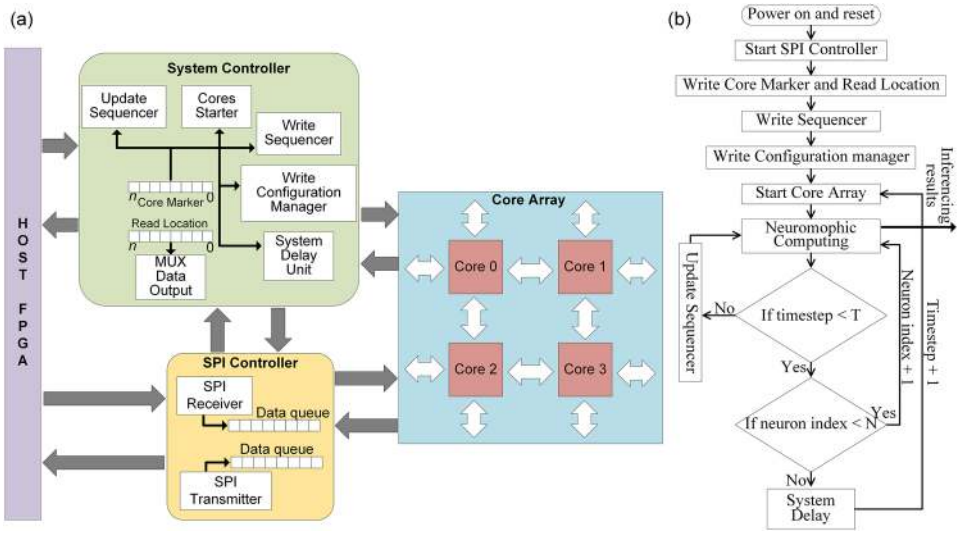

FIG. 1. (a) Schematic of the neuromorphic platform frame; (b) computing process of the neuromorphic system.

ALGORITHM 1. Method for implementing neuromorphic computing.

Input: Network structure and the number of layers (S, L); network input data $\{X^t\}_t^T$; the number of neurons in layer l $\{N^l\}_{l=1}^L$; parameters of layers ($\{w_{i,j}^{t,l}, V_j^{t,l}, O_i^{t,l-1}, L_j^l, H_j^l, R_j^l\}_{t,l=2,i=1,j=1}^{T,L,N^{l-1}N^l,}$); pre-synaptic neurons type $\{Y_i^l\}_{l=2,i=1}^{L,N^{l-1}}$; computing window T;
Output: Inferencing results
Forward (inference):
1: Construct the network on the platform according to S, L
2: **for** t =1 to T **do**
3: $O^{t,1} \leftarrow X^t$ // Input layer receives input data
4: **for** l =2 to L **do** // Hidden and output layers process data
5: **for** j = 1 to N^l **do** // Each neuron process sequentially
6: **for** i =1 to N^{l-1} **do** // Integrate each synapse
7: **if** $O_i^{t,l-1}$ = 1 **then** // Activated when the pre-synaptic neuron fire spikes
8: Update $w_{i,j}^{t,l}$ according to Y_i^{l-1}
9: Update $V_j^{t,l}$, $O_j^{t,l}$ according to Eq. (1)–(2)
10: **end if**
11: **end for**
12: **end for**
13: **end for**
14: **end for**

operating environment for the Core Array. The System Controller determines which cores are involved in the computing process. During the system initialization process, configuration data is sent to the System Controller from the host FPGA under control of the SPI controller, and then rearranged and written to the Configuration Manager of the target core(s).

There are two registers (Register 1 and Register 2) in the System Controller. The Register 1 (Core Marker) is used to mark the required core(s), while the Register 2 (Read Location) is used to mark the core(s) output to the host FPGA. Configuration information is written to the Registers 1 and 2 before core initialization via the Write Configuration Manager. The System Controller sequentially checks each bit of the Register 1. Once a high-level signal is detected, the configuration information is configured into the corresponding core. After completing a round of neuron switching, the whole system needs to wait for 25 clock cycles for the 255th neuron's spikes to reach the target synapses. This action is realized by the System Delay Unit. The cores of the Core Array can communicate with each other via the routing network by event packets, and the leaving routing interfaces are reserved around the Core Array for future system scale up. The detailed neuromorphic computing process is given in Figure 1(b), and the pseudo code for implementing memory-efficient neuromorphic computing is given in Algorithm 1.

Figure 2(a) shows the schematic of the neuromorphic core. It consists of a Neuron module, a Synapse Array Manager, a Time-Multiplexing Controller, a Sequencer, a Configuration Manager and a Router. Figure 2(b) shows the logic schematic of the Neuron module. Logically, the Neuron module only consists of a comparator, an adder, 2 registers, and 5 selectors; physically, the hardware-efficient neuron module only costs 69 LEs. The Neuron module (together with the Synapse Array module) can realize synaptic integration, leak

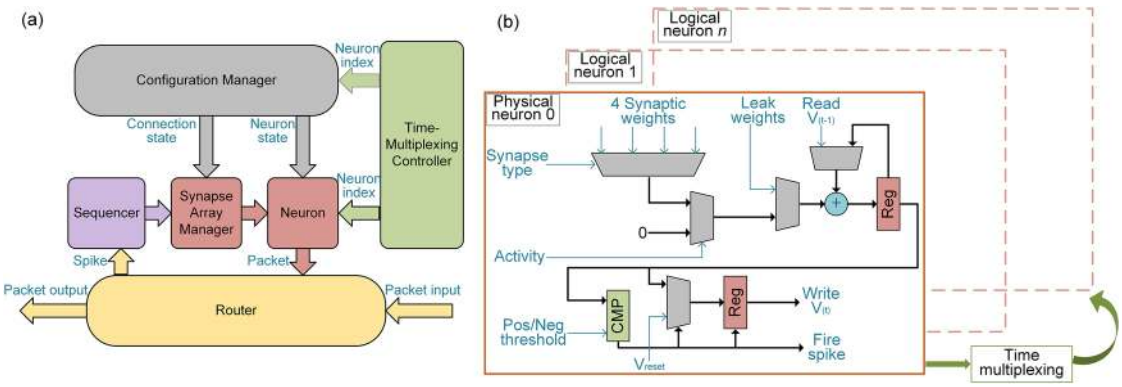

FIG. 2. (a) Schematic of the neuromorphic core; (b) logic schematic of the Neuron module.

subtraction, threshold checking, spike firing and reset process described in equations (1) and (2). The Synapse Array Manager and the Neuron module can be reconfigured by the Configuration Manager under control of the Time-Multiplexing Controller, and the Neuron module can be reused as 256 neurons (each neuron has 256 synapses). The Sequencer, consisting of 16 registers in 256-bit width and hardware logic units for data writing/reading/counting, can temporarily store input data from the host FPGA or other core. The Configuration Manager manages the configuration information, typically including the membrane potential, the type of synapse, the synaptic weight, the threshold voltage, the reset voltage, and the leakage voltage. A neuron can be configured to four 9-bit signed integer weights, and the specific weights are assigned to the corresponding synapses by the Neuron module and the Synapse Array Manager, which largely reduces the weight storage space. The spikes generated by a particular neuron can be sent to any target synapses in the Synapse Array by the Router module. Once a spike is generated in the Neuron module, its router encapsulates the information about the target core, the target neuron, and the target synapse into a packet and then forwards to the target core.

III. RESULTS AND DISCUSSIONS

The proposed neuromorphic platform was programmed in Verilog and compiled using Quartus II, and implemented on Terasic DE2-115 FPGA board. After compilation, placement, and routing, the DE2-115 board operates at 100 MHz. Different neural networks can be mapped to the versatile reconfigurable neuromorphic platform. Here, a fully-connected multi-layer neural network and convolution operations are demonstrated as examples.

Firstly, handwritten digit recognition is implemented with a multi-layer feedforward neural network on the neuromorphic platform. The images of 28×28 pixels from the MNIST handwritten digital database are resized in 16×16-pixel images. A three-layer artificial neural network consisting of a 256-neuron input layer, two 256-neuron hidden layers, and a 10-neuron output layer was trained with the 16×16 pixel images

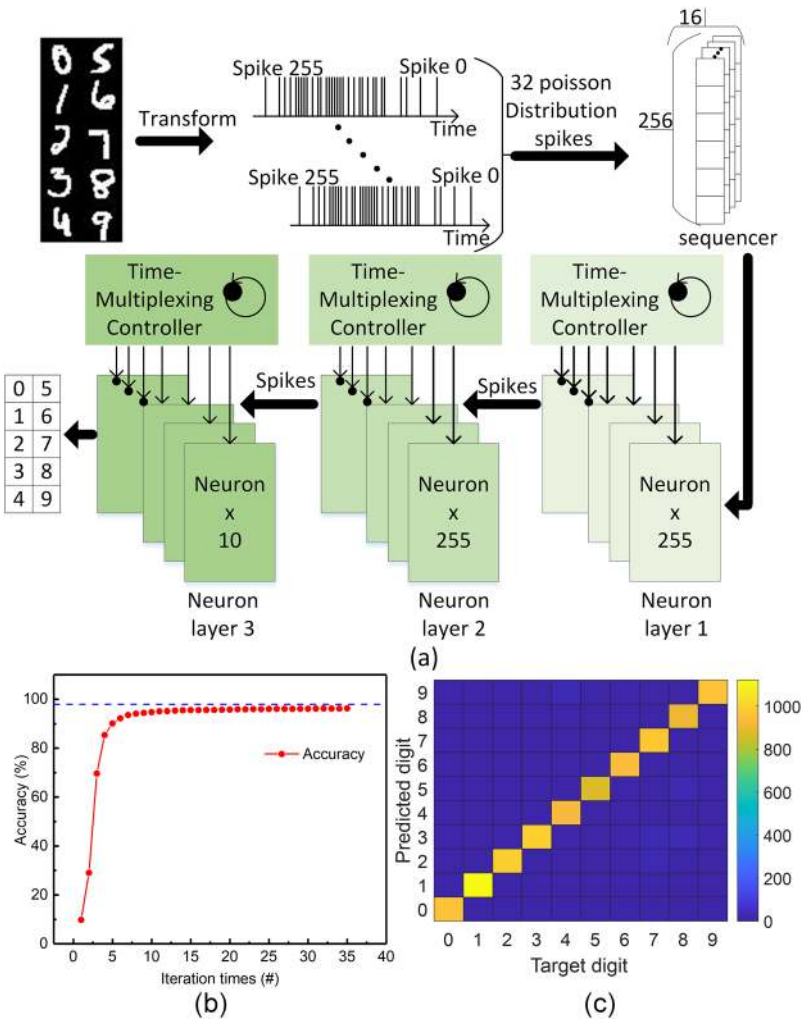

FIG. 3. (a) Schematic of the 3-layer spiking neural network implemented on the neuromorphic platform; (b) dependence of recognition accuracy on iteration times neuromorphic platform; (c) average recognition matrix on the test set digits of MNIST database.

by the back-propagation algorithm on the Matlab, and the Rectified Linear Unit (ReLU) activation function and zero bias were applied to all neurons during the training process.[21] The trained artificial neural network was then converted into a spiking neural network of the same size. Pixel grayscale information encoded by Poisson-distributed 32-bit spike trains (a 16×16-pixel image corresponds to 256 spike trains) were fed to the spiking neural network as input signal.[22]

In an iteration, one bit in each spike trains was processed, and thus the 32-bit spike trains need to be iterated 32 times. The spiking neural network was mapped into the FPGA-based neuromorphic platform, as schematically illustrated in Figure 3(a). Spike trains were temporarily stored in the Sequencer. After the spike trains stored in the Sequencer were fed to the Neuron module, the System Controller switches this core into waiting state and updates the data of the Sequencer via the Update Sequencer module. After completing the updating process, the Cores Start module resends a start signal to this core to continue data processing. In each iteration, if any neuron in the output layer fires a spike, the corresponding spike sum will be added by 1, and output neurons with the most significant spike sum will be regarded as the predicted result.

The trained artificial neural network can recognize handwritten digits from the MNIST test set with an accuracy of 97.94%. Figure 3(b) shows the dependence of the recognition accuracy on the number of iterations. The recognition accuracy increases gradually from ~10% to 96.26% after iterating 32 times, indicating little degrade compared with that of the artificial neural network. As shown in Figure 3(c), the FPGA-based neuromorphic platform can successfully recognize the digits, and the recognition accuracy is 96.26%. The small degrade of the recognition accuracy (by ~1.5%) was induced by the simple weight storage scheme (4 signed integer weights for each neuron). However, it is acceptable and cost-effective considering the storage compactness, the computing simplicity, and potential scalability it brings.

The versatile neuromorphic platform is suitable for implementing convolution operations. Convolution operations on images including edge detection and high-pass filtering are performed on the neuromorphic platform, as schematically shown in Figure 4(a). Variables n_n and k_s respectively represent the index of neuron and the kernel size (convolutional window size). Figure 4(b) shows the original 256×340-pixel image used for demonstrating convolution operation. Pixel grayscales are encoded by Poisson-distributed 32-bit spike trains. Each output neuron of the convolutional layer is mapped to a single neuron of the Neuron module, and all cores simultaneously process the corresponding convolutional windows. Figure 4(c) and (d) show the images processed with a 3×3 edge detection kernel ($\begin{bmatrix}-1 & 0 & -1\\ 0 & 4 & 0\\ -1 & 0 & -1\end{bmatrix}$) and a high-pass filter ($\begin{bmatrix}-1 & -1 & -1\\ -1 & 9 & -1\\ -1 & -1 & -1\end{bmatrix}$), respectively. The neuromorphic platform can successfully realize different convolution operations, and thus it holds the potential to map the popular convolutional neural networks, which is our ongoing work.

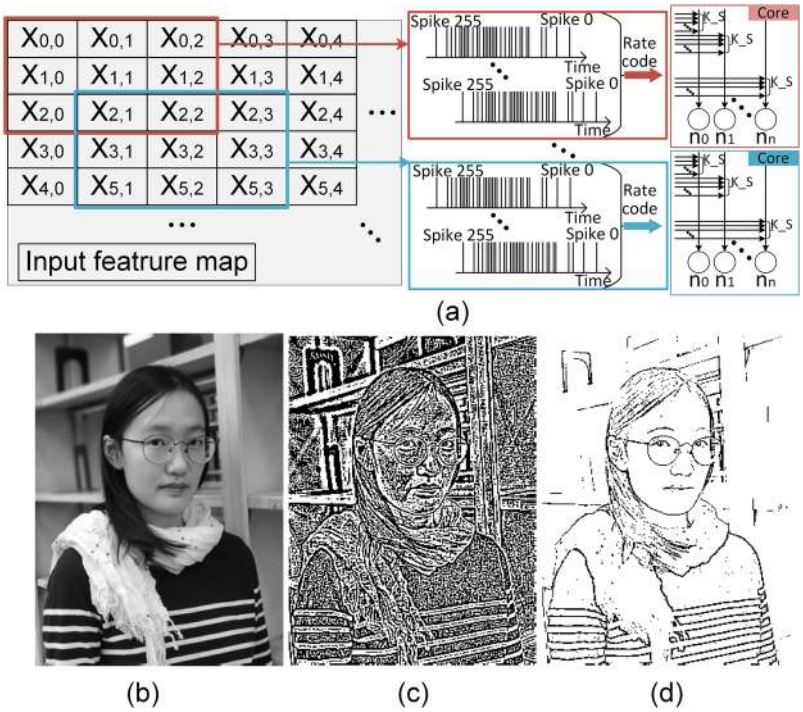

FIG. 4. (a) Schematic of the convolution operation implemented on the neuromorphic platform; (b) original image for processing. Photo of author Z. W. Liu, taken by author C. M. Zhang; (c) image processed with an edge detection kernel; (d) image processed with a high-pass filter.

TABLE I. Comparison of previous work and this work.

Platform	Weights	Clock	LE/Neuron	Network	Time	Power
B. Glackin et al. 2005[23]	N^2	100Mhz	63	100×100×100	115.7s	N/A
A. Cassidy et al. 2007[13]	N^2	50Mhz	431	N/A	N/A	N/A
Y. J. Qi et al. 2014[16]	N^2	50Mhz	115	279×50×16	140000s	N/A
This work	$N \times 4$	100Mhz	69	256×256×10	370ms	293mW

Thanks to the compact, simple LIF neuron model, memory-saving weight storage scheme, and hardware-saving multiplexing scheme, the neuromorphic platform has excellent memory efficiency. Table I shows the comparison of the present work and pioneer research works. A physical neuron in this work only costs 69 LEs, which is close to that of the neuron that only has positive threshold property in Ref. 23; while the present platform completes a neural network computing task with only 370 ms and 293 mW, which is time- and energy-efficient. At the same time, the present platform reduces the number of synaptic weights from N^2 to $N \times 4$ (N denotes the total number of neurons in the system). Besides, fully-connected spiking neural network implemented on the neuromorphic platform is also 64× more memory bandwidth efficient than that on conventional CPUs (Intel i5-4200U CPU). The memory bandwidth required to complete single neuromorphic computing can be calculated by $F \times \sum_i^n (N_i \times P_i)$ (F is the clock frequency, N_i is the number of synapses neuron i owns, and P_i represents the bit width of all parameters neuron i owns including synaptic weight, leaky weight, and etc.). All cores in the FPGA-based neuromorphic platform have independent storage space and neuromorphic computing unit, and is able to make full use of FPGA parallel computing. The neuromorphic platform has an average 10.7× speedup over the Matlab implementation on CPU. The Power Analyzer, a powerful analysis and optimization tool provided by Altera, is used to estimate power consumption of the FPGA-based neuromorphic platform. The FPGA-based neuromorphic platform is 51.23× more energy efficient on average than the Intel i5-4200U CPU.

IV. CONCLUSIONS

We present a scalable and reconfigurable neuromorphic platform implemented on an FPGA board whose neuron is based on the leaky Integrate-and-Fire model. Each core of the neuromorphic platform shares the same physical circuit to realize the neuromorphic computation of 256 neurons. This neuromorphic platform can achieve high-accuracy (96.26%) handwritten digit recognition with only four types of synaptic weight for each neuron. The neuromorphic platform can also perform convolution operations, making it promising for mapping convolution neural networks. Moreover, to implement fully-connected neural networks of the same size, the neuromorphic platform is 64× more memory bandwidth efficient, 51.23× more energy efficient, and 10.7× speedup than Intel i5-4200U CPU. This work provides a practical and feasible method for the construction of large-scale neuromorphic systems.

ACKNOWLEDGMENTS

This work was supported in part by NSFC under Project 61774028 and Project 61771097, in part by the Fundamental Research Funds for the Central Universities under Project ZYGX2016Z007, and in part by the Opening Project of Science and Technology on Reliability Physics and Application Technology of the Electronic Component Laboratory under Project ZHD201602.

REFERENCES

[1] Q. Yu, H. Tang, J. Hu, and K. C. Tan, *Neuromorphic Cognitive Systems*, 1st ed. (Springer International Publishing AG, Cham, 2017), p. 1.
[2] J. Von Neumann, *The Computer and the Brain* (Yale University Press, New Haven, 2000).
[3] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, ArXiv Prepr. ArXiv1705.06963 (2017).
[4] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, Proc. IEEE **102**, 699 (2014).
[5] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, in *ISCAS 2010–2010 IEEE Int. Symp. Circuits Syst. Nano-Bio Circuit Fabr. Syst.* (2010), pp. 1947–1950.
[6] G. Indiveri, F. Corradi, and N. Qiao, in *Tech. Dig.–Int. Electron Devices Meet. IEDM* (2015), p. 4.2.1–4.2.4.
[7] A. Sripad, G. Sanchez, M. Zapata, V. Pirrone, T. Dorta, S. Cambria, A. Marti, K. Krishnamourthy, and J. Madrenas, Neural Networks **97**, 28 (2018).
[8] A. Calimera, E. Macii, and M. Poncino, Funct. Neurol. **28**, 191 (2013).
[9] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, A. Amir, D. B. D. Rubin, F. Akopyan, E. McQuinn, W. P. Risk, and D. S. Modha, in *Proc. Int. Jt. Conf. Neural Networks* (2013).
[10] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, Science (80-.). **345**, 668 (2014).
[11] D. Ma, J. Shen, Z. Gu, M. Zhang, X. Zhu, X. Xu, Q. Xu, Y. Shen, and G. Pan, J. Syst. Archit. **77**, 43 (2017).
[12] A. Cassidy, S. Denham, P. Kanold, and A. Andreou, Conf. Proc. - IEEE Biomed. Circuits Syst. Conf. Healthc. Technol. BiOCAS2007 75 (2007).
[13] A. Cassidy, A. G. Andreou, and J. Georgiou, in *2011 45th Annu. Conf. Inf. Sci. Syst. CISS 2011* (2011).
[14] D. Wang, L. Deng, P. Tang, C. Ma, and J. Pei, in *2015 15th Non-Volatile Mem. Technol. Symp. NVMTS 2015* (2016).
[15] E. I. Guerra-Hernandez, A. Espinal, P. Batres-Mendoza, C. H. Garcia-Capulin, R. J. De Romero-Troncoso, and H. Rostro-Gonzalez, IEEE Access **5**, 8301 (2017).
[16] Y. Qi, B. Zhang, T. M. Taha, H. Chen, and R. Hasan, in *Natl. Aerosp. Electron. Conf. Proc. IEEE* (2015), pp. 255–258.
[17] W. S. McCulloch and W. Pitts, Bull. Math. Biophys. **5**, 115 (1943).
[18] A. L. Hodgkin and A. F. Huxley, Bull. Math. Biol. **52**, 25 (1990).
[19] E. M. Izhikevich, IEEE Trans. Neural Networks **14**, 1569 (2003).

[20] L. F. Abbott, Brain Res. Bull. **50**, 303 (1999).
[21] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, in *Proc. Int. Jt. Conf. Neural Networks* (2015).
[22] M. V. Tsodyks and H. Markram, Proc. Natl. Acad. Sci. **94**, 719 (1997).
[23] B. Glackin, T. M. McGinnity, L. P. Maguire, Q. X. Wu, and A. Belatreche, Comput. Intell. Bioinspired Syst. **3512**, 552 (2005).