Hardware-Efficient Accelerator for Spiking Neural Networks with Synchronous Spike Storage and Weight Address Control

Jeong-Eun Ko, Woo-Vin Choi, Min-Seok Kim, and Joo-Hyung Chae[a]
Department of Electronics and Communications Engineering, Kwangwoon University
E-mail : {zasx030531, q681, kms12051, jhchae}@kw.ac.kr

***Abstract*—Spiking neural networks (SNNs) are attracting attention for their energy-efficient computation and real-time processing capabilities, as they emulate the time-based signal processing of the human brain. However, prior hardware implementations have the disadvantages of high resource usage and long inference latency. To solve this problem, we present a hardware-efficient FPGA-based SNN architecture. Through voltage scaling, hardware-efficient optimization of synaptic transmission delay, and reformulation of the neuron equation with power-of-two scaling, hardware resources are greatly reduced. Furthermore, the synchronous spike storage and controller modules are introduced to simplify the data path. The proposed design was implemented on the XC7Z7020 board, achieving an inference latency of 1.41 ms/image and a 103.8× speedup over the CPU on the MNIST dataset with only a 0.3% accuracy drop.**

***Keywords*—field-programmable gate array (FPGA), hardware accelerator, spiking neural network (SNN), real-time processing**

I. INTRODUCTION

Spiking neural networks (SNNs) are inspired by the event-driven process of biological neurons [1]. By leveraging temporal spike-based information processing, SNNs can achieve better energy efficiency than conventional artificial neural networks [2]. However, SNNs inherently require multiple timesteps for information transmission, which results in significantly higher inference latency compared to convolutional neural networks and deep neural networks [3].

Recently, various studies on field-programmable gate array (FPGA)-based SNN implementations have been reported, but they have several limitations. Prior work in [4] proposed a memory-efficient compression and join mechanism but did not achieve significant speedup in practice. Another approach [5] simplifies SNN multiplications and achieves approximately a 20.5× speedup; however, resource consumption is still high, using more than approximately 40k look-up tables (LUTs) and approximately 60k flip-flops. Yet another study [6] supports up to 16,384 neurons on an FPGA, but the inference latency is limited to 6.21 ms per image, which is insufficient for real-time processing. In other words, the above approaches still require a large number of neurons, consume excessive FPGA resources, and suffer from long inference latency, thereby limiting scalability and practical applications.

To address these challenges, we propose a hardware-efficient FPGA-based SNN accelerator. Our architecture introduces the following key techniques: 1) 10× voltage scaling is applied to mitigate the loss of small values while using the 16-bit fixed-point precision; 2) Synaptic transmission delays are optimized, improving hardware resource efficiency and inference accuracy; 3) Neuron equations are reformulated with power-of-two denominators, avoiding LUT-based division operations. Consequently, LUT usage is reduced from 214,267 to 35,598, LUTRAM usage from 300 to 0, and flip-flop usage from 25,698 to 19,697; and 4) We introduce the synchronous spike storage and controller module to mitigate data path complexity caused by parallelization in hardware module design. These efficiently store and manage spike information and coordinate weight delivery. Consequently, the proposed synchronous design further reduces LUT usage by 14.97% and flip-flop usage by 37.55% compared to the asynchronous design, while simplifying the overall structure. Thanks to these features, the proposed design achieves an inference latency of 1.41 ms per image, which is a 103.8× speedup over the baseline (CPU, Intel Core i3-1115G4, 16-bit fixed-point precision). It incurs only a 0.3% accuracy drop while maintaining performance.

The rest of this paper is organized as follows. Section II describes the design considerations for hardware optimization. Section III presents the hardware implementation of the SNN. Section IV details the experimental setup and results, and Section V concludes the paper.

II. DESIGN CONSIDERATIONS FOR HARDWARE OPTIMIZATION

A. Data Type, Precision Bit, and Voltage Scaling

The SNN model used in this study is adopted from an open-source implementation with 32-bit floating-point precision [7] based on the MNIST dataset. The network consists of 784 input neurons, 100 excitatory neurons, and 100 inhibitory neurons. Its connectivity structure is illustrated in Fig. 1: The input neurons are fully connected to all excitatory neurons; each excitatory neuron is connected one-

a. Corresponding author; jhchae@kw.ac.kr

Manuscript Received Jan. 21, 2026, Revised Apr. 10, 2026, Accepted Apr. 11, 2026

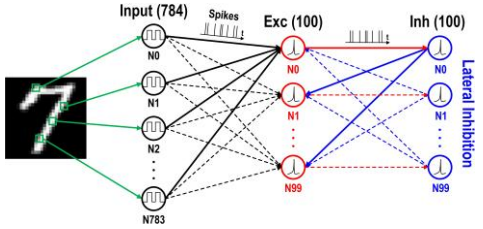
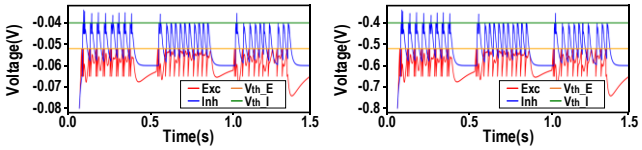


Fig. 1. The architecture of a spiking neural network.



(a) (b)

Fig. 2. Spike waveforms: (a) baseline and (b) 10× voltage scaled.

to-one with an inhibitory neuron; and each inhibitory neuron sends inhibitory signals back to all excitatory neurons except the one connected to itself, thereby implementing lateral inhibition.

Although the original network provides sufficient accuracy, it requires a large number of hardware resources. Therefore, the model should be transformed to a lower bit width of the fixed-point format to save hardware resources. However, the reduced precision can cause small values of the membrane potential to converge to zero during the quantization process, thereby degrading inference accuracy. To address this issue, the membrane potential range [–0.08 ~ –0.04] is extended to [–0.8 ~ –0.4]. By increasing the voltage by a factor of 10, more effective bits can be assigned within the reduced precision. Fig. 2(a) illustrates the spike waveform of the original model, where the spike voltage is measured for each image input over 0.5 s. Fig. 2(b) presents the spike waveform after applying 10× voltage scaling. As observed, the overall waveform shape remains consistent before and after scaling.

To search for optimal data precision, we analyze the inference accuracy according to the number of fractional bits, while the 1-bit sign and 3-bit integer are fixed (Fig. 3). All accuracy values are evaluated using quantized fixed-point representations. As shown, the 12-bit fractional bit achieves 76.5% accuracy, but fractional bits lower than 12 bits exhibit poor performance. The fractional bits greater than 12 bits demonstrate improved accuracy, but the improvement is minimal despite the additional hardware usage. Therefore, in this study, a 10× voltage-scaled model with 16-bit fixed-point precision (1-bit sign, 3-bit integer, and 12-bit fraction) is adopted, providing an effective balance between the hardware resource and inference accuracy.

B. *Hardware-Efficient Optimization of Synaptic Transmission Delay*

The baseline algorithm [7], implemented in the Brian2 environment, incorporates synaptic transmission delays of approximately 5 ms and 2.5 ms for E-to-I and I-to-E connections, respectively, to emulate axonal signal propagation. While such delays enhance biological plausibility in software simulators, their hardware realization

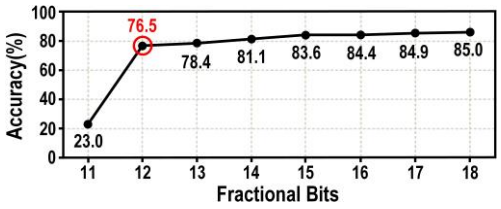


Fig. 3. Inference accuracy according to the number of fractional bits.

requires significant overhead, such as flip-flop chains or delay buffers. Since this work adopts a rate-coding scheme, where information is represented by the number of spikes within a fixed time window rather than precise spike timing, the effect of synaptic transmission delay is negligible. Therefore, in this work, we excluded these delays to prioritize hardware resource efficiency. As shown in Figs. 2(a) and (b), the artificial propagation latency causes the firing delays of excitatory and inhibitory spikes. In contrast, when delays are removed, spikes occur immediately without such temporal delays.

Optimizing the synaptic transmission delay provides two key advantages. First, it improves hardware resource efficiency. Since the impact of synaptic transmission delay on the functional behavior of the SNN is negligible, the delay can be removed without degrading performance. Second, it enhances inference accuracy. The proposed architecture is based on a WTA (winner-take-all) mechanism. When synaptic transmission delay is present, inhibitory feedback is delayed, allowing multiple excitatory neurons to generate additional spikes simultaneously. This reduces the distinctiveness of spike counts and weakens WTA competition. In contrast, removing the delay enables immediate inhibitory feedback, resulting in sharper WTA competition and more discriminative feature representations. Consequently, the inference accuracy improves from 79.7% to 83.5%, both measured under the floating-point (FP32) software model.

C. *Neuron Equation Optimization by Power-of-Two Scaling*

In hardware implementations, divisions by arbitrary constants require additional resources because they cannot be directly mapped to simple logic. Such operations are typically realized using LUTs, which significantly increase resource usage. To reduce this overhead, the neuron equations are reformulated by approximating the denominators with powers of two, replacing the constants 1000, 100, and 50 with 1024, 128, and 32, respectively. Since divisions by powers of two can be implemented through simple bit shifts, additional LUTs are not required.

The reformulated equations are presented in (1)–(5). $v(t)$ denotes the membrane potential, while ge and gi represent the excitatory and inhibitory synaptic conductance, respectively. (1)–(3) correspond to excitatory neurons, where (1) updates the membrane potential according to synaptic inputs, (2) describes the gradual decay of excitatory conductance, and (3) represents the change in inhibitory conductance:

$$v(t+1) = v + \frac{-v(ge+gi+1) - gi - 0.6}{1024} \quad (1)$$

$$ge(t+1) = ge + \frac{-ge}{32} \quad (2)$$

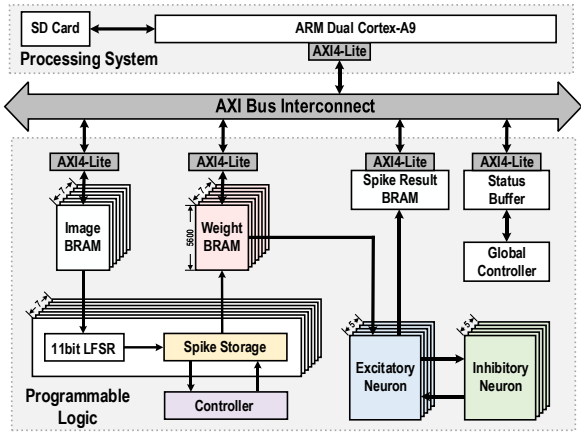


Fig. 4. Overall architecture of the proposed FPGA-based SNN accelerator.

TABLE I. Spike Counts According to Bit Width by Class

	0	3	5	7	9
CPU	3,429	4,105	4,421	2,584	3,103
10 bit	6,026	7,804	7,904	4,614	5,431
11 bit	3,016	3,887	3,928	2,306	2,715
12 bit	1,493	1,925	1,955	1,157	1,349

$$gi(t+1) = gi + \frac{-gi}{128} \tag{3}$$

Similarly, (4) and (5) describe inhibitory neurons, where (4) updates the membrane potential, and (5) expresses the decay of excitatory conductance:

$$v(t+1) = v + \frac{-v\,(ge+1) - 0.6}{1024} \tag{4}$$

$$ge(t+1) = ge + \frac{-ge}{32} \tag{5}$$

The proposed equation-reformulating method reduces LUT usage by approximately 83%, lowering the count from 214,267 to 36,376. Although inference accuracy slightly decreases, the power-of-two approximation efficiently reduces resources in the SNN hardware design.

In addition, the constant 0.6 in the equations represents the resting membrane potential of the neuron and is treated as a fixed parameter. In hardware, it is pre-converted into a fixed-point representation and processed using an addition operation, enabling efficient implementation without additional arithmetic operations.

III. HARDWARE IMPLEMENTATION FOR SPIKING NEURAL NETWORK ACCELERATION

Fig. 4 illustrates the overall hardware architecture of the proposed SNN accelerator. The processing system of an ARM Cortex-A9 communicates with the programmable logic via the AXI4-Lite interface. The programmable logic comprises image block RAMs (BRAMs), 11-bit linear feedback shift registers (LFSRs), synchronous spike storage, a controller, weight BRAMs, and neuron modules.

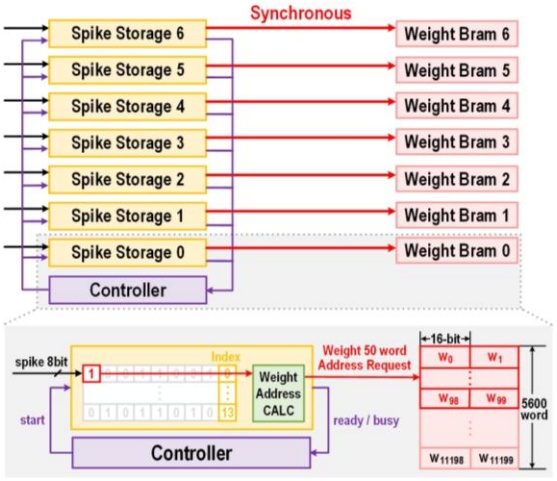


Fig. 5. Detailed implementation of synchronous spike storage.

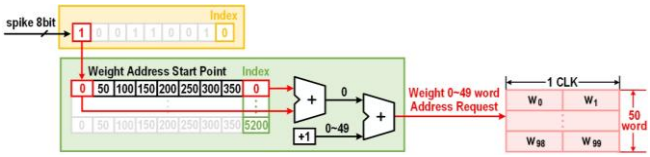


Fig. 6. Detailed weight address calculation.

A. 11-Bit Linear Feedback Shift Register

Poisson spike generation is implemented using an LFSR. The LFSR generates pseudo-random numbers each clock cycle using a simple shift-and-XOR structure, making it more resource-efficient than conventional random number generators. To improve throughput, the LFSR modules are implemented as seven parallel structures.

In each LFSR module, the eight 4-bit input image feature maps are compared with eight LFSR data, and an output spike is generated when the input data is greater than the LFSR data. Thus, an 8-bit spike is generated in each LFSR module.

To optimize the bit width of the LFSR, we compare the spike counts, as shown in Table I. The 10-bit LFSR generates too many spikes, whereas the 12-bit versions generate too few. Since the 11-bit LFSR offers results most similar to those obtained using the CPU, it is selected as the final configuration, providing an optimal balance between accuracy and efficiency.

B. Synchronous Spike Storage and Controller

Fig. 5 shows the structure of the proposed synchronous spike storage. Each spike storage stores fourteen 8-bit spike data generated by the LFSR module and records the spike position used to identify the weight address. In addition, each spike storage is connected to the weight BRAM, which consists of a 32-bit word structure, and two 16-bit weights are stored in a single word.

The controller controls and synchronizes the seven spike storages using three control signals: 1) Ready: represents that the spike generated by the LFSR has been written to the storage; 2) Busy: indicates that the weight address calculation (WAC) module is calculating the address, and the weight in the calculated address is being then transmitted to the excitatory neuron; and 3) Start: when at least one spike

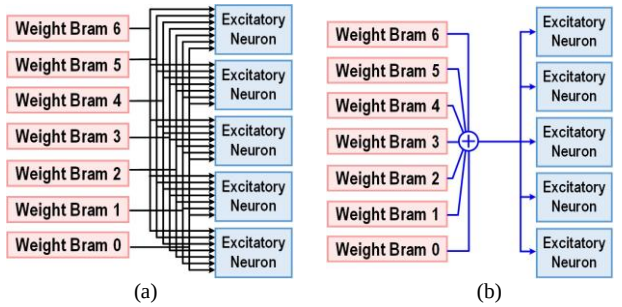


Fig. 7. (a) Asynchronous and (b) synchronous weight transmission architectures.

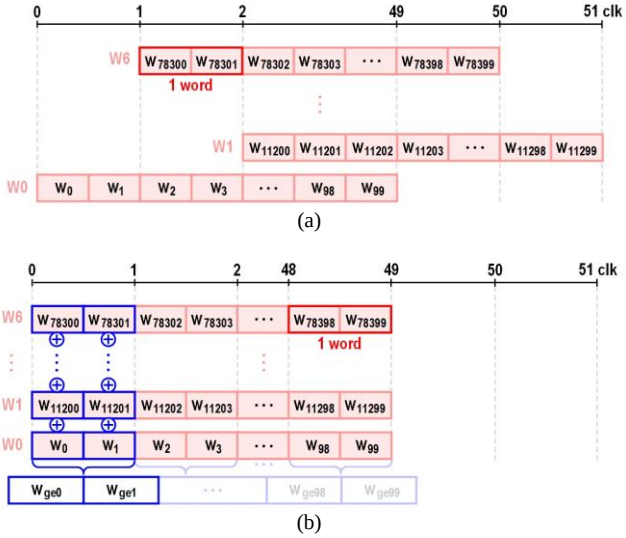


Fig. 8. Timing diagrams of the (a) asynchronous and (b) synchronous weight transmission architectures.

storage is in the ready state, this signal is broadcast, and the spike storage in the ready state enters the busy state.

Fig. 6 illustrates the detailed operation of the WAC. Based on the spike position and index, the weight address start point is determined. Depending on the spike position, the address increases in units of 50; for instance, the first position is 0, the second position is 50, and the third position is 100. The index in the WAC helps to calculate the exact starting address based on the index of the spike storage. The WAC operation is performed using two adders. First, the weight address start point and index are added at the first adder to calculate the start address of the spike. The second adder then adds 0 to 49 to this start address, generating an address that increases by one each cycle.

C. Synchronous Weight Transmission Architecture

Fig. 7(a) shows the weight transmission architecture without the proposed controller, where each spike storage asynchronously requests weights. In this case, the data paths from the weight BRAMs to the excitatory neurons are separated into seven channels, which complicates the system design and management. In contrast, in the proposed architecture, the weight request is synchronized, and the weight is accumulated as a single value for each clock cycle and transmitted to the excitatory neuron, as shown in Fig. 7(b).

Fig. 8 presents timing diagrams of the asynchronous and synchronous weight transmission architectures. In the asynchronous architecture, the weight request occurs in different clock cycles (Fig. 8(a)), whereas all spike storage performs weight requests in the same clock cycle in the proposed architecture (Fig. 8(b)). A blue box represents the sum of weights in each aligned clock cycle. The accumulated values are then sequentially transferred to the excitatory neurons.

TABLE II. Performance Comparison with Prior Works

	[5]	[6]	[8]	**This Work**
Clock (MHz)	200	200	100	**100**
Data Format	Custom FP (26-bit)	16-bit Fixed	16-bit Floating	**16-bit Fixed**
Neuron Model	LIF	LIF	LIF	**LIF**
FPGA Platform	XC7Z7020	XC7Z045	Virtex7	**XC7Z7020**
# of Neuron	1,294	16,384	1,094	**984**
Dataset	MNIST	MNIST	MNIST	**MNIST**
Latency (ms/image)	1,670	6.21	3.15	**1.41**
Speedup	20.5×	N/R	N/R	**103.8×**
Accuracy Loss (%)	0.32	1.42	N/R	**0.3**

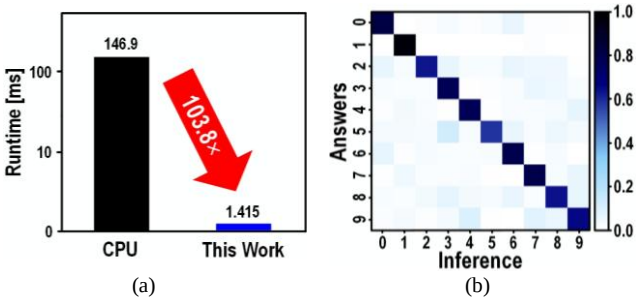


Fig. 9. Measurement results: (a) runtime comparison of a single image and (b) confusion matrix of inference results.

Consequently, the proposed synchronous architecture can improve the weight access speed and reduce the number of LUTs by 14.97% (6,404) and flip-flops by 37.55% (11,841) compared to the asynchronous structure.

IV. EXPERIMENTAL RESULTS

To verify the proposed design, the MNIST dataset was used, and 50,000 training images and 10,000 test images were utilized. The software baseline for performance comparison was obtained by executing in the C language on an Intel® Core™ i3-1115G4 processor (3.00 GHz, 2 cores) with a 8-GB RAM, using a single-threaded implementation. The proposed hardware was implemented on an Xilinx Zynq-7000 SoC Z7-20 board operating at 100 MHz.

Table II summarizes the FPGA resource utilization of the proposed design based on the Xilinx Zynq-7000 SoC Z7-20 board. Here, “optimization” refers to the hardware optimization procedures described in Section II. By optimizing synaptic transmission delay, LUTRAM utilization was reduced from 1.72% to 0%, and flip-flop utilization decreased from 24.18% to 18.51%, since eliminating these elements simplified the signal path and reduced the need for additional storage. Before optimization, LUT utilization was 402.76%; thus, the SNN model could not be implemented in our FPGA hardware. By reformulating the neuron equations with power-of-two denominators, LUT utilization was significantly reduced to 68.38%, as divisions by arbitrary constants no longer required LUT-based operations. These optimizations enabled the proposed design

TABLE III. FPGA Resource Utilization

	Baseline (w/o Optimization)	**This Work (w/ Optimization)**
LUT	214,267 (402.76%)	36,375 (68.37%)
LUTRAM	300 (1.72%)	0 (0%)
BRAM	71 (50.71%)	71 (50.71%)
FF	25,723 (24.18%)	19,697 (18.51%)
DSP	226 (102.73%)	220 (100.00%)

to save substantial hardware resources while preserving the essential computational behavior of the SNN.

On the MNIST test set consisting of 10,000 images, the baseline required 1,469 s, whereas the FPGA completed the task in only 14.15 s. In other words, the proposed design achieved an inference latency of 1.415 ms per image, corresponding to a 103.8× speedup compared to the baseline, as shown in Fig. 9(a). Fig. 9(b) presents the confusion matrix to validate the inference results; the proposed design achieved an inference accuracy of 76.2%.

Table III presents a performance comparison with prior studies. Our design achieves lower inference latency than prior works, even with a low number of neurons, and its inference speed is significantly higher compared to the baseline.

V. CONCLUSION

A hardware-efficient FPGA-based SNN accelerator is presented. We adopt a 16-bit fixed-point precision that reduces hardware cost while preserving spike dynamics by applying 10× voltage scaling. Furthermore, synaptic transmission delay optimization and neuron equation reconstruction using the powers of two denominators reduce LUT usage by approximately 83%, eliminate LUTRAM, and reduce flip-flop usage by approximately 23%. In addition, the proposed synchronous spike storage and the controller module simplify the data paths. Compared to the asynchronous design, it reduces LUT utilization by 14.97% and flip-flop utilization by 37.55%. Experimental results show that the proposed design achieves a runtime of 1.415 ms/image, which is 103.8 times faster than the CPU baseline of 146.9 ms/image, with only a 0.3% decrease in accuracy. Therefore, the proposed optimization strategy provides applicable hardware solutions for neuromorphic computing by effectively mitigating inference delays and resource inefficiencies in SNNs.

ACKNOWLEDGMENT

This work was supported in part by the excellent researcher support project of Kwangwoon University in 2025 and in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (Nos. RS-2024-00334247 and RS-2024-00414230). The EDA tools were supported by the IC Design Education Center (IDEC), Korea.

REFERENCES

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

[2] M. Heidarpur, A. Ahmadi, M. Ahmadi, and M. R. Azghadi, “CORDIC-SNN: On-FPGA STDP learning with Izhikevich neurons,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 7, pp. 2651–2664, Jul. 2019.

[3] Q. Meng, M. Xiao, S. Yan, Y. Wang, Z. Lin, and Z.-Q. Luo, “Training high-performance low-latency spiking neural networks by differentiation on spike representation,” *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 12434–12443, 2022.

[4] R. Yin, Y. Kim, D. Wu and P. Panda, “LoAS: Fully temporal-parallel dataflow for dual-sparse spiking neural networks,” *Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, pp. 1107–1121, 2024.

[5] Y. Nevarez, D. Rotermund, K. R. Pawelzik, and A. García-Ortiz, “Accelerating spike-by-spike neural networks on FPGA with hybrid custom floating-point and logarithmic dot-product approximation,” *IEEE Access*, vol. 9, pp. 80603–80620, May 2021.

[6] J. Han, Z. Li, W. Zheng, and Y. Zhang, “Hardware implementation of spiking neural networks on FPGA,” *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 479–486, Aug. 2020.

[7] D. Larionov, “MNIST spiking neural network,” Kaggle, [Online]. Available: https://www.kaggle.com/code/dlarionov/mnist-spiking-neural-network.

[8] S. Li, Z. Zhang, R. Mao, J. Xiao, L. Chang, and J. Zhou, “A fast and energy-efficient SNN processor with adaptive clock/event-driven computation scheme and online learning,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 4, pp. 1543–1552, Apr. 2021.

Jeong-Eun Ko is currently pursuing her B.S. degree in the Department of Electronics and Communications Engineering at Kwangwoon University, Seoul, Korea.

Woo-Vin Choi is currently pursuing her B.S. degree in the Department of Electronics and Communications Engineering at Kwangwoon University, Seoul, Korea.

Min-Seok Kim is currently pursuing her B.S. degree in the Department of Electronics and Communications Engineering at Kwangwoon University, Seoul, Korea.

Joo-Hyung Chae received his B.S. and Ph.D. degrees in Electrical Engineering from Seoul National University, Seoul, South Korea, in 2012 and 2019, respectively.

In 2013, he joined SK hynix, Icheon, South Korea, as an intern at the Department of LPDDR Memory Design. From 2019 to 2021, he was with SK hynix, Icheon, South Korea, where his work focused on GDDR memory design. In 2021, he joined Kwangwoon University, Seoul, South Korea, where he is currently an Associate Professor of Electronics and Communications Engineering.

Dr. Chae received the Doyeon Academic Paper Award from the Inter-University Semiconductor Center (ISRC), Seoul National University, in 2020. He has been serving as a TPC member for the IEEE Asian Solid-State Circuits Conference (A-SSCC). He has also served as an OC member for the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), International Conference on Electronics, Information, and Communication (ICEIC), International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), and International Conference on Consumer Electronics (ICCE) Asia.

His research interests include the design of high-speed and low-power I/O circuits, clocking circuits, memory interfaces, and mixed-signal in-memory computing.