An Efficient Event-driven Neuromorphic Architecture for Deep Spiking Neural Networks

Duy-Anh Nguyen*†, Duy-Hieu Bui*, Francesca Iacopi†, Xuan-Tu Tran*‡
*SISLAB, VNU University of Engineering and Technology – 144 Xuan Thuy road, Cau Giay, Hanoi, Vietnam.
†School of Electrical and Data Engineering, University of Technology Sydney – Broadway 2007, New South Wales, Australia.
‡Corresponding author's email: tutx@vnu.edu.vn

***Abstract*—Deep Neural Networks (DNNs) have been successfully applied to various real-world machine learning applications. However, performing large DNN inference tasks in real-time remains a challenge due to its substantial computational costs. Recently, Spiking Neural Networks (SNNs) have emerged as an alternative way of processing DNN's task. Due to its event-based, data-driven computation, SNN reduces both inference latency and complexity. With efficient conversion methods from traditional DNN, SNN exhibits similar accuracy, while leveraging many state-of-the-art network models and training methods. In this work, an efficient neuromorphic hardware architecture for image recognition task is presented. To preserve accuracy, the analog-to-spiking conversion algorithm is adopted. The system aims to minimize hardware area cost and power consumption, enabling neuromorphic hardware processing in edge devices. Simulation results have shown that, with the MNIST digit recognition task, the system has reached an accuracy of 94.4% with a core area of 15 μm^2 at a maximum frequency of 250 MHz.**

***Index Terms*—Hardware Accelerator, Convolutional Neural Network, Event-driven Neural Network, Neuromorphic Computing**

I. Introduction

OVER the past few years, modern deep neural networks (DNNs) architectures such as AlexNet [1], VGG-16 [2], ResNet [3] have contributed to the success of many machine learning applications. Ranging from the small, simple task of handwritten digits recognition [4] to challenging datasets with millions of images with 1000s classes [5], DNNs have proven to be the de facto standard with better-than-human accuracy. However, inference on such large networks, e.g., classification on a single image from ImageNet, requires significant computational and energy costs, limiting the uses of such networks on powerful GPUs and datacenter accelerators such as Google TPUs [6].

The VLSI research community has made considerable research efforts to push the DNNs computing task on mobile and embedded platforms. Notable research trends include developing specialized dataflow for Convolutional Neural Network (CNN) to minimize power consumption of DRAM access [7], reducing the network size for mobile applications [8], [9], model compression (pruning redundant parameters while preserving accuracy) [10], quantization of parameters [11] and applying new computing paradigm, such as computing in log-domain [12], in frequency-domain [13] or stochastic computing [14]. These techniques rely on the traditional frame-based operation of DNN, where each frame is processed sequentially, layer by layer until the final output recognition can be made. This may result in long latency and may not be suitable for applications where fast, real-time classification is crucial.

Spiking Neural Network (SNN) has been widely adopted in the neuroscience research community, where it serves as a model to simulate and study the behaviors of human brain [15]. Recently, it has emerged as an efficient way of doing inference tasks on complex DNN architectures. The event-based mode of operations is particularly attractive for complex DNN workloads for several reasons. Firstly, the output classification result can be queried as soon as the first output spike arrives [16], reducing the latency and computational workload. Secondly, simple hardware-efficient Integrate-and-Fire (IF) neuron models may be used in SNN, replacing the expensive multiplication operation with addition. Thirdly, SNN has been reportedly proven to be equal in terms of recognition accuracy with state-of-the-art DNN models [17], [18]. With clever and efficient algorithms to convert the parameters of traditional DNN models to spiking domain, SNN opens up the possibility of leveraging the plethora amount of pre-trained DNN models and training techniques in the literature, without the need to develop specific networks models for SNN.

Even though DNN-to-SNN conversion algorithms have proven to be useful, specific hardware accelerator targeting this method is still lacking in the literature. In this work, we propose an efficient event-driven neuromorphic architecture to support the inference of image recognition tasks. The main contributions of this paper include a novel digital IF neuron model to support SNN operations, and a system-level hardware architecture which supports handwritten digit recognition with the MNIST dataset [19]. Simulation results have shown that the hardware system only incurs negligible loss (0.2%) with 10-bit precision format compared to the software floating point results. Hardware implementation results with a standard 45nm library also show that the system is resource efficient with a gate equivalent (GE) count of 19.2k (2-input NAND), at a maximum frequency of 250 *MHz* and throughput of 325,000 frames-per-second.

The remaining part of the paper is organized as follows. Section II presents some preliminaries regarding SNN and the conversion algorithms adopted in this work. Section III

introduces the hardware architecture in details. Section IV covers the simulation and implementation results. Finally, Section V concludes the paper.

II. SNN Preliminaries

In this section, the basic theory and methods for DNN-to-SNN conversion are presented. This work adopt the methods introduced in [16], [20]. More detailed information can be found in these works.

A. Introduction to SNN

The human brain, despite possessing a great computational power, only consumes an average power of 20 Watts [21]. This is thanks to a very large interconnection networks of primitive computing elements called the *neurons* and *synapses*. Figure 1 shows a schematic diagram of a biological neuron. Each neuron consists of many *dendrites*, which act as input

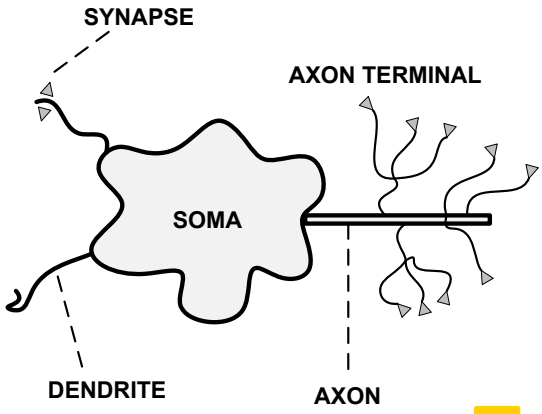


Fig. 1: Schematic diagram of a biological neuron.

device. The dendrites receive inputs from connected neurons in the previous layer of the networks. The connection between neurons from the previous layer and the dendrites are called *synapses*. Each neurons may have an arbitrary number of such connections. The membrane voltage of the soma integrates those inputs, and transmits the outputs to the next layer through the *axon* and its many *axon terminals*, which act as the output devices.

Inspired from the working mechanism of such biological neuron, many research efforts have been made to create biological plausible neuromorphic computing paradigm to solve many difficult tasks for traditional Von Neumann computers, while maintaining a very low energy consumption profile. SNN recently attracted many research interests as a feasible candidate for future neuromorphic computing. SNN is the third generation of Artificial Neural Networks, and it is particularly suitable for low-power hardware implementation due to its event-driven operations, while still maintaining equivalent computing power to its DNN counterpart [22]. The general behaviour of a neuron in SNN is depicted in Fig. 2 (recreated from [23]). The neurons in SNN operates with binary input and output spikes. When a neuron receives pre-synaptic spikes from previous layers, the membrane potential will integrate these spikes with the corresponding weights. Each neuron population will have its own threshold potential. If the membrane potential crosses this threshold value, the neuron will emit an output spike to the neurons in the next layer. After emitting a spike, the neuron will enter a refractory state within a specific refractory period, in which incoming spikes are not integrated.

B. Rate-coded input and output representations in SNN

A fundamental shift of SNN from traditional DNN operations is how the inputs to the networks are represented. In frame-based DNN operations, inputs to the first layer of the network are analogue values (for example, the pixel intensity values of an image). SNN operates on binary input spike trains generated as function of each simulation time step. There are different ways to represent information with binary input spikes, which can be broadly classified as *rate-coding* or *temporal coding*. In rate-coding scheme, the information encoded as the mean firing rate of emitting spikes over a timing window [16] [20]. In temporal coding, the information is encoded in the timing of emitting spikes [24]

In this work, we adopt the rate-coding scheme in [16]. The inputs to the first layer in SNNs are rate-coded binary input spikes. The inputs spike trains are generated in each time step based on a Poisson process as indicated in (1):

$$I(x_i, t) = \begin{cases} 1 & \text{if} \quad x_i \leq X \\ 0 & \text{if} \quad x_i > X \end{cases} \tag{1}$$

where x_i are the analog inputs from neuron i and $X \sim U[(0, 1)]$ is a random variable uniformly distributed on $[0, 1]$. With a large enough time window, the number of binary input spikes generated is directly proportional to the analog input value.

Given a set of input spike trains, the spikes are accumulated and transmit through each layer of the networks. Each layer of the network can start its operation as soon as there is a spike input coming from the previous layer. At the final output layer, the inference is made based on the cumulative spike counts, e.g., the output neuron with the highest output spike counts is deemed to be the classification result.

C. Synaptic operations in SNN

In traditional DNN, the analog inputs values are accumulated and go through an activation function. Various activation functions are introduced in the literature, such as the sigmoid, tanh, or Rectified Linear Unit (ReLU). The ReLU activation function, firstly introduced in [1], is currently the most used activation function in modern DNN architectures. A DNN neuron with ReLU activation, receiving x_i inputs from the previous layer, each with synaptic weights w_i and zero bias, produce the following output:

$$y = max(0, \sum_i w_i x_i) \tag{2}$$

The conversion algorithm from DNN to SNN is firstly introduced in [20] by Cao *et al.*. The author proposed the

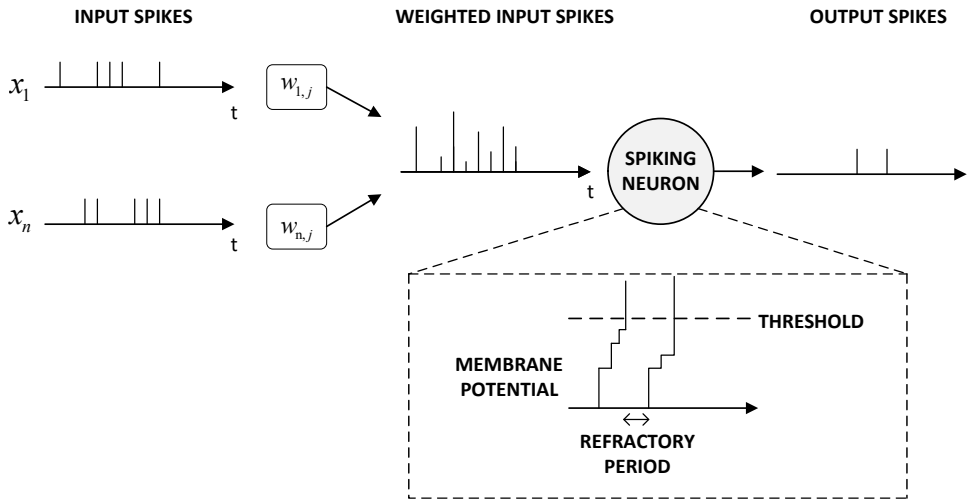


Fig. 2: Schematic diagram of operations of a spiking neuron [23].

conversion process by noting an equivalence between traditional ReLU neuron and an IF Neuron without a leaky and refractory period. Given that $I_i(t)$ is the spike input from previous layer's neuron i in time step t, V_m is the membrane potential of neurons, the synaptic integration in each time step t is expressed in the following equation

$$V_m(t+1) = V_m(t) + \sum_i w_i.I_i \tag{3}$$

After input accumulation, the neuron will check for the reset condition, generate output spikes and reset the membrane potential as follows:

$$O(t) = \begin{cases} 1 & \text{if} \quad V_m(t) \geq V_{th} \\ 0 & \text{if} \quad V_m(t) < V_{th} \end{cases} \tag{4}$$

$$V_m(t+1) = \begin{cases} 0 & \text{if} \quad V_m(t) \geq V_{th} \\ V_{th} & \text{if} \quad V_m(t) < V_{th} \end{cases} \tag{5}$$

The neurons will generate an output spike if their membrane potential cross a predefined threshold V_{th}, and the membrane potentials will be reset to zero.

It is intuitive to see the correlation between IF neuron and DNN's neuron with ReLU activation. The input spikes $I_i(t)$ are rate-coded so the average value $\mathbb{E}(I_i(t)) \propto x_i$.. If the weight vector $\mathbf{w}$ is positive, the output spike rate is $\mathbb{E}(O(t)) \propto \mathbb{E}(I_i(t))$, which corresponding to the positive region $w_i x_i$ of the ReLU function in (2). On the other hand, if $\mathbf{w}$ is negative, the input spikes never cause the neuron to produce any output spike. Hence, the output spike rate is clamped to 0.

It has been shown that the major factor affecting the classification accuracy in converted SNN models is the ratio between the threshold V_{th} and the learned weights $\mathbf{w}$ [16]. A high ratio can quickly cause a deep network to not produce any output spike for a long simulation time. A low ratio can cause the network to lose its ability to distinguish between input spikes with different weights; hence inputs information loss may occur. Major research efforts in this conversion algorithm have been dedicated to finding the balanced ratio [16] [17]. In this work, the weight-threshold balance approach in [16] has been adopted.

III. Hardware Architecture

In this section, the proposed hardware architecture is discussed in details.

A. Digital Neuron - the basic processing element

The basic processing element (PE) of the proposed hardware architecture is an efficient digital design of an IF neuron, which dynamics have been described in Section II-B. Fig. 3 shows the dataflow of one PE in different modes of operations.

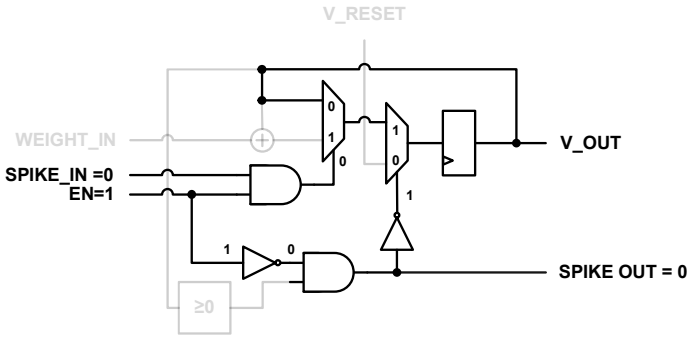
The operation of a single PE is governed by the flag *EN*. When $EN = 1$, the PE is in synaptic integration mode and will integrate the incoming input spikes with their corresponding weights. When $EN = 0$, the PE will check for the threshold condition and reset and fire if the integrated potential crosses the threshold.

1) Synaptic Integration Mode: If there is an input spike, the PE integrates its current membrane potential value with the corresponding weight. If there is no input spike, the PE skips the integration.

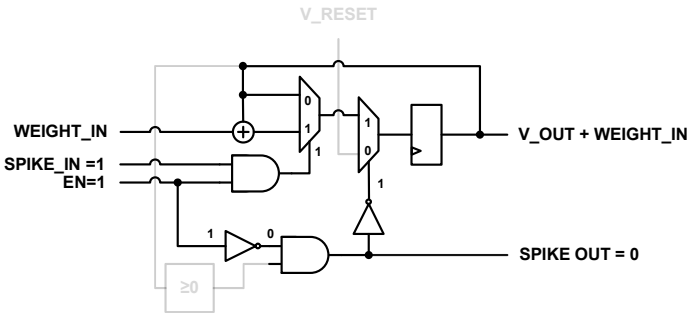
2) Reset and Fire Mode: In this mode, the PE will check its current membrane potential V_m against a predefined threshold value V_{th}. If $V_m > V_{th}$, the PE will reset V_m to a reset value V_{RESET}. In this work, V_{th} is set to 1 ,V_{RESET} is set to 0.

B. System-level Architecture

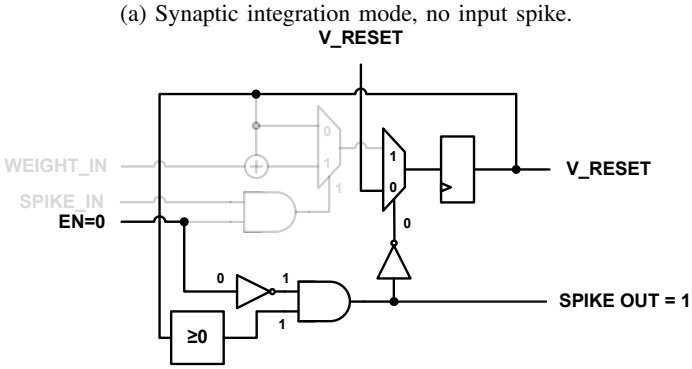
The system level architecture is presented in Fig. 4. The system supports the SNN conversion of a fully-connected, feedforward DNN with one hidden layer.



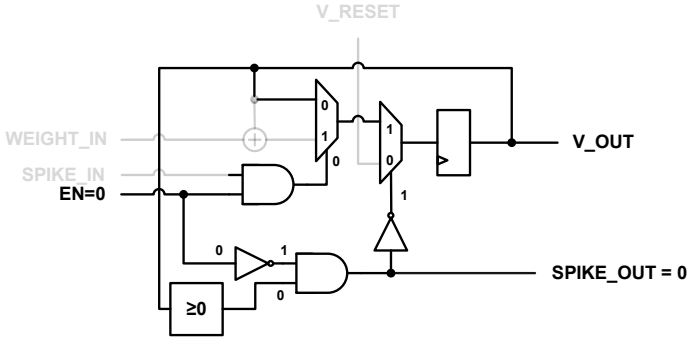
(a) Synaptic integration mode, no input spike.



(b) Synaptic integration mode, one input spike.

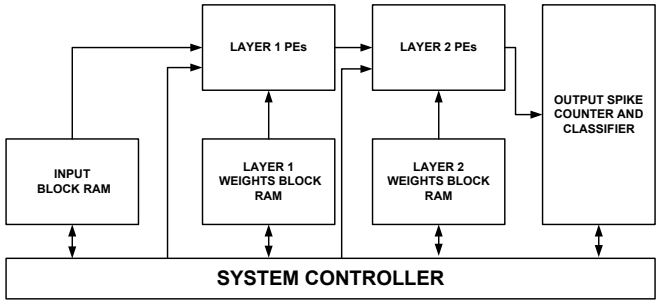


(c) Reset and Fire mode
Reset to $V_{RESET} = 0$, one output spike.

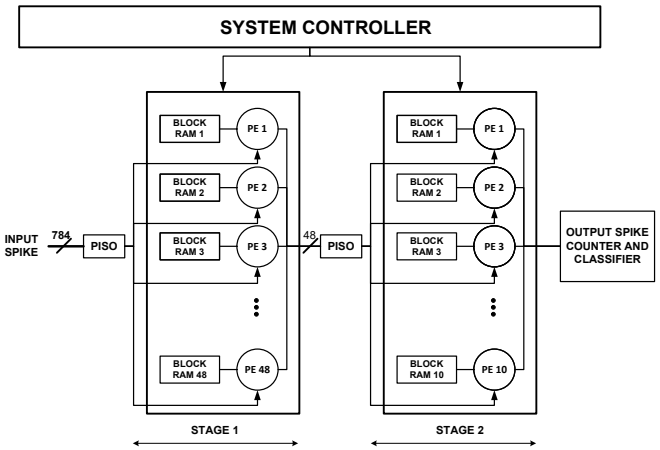


(d) Reset and Fire mode
No reset, No output spike.

Fig. 3: Microarchitecture of a single PE and its dataflow in different modes of operation.



(a) System block diagram.



(b) System top-level architecture.

Fig. 4: The system level architecture.

1) PEs complex: Each layer of PEs is grouped into a PE complex as shown in Fig. 4b. A PEs complex consists of a number of PEs working in parallel. Each PE is associated with an SRAM block of size 16b×1024. The memory bank is used to store the off-line weights after training and will be controlled by the system controller.

2) Parallel input - Serial output shift register: Two parallel input, serial output (PISO) shift registers are used to transmit the output spikes between each layer.

3) Output spike counter and classifier: This block counts the output spikes from the second PEs complex and makes the classification based on the class with the most spike count.

4) System controller: The system controller governs the operations of all the individual blocks in the system. The control flow is as follows:

- Step 1: At the arrival of new input spikes from new images, the V_m of each PE is cleared and set to zero. The trained off-line inputs are loaded into the weight memory banks.

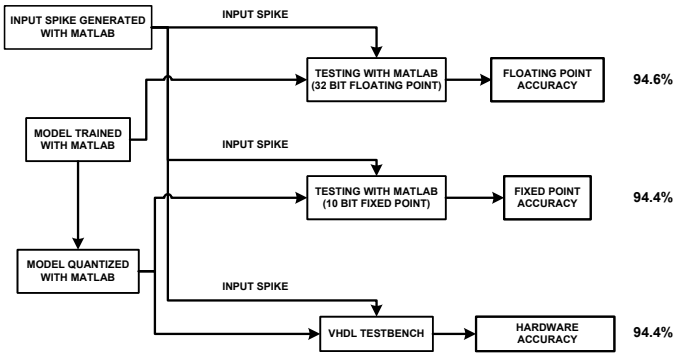


Fig. 5: Simulation flow and results.

- Step 2: At the beginning of each time step, input spikes are loaded from the host memory to the input block ram. The first PISO shift register will then load the input spikes and transmit it to the first PEs complex. The spikes are processed sequentially but all the PEs in the same PEs complex will operate in parallel.
- Step 3: The output spikes from the first hidden layer are loaded into the second PISO shift register. The spikes are processed in the second PEs complex and the output spikes will be counted at the output counter. This marks the end of one simulation time step.
- Step 4: Step 2 and 3 are repeated for a number of predefined time steps. After the system has finished processing for all time steps, the system will wait for new input images.

IV. Simulation and Implementation Results

A. Software-Hardware simulation results

The handwritten digit recognition application is chosen as a benchmark for the system's performance. One of the most popular dataset for this task is the MNIST dataset [19]. This dataset contains 60,000 training images and 10,000 testing images, with each image of size 28 $\times$ 28 pixels.

The network model size is 784$\times$48$\times$10. The network was trained and simulated in MATLAB with the open-source scripts from the authors of [16][1]. The software-hardware simulation flow is depicted in Fig.5 The off-line weights trained with 32b floating point accuracy are quantized with MATLAB to 10b fixed point format. The input spikes are generated from the testing images and are given as inputs to the inference phase.

The system architecture has been realized with VHDL at RTL level. The same set of input spikes are loaded into the VHDL testbench to verify the correctness of the hardware design. Table I shows the simulation results. It can be seen that the quantization process incurs a negligible loss of accuracy (0.2%) compared to the floating point implementation. The hardware simulation results matched the quantized software simulation results.

[1]https://github.com/dannyneil/spiking_relu_conversion

TABLE I: Simulation results

	Recognition Accuracy
MATLAB 32b Floating Point	94.6%
MATLAB 10b Fixed Point	94.4%
VHDL 10b Fixed Point	94.4%

TABLE II: Implementation ~~results~~ for a single neuron

Author	Merolla [25]	Jouber	
Publication	CICC 2011	IJCNN	
Implementation	Digital	Digital	
Technology	IBM 45*nm* SOI	65*nm*	
Neuron Area (mm^2)	0.00325	0.0005	
Neuron Type	IF	LIF	
Frequency	200*MHz*	256*MHz*	250*MHz*

pls add the fact that they all have comparable accuracy....

B. Hardware implementation results

The system has been implemented with a 45*nm* NANGATE library. This section reports the implementation results.

1) Results for a single PE: Table II compares the results of a single PE with related works. Compared to other related works with digital implementation of a single IF neuron, our design can achieve better hardware area cost and operate at comparable frequency. The author in [26] has proposed and implemented LIF neuron model in both digital and analog technology. This work has achieved 4.2$\times$ reduction in terms of area cost compared to the digital implementation [26], mainly due to the compact design of the neuron block, with the synapse weight values are implemented as simple block memories.

2) Results for system level architecture: Table III compares the results of our system level architecture with related works in the literature. Compared to the other works with the MNIST handwritten digit recognitions application, we could achieve much lower hardware area cost. This is thanks to the smaller number of neurons per core (256 neurons, 2 layer network model). However, even with a small number of neurons per core, we could still achieve a comparable performance accuracy of 94.4%.

The system-level implementation results have shown that our system is lightweight with only a core area of 15 μm^2 (19.2k 2-input NAND Gate Equivalent). At the maximum clock frequency of 250 *MHz*, the system can reach a throughput of 325,000 frames per second.

V. Conclusion

Significant research efforts have been made to push the inference phase of machine learning applications on embedded devices. In this work, we propose a lightweight neuromorphic architecture that can be applied to the handwritten digit recognition application. The simulation results show that even with limited fixed-point precision, our hardware system can reach a similar accuracy compared to floating point software implementation. Hardware implementation results have shown that our system is resource-efficient and can satisfy the constraints of real-time applications. For future works, the system

TABLE III: Implementation results for system level architecture

Author	Merolla [25]	Seo [27]	Davies [28]	Lee [29]	Zheng [30]	Knag [31]	
Publication	CICC 2011	CICC 2011	IEEE Micro 2018	ISCA 2018	ISCAS 2018	JSSC 2015	**This work**
Implementation	Digital	Digital	Digital	Digital	Digital	Digital	Digital
Technology	IBM 45*nm* SOI	IBM 45*nm* SOI	14*nm* FinFET	45*nm* TSMC	65*nm*	65*nm*	NANGATE 45*nm*
Core Area[mm^2]	4.2	0.8	0.4	9.26/7.62	1.1	3.06	0.015
Neurons per core	256	256	1024	12/72	60	256	58
Learning Type	Offline	Online	Online	Online/Offline	Online	Online	Offline
Neuron Type	IF	LIF	Adaptive LIF	LIF & Variants	Modified LIF	LIF	LIF
Frequency	200*MHz*	N.A	N.A	500/250*MHz*	167*MHz*	310*MHz*	250*MHz*
Solve MNIST	Yes	No	Yes	No	Yes	No	Yes
MNIST Accuracy	94%	N.A	96%	N.A	91%	No	94.4%

can be adapted to a more generic scalable neurosynaptic core (supports different networks topology like convolutional neural networks, recurrent neural networks etc.). Possible online learning mode to support unsupervised learning algorithm will also be considered.

ACKNOWLEDGMENT

This research is partly funded by Vietnam National University, Hanoi (VNU) under grant number QG.18.38. The authors would like to thank the National Foundation for Science and Technology Development of Vietnam (Nafosted) for their travel grant.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105.

[2] C. Szegedy, W. Liu *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[3] K. He, X. Zhang *et al.*, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[4] Y. Lecun, L. Bottou *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[5] O. Russakovsky, J. Deng *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec 2015.

[6] N. P. Jouppi, C. Young *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 1–12.

[7] Y. H. Chen, T. Krishna *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan 2017.

[8] F. N. Iandola, S. Han *et al.*, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5MB model size," *ArXiv e-prints*, Feb. 2016.

[9] A. G. Howard, M. Zhu *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv e-prints*, p. arXiv:1704.04861, Apr 2017.

[10] S. Han, X. Liu *et al.*, "Eie: Efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 243–254.

[11] M. Courbariaux, I. Hubara *et al.*, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *arXiv e-prints*, p. arXiv:1602.02830, Feb 2016.

[12] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional Neural Networks using Logarithmic Data Representation," *arXiv e-prints*, p. arXiv:1603.01025, Mar 2016.

[13] C. Ding, S. Liao *et al.*, "CirCNN: Accelerating and Compressing Deep Neural Networks Using Block-CirculantWeight Matrices," *arXiv e-prints*, p. arXiv:1708.08917, Aug 2017.

[14] A. Ardakani, F. Leduc-Primeau *et al.*, "Vlsi implementation of deep neural network using integral stochastic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2688–2699, Oct 2017.

[15] W. Gerstner and W. Kistler, *Spiking Neuron Models: An Introduction*. New York, NY, USA: Cambridge University Press, 2002.

[16] P. U. Diehl, D. Neil *et al.*, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[17] A. Sengupta, Y. Ye *et al.*, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in Neuroscience*, vol. 13, p. 95, 2019.

[18] B. Rueckauer, I.-A. Lungu *et al.*, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, p. 682, 2017.

[19] "The MNIST database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/

[20] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, May 2015.

[21] J. W. Mink, R. J. Blumenschine, and D. B. Adams, "Ratio of central nervous system to body metabolism in vertebrates: its constancy and functional basis." *The American journal of physiology*, vol. 241, pp. R203–12, Sep 1981.

[22] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659 – 1671, 1997.

[23] M. Bouvier, A. Valentian *et al.*, "Spiking neural networks hardware implementations and challenges: A survey," *J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 2, pp. 22:1–22:35, Apr. 2019. [Online]. Available: http://doi.acm.org/10.1145/3304103

[24] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLOS Computational Biology*, vol. 3, no. 2, pp. 1–11, 02 2007. [Online]. Available: https://doi.org/10.1371/journal.pcbi.0030031

[25] P. Merolla, J. Arthur *et al.*, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, Sep. 2011, pp. 1–4.

[26] A. Joubert, B. Belhadj *et al.*, "Hardware spiking neurons design: Analog or digital?" in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, June 2012, pp. 1–5.

[27] J. Seo, B. Brezzo *et al.*, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, Sep. 2011, pp. 1–4.

[28] M. Davies, N. Srinivasa *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, January 2018.

[29] D. Lee, G. Lee *et al.*, "Flexon: A flexible digital neuron for efficient spiking neural network simulations," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, June 2018, pp. 275–288.

[30] N. Zheng and P. Mazumder, "A low-power hardware architecture for on-line supervised learning in multi-layer spiking neural networks," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.

[31] P. Knag, J. K. Kim *et al.*, "A sparse coding neural network asic with on-chip learning for feature extraction and encoding," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 1070–1079, April 2015.