Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 10, Number 1, pages 30–35, January 2021

An Energy Efficient Stochastic+Spiking Neural Network

Takeshi Nomura, Renyuan Zhang, Yasuhiko Nakashima
Nara Institute of Science and Technology (NAIST)
Nara, Japan
{nomura.takeshi.no3, rzhang, nakashim}@is.naist.jp

***Abstract*—A feasibility study of stochastic+spiking neural network is presented for reducing the hardware implementation cost. By using a set of time-based stochastic computing (TBSC) circuits, the stochastic numbers (SNs) in continuous time-domain are directly fed into the input layer of the spiking neural networks (SNNs) without any additional spike-coding mechanism. The analog circuits behaving as synapses and neurons are designed to fit the TBSC coding and generate spikes for the rest of layers. The transistor counting is compact as 22 per synapse and 22 per neuron. Several real-world tasks based on pattern recognition data-set including MNIST are verified and estimated. For proof-of-concept, a 0.18um CMOS technology is used to design and simulate our proposed SNN. Implementing the exampled pattern recognition tasks, the recognition accuracy loss is below 4% compared to well-trained artificial neural networks (ANNs). The average firing energy is 0.94pJ per spike, which is 0.5x of state-of-art of low power SNN implementations. The energy consumption of MNIST is estimated as 0.88uJ per classification.**

***Index Terms*—Spiking neural network, time-based stochastic computing, energy efficient**

I. INTRODUCTION

Artificial Neural Networks (ANNs) have archived state-of-the-art performances in the wide field of applications such as image recognition [1], [2], object detection [3], and speech recognition [4]. In recent decades, the greedy demands on the complexity and quality of service at application-end lead to the deep and complicated NNs, which are usually implemented by massive computations [5], [6]. Due to the large amount of computations, one of popular fashions is to implement ANNs (both of training and inference) by multi-core platforms such as GPGPU. The critical parts of ANN implementations are large multiply-accumulations (MACs). It has been found that general purpose accelerators like GPGPUs hardly reach extremely high speed for some complex tasks. A general strategy for accelerating ANN in further is carrying out necessary computations on VLSI chips for domain specified accelerations (DSAs) [7], [8]. However, straight implementations of MACs are resource- and power-hungry for any types of hardware. To breakthrough the general trade-off between performance and cost, it is expected to escape from the conventional ANN topology.

Spiking Neural Networks (SNNs), a biological inspired computational model, has appeared the potential of ultra efficient and convenient implementations as a promising candidate for substituting ANNs. In SNN theory, data is represented by a series of stimulus similarly to the processing styles inside bio-brain [9], [10]. By encoding the information to stimulus (known as spikes), the massive and expensive MACs of ANNs are avoided through observing and processing spikes in specific schemes [11]. Since SNNs are event-driven, most of its neurons and synapses are off-state and idle till events. The processing power could be naturally lower than straight computations for ANNs in further. Thus, the spiking fashion of NNs lead the current trend of implementations of compact hardware with low power [12]–[14]. Harvesting rich benefits in terms of hardware efficiency, SNNs processors still suffer from two challenges on the other hand. Firstly, the training process of SNN is not as friendly as ANNs due to the discrete representation of data. One of solutions is to implement on-chip training by employing some advanced devices or materials, memristor (RRAM [15]) and floating gate transistors [16] for instances. Another practical strategy is implementing the training phase off-chip through any friendly schemes such as ANN; then, well-trained NNs are converted into spiking schemes from specific methodologies [17]–[19] for inference on-chip. In this manner, high quality inference can be achieved by efficient hardware [20]. Beyond the training strategies, the data form conversion is a serious and common bottleneck of any type of SNNs. In general SNN theory, complex information can be carried on various appearance-features of spikes including density, distribution, rate, and timing. For instance, a typical representation is known as spike time dependent plasticity (STDP) [20], which convert the data into the spike series appearing as Poisson distribution. Obviously, the translation or conversion of data form is expansive in sense of hardware resource and processing time. Thus, a translation-free scheme of SNN is demanded for efficient implementation. Various features of spikes can be employ to carry complex information including

In this work, the energy efficient scheme of SNN is proposed for implementations of off-chip training and on-chip inference. To perform the inference, the corresponding ANNs are pre-trained and converted to the spiking scheme. The data representation is achieved by the novel time-based stochastic computing (TBSC) circuits without additional data-converting mechanism and hardware, which are employed as the input layer of proposed SNN. In this manner, the hardware resource and energy consumption is reduced. In the hidden and output layer, the analog current driving circuits along with the membrane capacitors are designed as neurons

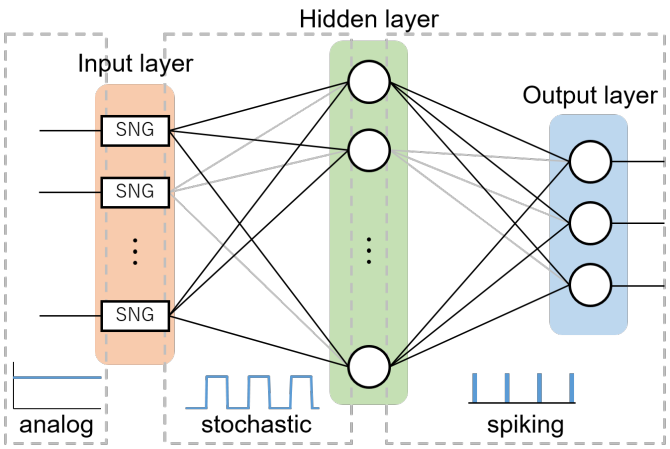

Fig. 1. Overview of stochastic+spiking neural network

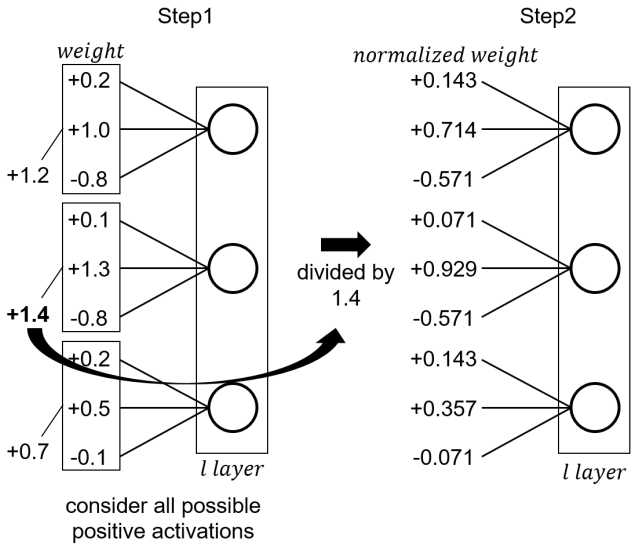

Fig. 2. Model-based normalization algorithm

and synapses, where the multiplication and accumulation are avoided. The transistor counting of each synapse and neuron is 22, respectively. For proof-of-concept, several toy-examples are demonstrated by classifications through real-world dataset. All the classification tasks are performed with the accuracy loss below 4% compared to well-trained ANNs. The energy per spike and per inference are $0.94pJ$ and $0.88uJ$, respectively, which are 0.5x and 0.79x of state-of-art SNNs.

II. PRELIMINARY

A. Overview

The overview of proposed stochastic+spiking neural network (SSNN) is illustrated in Fig. 1. A general full-connection topology of NN is employed for behaving inference on-chip. The analog signals carrying data are fed into the proposed SSNN as input layer. All the synapses which link the input layer to hidden layer are performed and processed in the form of time-based stochastic numbers (see the details in the following section, SNG indicates the stochastic number generator); and the synapses linking the hidden layer to output layer are represented by ordinary spiking scheme. In this manner, the complicated conversion and generation of spike coding is substituted by our novel stochastic computing circuits.

B. Integrate-and-Fire model

The spiking neuron model used in this work is the simple integrate-and-fire (IF) model without any leak. The IF spiking neuron accumulates the input spikes $\mathbf{x}_i(t)$ weighted by w_i to the membrane potential $v_{mem}(t)$ at time-step t and generates a spike when the membrane potential exceeds a certain threshold v_{th}. When a spike is generated, the membrane potential is reset to the initial voltage.

$$v_{mem}(t+1) = v_{mem}(t) + \sum_i w_i \mathbf{x}_i(t) \tag{1}$$

Note that the neuron dynamics is independent of the actual magnitude of the time-step [18].

C. Conversion-based approache

Since our proposed circuit is dedicated to off-chip learning, the necessary weights for inference must be learned externally. For the off-chip training, various training algorithms have been reported, local learning like STDP, gradient-based learning, and the conversion approach for instances. In this work, the ANN-to-SNN conversion technology is adapted for simplify the training process, which trains an ANN preliminary and converts all the parameters into the form of SNN. The Model-based normalization [17] is used to convert parameters. This normalization re-scales all the weights by the maximum possible positive input of the layer. Figure 2 shows the step-by-step procedure for this normalization.

Firstly, the sum of all positive weights connected to each neuron on l layer is calculated; and the biggest one is picked up (step1). Then, all the weights are re-scaled by using the picked summation (step2). At last, the normalization is processed for all the layers. The detail and proof of such technology is out of the scope of this work, which is seen in SNN theory by P. Diehl et al [17].

D. Data-encoding and Spike-representation

Figure 3 illustrates a simplifed example of the data representation and propagation. In this example, a simple data path is introduced for entire link from the analog input to the spiking output in the proposed SSNN. In the input layer, the time-based stochastic computing (TBSC) is employed to carry the data. The duty-cycle of periodic signals is used to represent a specific data within the range of 0 to 1 (seen as pulse width modulation (PWM)) [21]. Feeding this PWM coupled by the specific weights into a neuron N_1 of next layer, the multiplication of input and weights is reflected by the output of N_1, which is seen as a series of spikes V_{N1_out}. Then, the spikes are fed into the former neuron N_2 coupled with its

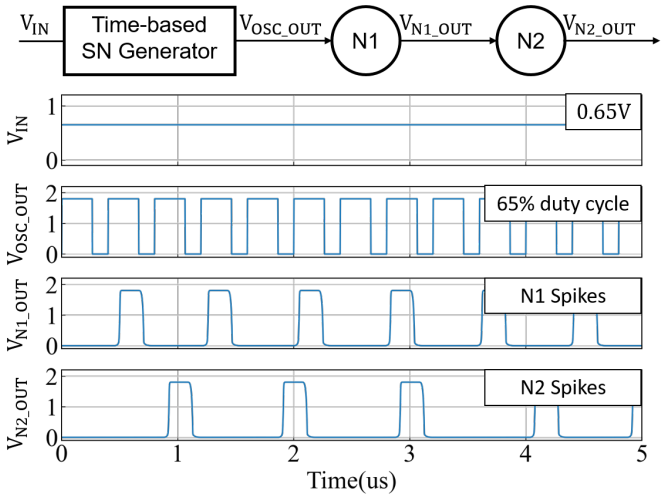

Fig. 3. Example of data-encoding and spike-representation

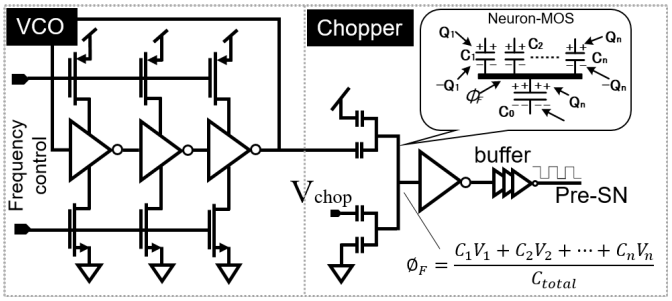

Fig. 4. Time-Based Stochastic Nummber Generator

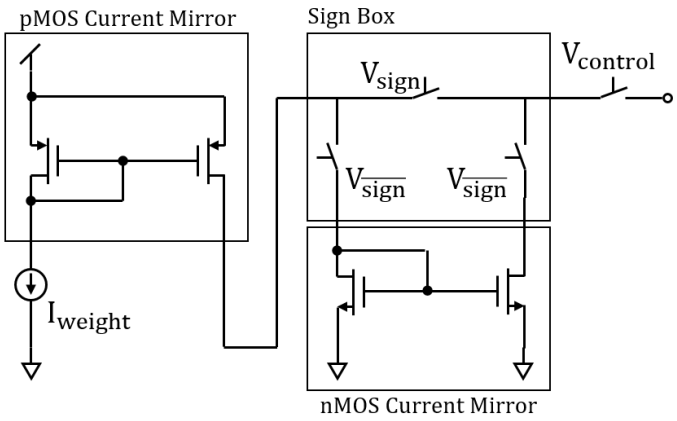

Fig. 5. Schematic diagram of the synapse.

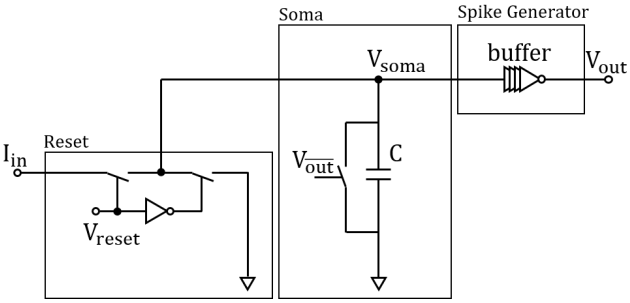

Fig. 6. Schematic diagram of the neuron.

corresponding weight. Obviously, the spike-firing rate is used to perform NN behavior in this work.

One of the key issues of such an SSNN is the efficient generation of time-based SNs. A compact SNG is applied to convert analog signal to PWMs as shown in Fig. 4. The ring-oscillator powered by controllable current is designed to generate saw-tooth oscillation. Then, the capacitor-coupling technology is used on the gate of MOS transistors (known as Neuron-MOS mechanism) for linearly programming the gate potential and switching threshold. This part chops the oscillation into arbitrary duty-cycle from the shift of threshold level. In this manner, the time-based stochastic number is efficiently generated. As the bonus, the PWM signals compresses the rate/density of spikes compared to the conventional scheme of SNN, which helps reducing the processing time and energy consumption. Time-baed stochastic computing (TBSC) have been used for our data-encoding.

III. HARDWARE IMPLEMENTATION

The snapse and neuron implementations in any type of NNs are expensive due to large scale of MACs, which are hardly realized in fully parallel on-chip. An important benefit of SNN is that MACs are substituted by the spike accumulations with flexible impacts (weights for instance).

The current mode synapse circuit is designed in this work as shown in Fig. 5. For single synapse, a current source is switched to link the neuron by the pulses generated from the previous layer (either SN or spikes). Obviously, the strength or impact lies on both of current value and pules rate or width. This phenomenon reflects the multiplication between input and weights with signs efficiently. When the positive (negative) weighted input comes, the V_{sign} is set to HIGH (LOW) and the current flow out from (into) this specific synapse. By using the proposed circuit, only a set of current mirror and several CMOS switches are necessary with 22 transistors per synapse.

Figure 6 shows proposed neuron circuit. Collecting the current from all the synapse branches into I_{in}, a membrane capacitor is employed to accumulate the impact of all the spikes in both of time and space domain. Entire neuron circuits consists of three functional blocks (reset, soma, spike generator) of 22 transistors in total. Soma block receives the weighted current from synapses as input and charges and discharges its capacitor C. When the potential of the capacitor C exceeds a threshold of the buffer in the spike generator block, the spike generator block becomes active and generates one spike. After that, a switch connected in parallel to the capacitor C is turned on, and the potential of the capacitor C is reset to 0V (resetting state). Note that when the membrane potential is 0V, even if a negative input is received, it does not fall below 0V.

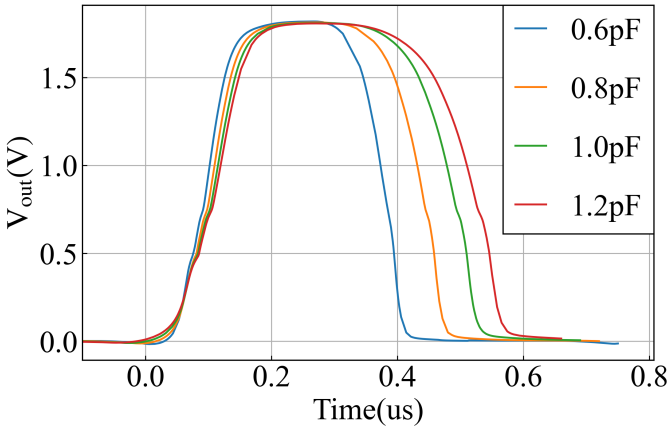

Fig. 7. Spike widths with respect to the capacitance of buffer. As the capacitance of the buffer increases, the width of the spike increases.

TABLE I
TRAINING FONFIGURATION FOR ANN

Network Stracture	fully-connected 13-30-3
Activation Function	Relu without Bias
Data Set	Wine data-set [22]
Optimizer	Adam [23]
Normalization	$0.2 \sim 1$

Here, the spike width depends on the configuration of buffer block. Although the spike strength is regularized for entire SSNN, various widths lead to various features of spike rate. A suitable width should be set according to the corresponding mathematical behavior as expected. By configuring buffer capacitance, the spike width is adjusted in any feature as shown in Fig. 7 by circuit simulation results.

IV. EXPERIMENTAL SETUP

In this section, we outline the details of the experimental setup. For instance, a toy example of small NN is demonstrated for conveniently observing the signals inside our proposed SSNN.

A. Traing ANN and Convert to SNN

First of all, a small scale of NN with three layers are introduced as a preliminary experiment. The real-world data-set for wine classification with 13 dimensions and three classes [22] are used for observation. The ANN was constructed and trained according to Tab. I and converted into SNNs of the same structure using a conversion approach [17]. The input data is normalized given by

$$x_{norm,i} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{2}$$

, where x_{norm} is normalized to x, and x_{min} is the minimum value of x and x_{max} is the maximum value of x. The output neurons 1st, 2nd, and 3rd correspond to labels 1, 2, and 3, respectively.

B. Circuit simulation

The network is constructed as $13-30-3$ as shown in Fig. 8(a) for the toy example. Employing the circuits mentioned above, the VLSI circuit is designed simulated for entire SSNN in a $0.18\mu m$ CMOS technology. The converted weights were set by the current source on the synapse and VDD is set to 1.8V. The input voltage range of the TBSC generator is 0.2V$\sim$1V, and the generator output duty ratio is proportional to the input voltage. If the input is 0.5V, the duty cycle is 50% (the on-time is 200ns) for instance. As the observation of inference result, three output neurons are observed from their firing rates. From the winner-take-all mechanism, the most-frequently firing neuron indicates the classification label. Similarly to other SNN implementations, the observation time is flexible according to configuration, specific data-set, and expected accuracy. In this work, a constant time window of $5\mu s$ is applied per inference.

V. CIRCUIT SIMULATION RESULT

A. Toy example of small SSNN

In this example, 13 input signals, 30 hidden neurons, and 3 output neurons can be observed by circuit simulation results for 178 test samples. All the samples are classified by the proposed circuit from the simulation results. Among these 178 samples and various observation points, one of samples along with three output neurons and two exampled hidden neurons are demonstrated as shown in Fig. 8(b). In this demonstration, the correct label is expected as the first neuron winning. The voltage waveform is, from the top, the output of the TSBC generator, the output of the hidden layer neurons and membrane potential, and the outputs of the three output layer neurons. The first, second, and third output neurons are firing for seven, three, and zero times, respectively. Therefore, the classification result from the circuit simulation is correctly observed. By the circuit simulations for all 178 samples, the performance and cost of proposed SSNN is summarized in the Tab. II. As the key performance, the classification accuracy is investigated. The hardware implement of SSNN leads to the accuracy loss of 4% compared to the well trained ANN as the upper bound. Since the energy consumption depends on the dynamic of inference data, the average energy with respects to synapse, neuron and spike is summarized.

Figure 9 shows the comparison between the ideal waveform (dotted line) and the circuit simulation (solid line) for the membrane potential of the 5th, 6th, and 7th neurons in the hidden layer. The ideal waveform is calculated by the mathematical model with python based on the weights obtained by the convert approach. The membrane potential of V_{N5_MEM} is 0V, as well as the ideal one. The amplitudes of the ideal waveform of both V_{N6_MEM} and V_{N7_MEM} are consistent with each other. However, there is an error in the period of V_{N6_MEM} in the ideal waveform and circuit simulation. It is seen that the physics behavior from circuit simulation is almost equivalent to the mathematical model.

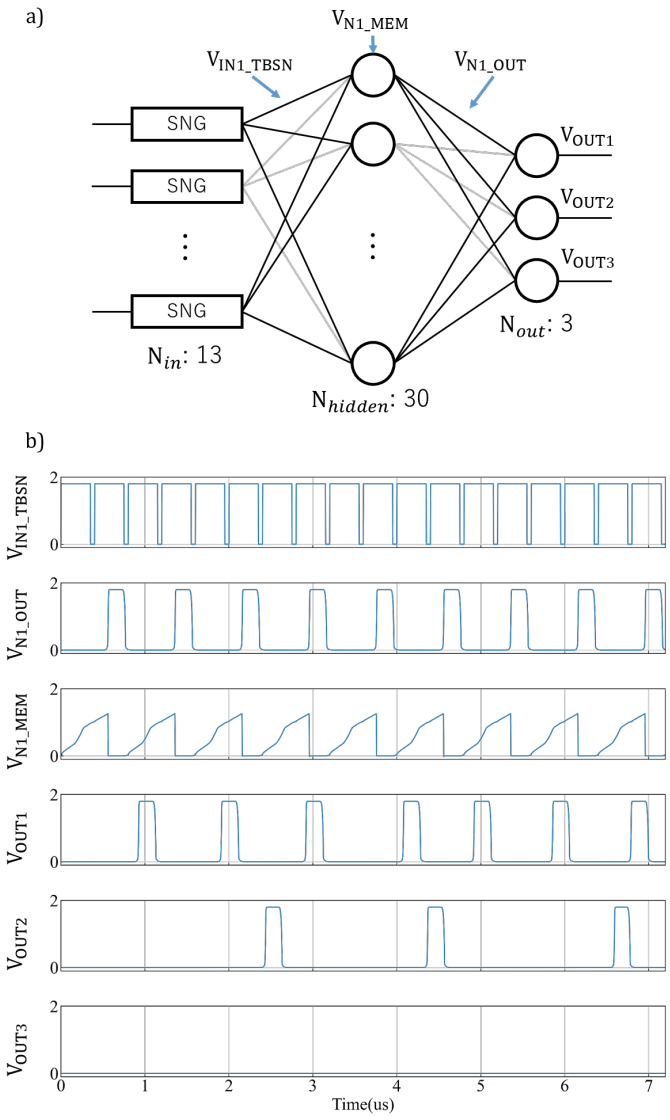

Fig. 8. Structure of small SSNN as toy example and its circuit simulation results

TABLE II
SUMMARY OF 13-30-3 SSNN IMPLEMENTATION

	Performance
Accuracy Loss	*4%
Total power Consumption	**1.424(mW)
Average Synapse power	0.294(uW)
Average Neuron power	12.12(uW)
Energy per spike	0.94(pJ/spike)

*Original ANN has 99.9% accuracy, 99.9% after the conversion to SNN, and 96.0% when running the circuit.
**33 neurons, 460 synapses and 13 TBSC generators are at 0.144mW, 0.4mW and 0.88mW respectively

B. Comparisons

For verifying the performance of larger SSNN than previous toy example, additional experiments have been done by using the MNIST data-set for recognizing hand writing letters. The

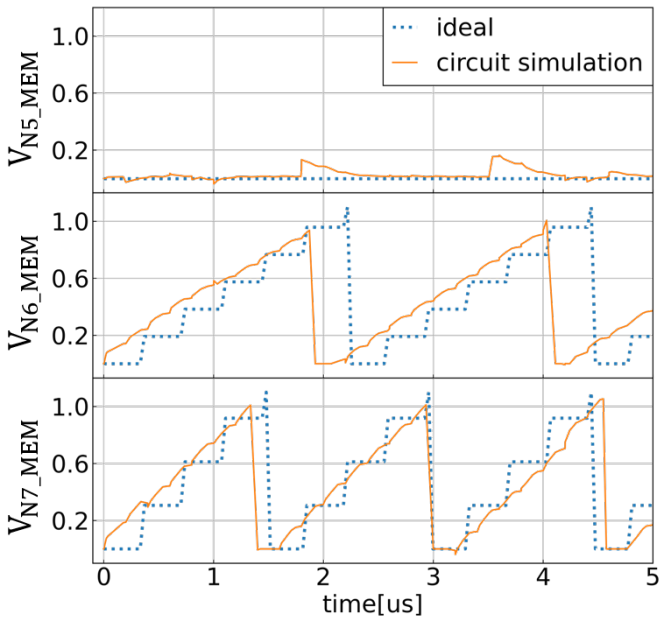

Fig. 9. Dynamic behaviors of circuit and mathematical model

TABLE III
COMPARIOSNS AMONG VARIOUS SNN IMPLEMENTATION

	[24]	[25]	Proposed neuron
Technology	65nm	65nm	180nm
Analog/Digital	Analog	Digital	Analog
Energy per spike (pJ/spike)	2	41.3	0.94
Max. spike rate (Mspike/s)	1.9	1.9	5.3

plain pixel values are directly fed into SSNN as input data. Namely, only full connection NN is employed for reorganization. Only the mathematical model by python is used to emulate the behavior of our SSNN instead of real circuit simulation due to the explosion of simulation time by HSPICE. The hardware cost is estimated on the basis of above toy example. From Tab. III, the proposed neuron circuit consumes 0.5x energy of the analog one [24] and 43.94x energy per spike than the digital one [25] in state-of-art. Therefore, the proposed SSNN architecture can achieve lower energy per classification as shown in Tab. IV in general.

We estimated the performance of MNIST based on Table II for comparison and shows this comparison in Table IV

VI. CONCLUSION

A topology of stochastic+spiking neural network is proposed in this work. The off-chip training and on-chip inference is adapted. By converting the data into the form of time-based stochastic number, the complicated spike coding algorithms/-circuits are avoided. A set of current mode circuit for synapses and neurons are designed and simulated in a $0.18\mu m$ CMOS technology. Assembling massive synapses and neuron in fully parallel, several inference tasks for real-world data-sets are demonstrated with the average accuracy loss of 4%. From the circuit simulation results, the proposed SSNN circuit achieves

TABLE IV
COMPARISON FOR MNIST IMPLEMENTATION

	[26]	[27]	Proposed Circuit
Network Type	SNN	ANN	SNN
Analog/Digital	Digital	Digital	Analog
Technology	65nm	28nm	180nm
Coding Scheme	Rate	N/A	Time-based SC
Network Topology	FC 3 Layer	FC 5 Layer	FC 3 Layer
Number of Neruons	*316	1562	510
Energy per Classification[uJ]	1.12	360	**0.88

*MNIST data-set down sampled from 28×28 to 16×16.
**Cost is estimated on the basis of previous toy example.

low energy of 0.94pJ per spike, which is 0.5x of state-of-art of low power SNN implementations. Moreover, the large scale NN for MNIST is also emulated by mathematical model and estimated by circuit simulations with the energy consumption of 0.88uJ per classification, which is superior to state-of-art digital and analog SNN implementations.

ACKNOWLEDGMENT

This work was supported by JST, PRESTO Grant Number JPMJPR18M7, Japan. The authors would like to thank the VLSI Design and Education Center (VDEC) of the University of Tokyo in collaboration with Rohm Corporation, Synopsys, Inc., and Cadence Design Systems, Inc.

REFERENCES

[1] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. volume 22, pages 3207–3220. MIT Press, 2010.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[4] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[7] Arash Ardakani, Carlo Condo, and Warren J Gross. Sparsely-connected neural networks: towards efficient vlsi implementation of deep neural networks. *arXiv preprint arXiv:1611.01427*, 2016.

[8] Sourya Dey, Kuan-Wen Huang, Peter A Beerel, and Keith M Chugg. Pre-defined sparse neural networks with hardware acceleration. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):332–345, 2019.

[9] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Third generation neural networks: Spiking neural networks. In Wen Yu and Edgar N. Sanchez, editors, *Advances in Computational Intelligence*, pages 167–178, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[10] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[11] Ulrich Roth, Axel Jahnke, and Heinrich Klar. Hardware requirements for spike-processing neural networks. In José Mira and Francisco Sandoval, editors, *From Natural to Artificial Neural Computation*, pages 720–727, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[12] E. Chicca, D. Badoni, V. Dante, M. D'Andreagiovanni, G. Salina, L. Carota, S. Fusi, and P. Del Giudice. A vlsi recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory. *IEEE Transactions on Neural Networks*, 14(5):1297–1307, 2003.

[13] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[14] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[15] Tianqi Tang, Lixue Xia, Boxun Li, Rong Luo, Yiran Chen, Yu Wang, and Huazhong Yang. Spiking neural network with rram: Can we use it for real-world application? In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 860–865. IEEE, 2015.

[16] S. Ramakrishnan, P. E. Hasler, and C. Gordon. Floating gate synapses with spike-time-dependent plasticity. *IEEE Transactions on Biomedical Circuits and Systems*, 5(3):244–252, 2011.

[17] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. Liu, and M. Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.

[18] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13, 2019.

[19] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: Spiking neural network for energy-efficient object detection. *arXiv preprint arXiv:1903.06530*, 2019.

[20] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575:607–617, 11 2019.

[21] M. H. Najafi, S. Jamali-Zavareh, D. J. Lilja, M. D. Riedel, K. Bazargan, and R. Harjani. An overview of time-based computing with stochastic constructs. *IEEE Micro*, 37(6):62–71, 2017.

[22] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Antoine Joubert, Bilel Belhadj, and Rodolphe Héliot. A robust and compact 65 nm lif analog neuron for computational purposes. *2011 IEEE 9th International New Circuits and systems conference*, pages 9–12, 2011.

[25] A. Joubert, B. Belhadj, O. Temam, and R. Héliot. Hardware spiking neurons design: Analog or digital? In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5, 2012.

[26] 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)*, pages 1–5, 2018.

[27] Paul N Whatmough, Sae Kyu Lee, Hyunkwang Lee, Saketh Rama, David Brooks, and Gu-Yeon Wei. 14.3 a 28nm soc with a 1.2 ghz 568nj/prediction sparse deep-neural-network engine with¿ 0.1 timing error rate tolerance for iot applications. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 242–243. IEEE, 2017.