Chain vol. 1, no. 2, pp. 167–176, 2024
DOI: 10.23919/CHAIN.2024.000003
ISSN 2097-3470
CN 10-1915/TB
Research Article

Cost-effective and high-speed implementation of brain-like spiking neural network with encoding architectures on FPGA

Zhen Cao✉, Ziyi Zhang, Qi Sun, Biao Hou✉, Licheng Jiao, and Yintang Yang

School of Artificial Intelligence and School of Microelectronics, Xidian University, Shaanxi 710071, China

Received: 25 November 2023
Revised: 15 January 2024
Accepted: 19 January 2024
Online: 30 February 2024

KEYWORDS
SNN accelerator,
FPGA,
spiking encoding,
low-power,
neuromorphic computing

ABSTRACT

The spiking neural network (SNN) is the third generation of neural networks, valued for its biological interpretability, hardware compatibility, and low power consumption. While field programmable gate array (FPGA), based SNN accelerators show promise with their intelligent design and low power consumption, existing models lack a crucial spiking encoding module. This leads to increased pre-processing time on the CPU. To address these issues, this paper proposes an FPGA-based SNN accelerator with a spiking encoding module. This innovation reduces CPU pre-processing time by threefold without adding FPGA processing time. Achieving 92.62% accuracy on the MNIST dataset using LIF spiking neurons at 0.675 W, the accelerator outperforms CPU and GPU, being 14× faster and 2.5× faster, respectively, and more power-efficient with minimal resource usage. Given its pixel input, it can be seamlessly embedded in real-time classification systems for efficient classification and detection.

1 Introduction

Hailed as the third generation of neural networks, spiking neural network (SNN) is a neural network model that mimics the biological nervous system and simulates the way information is transmitted between neurons in the brain through spikes [1], as shown in Fig. 1. During the latter decades of the 20th century, specifically in the 1980s and 1990s, a burgeoning interest in the theoretical underpinnings and learning algorithms of SNNs emerged among researchers. A pivotal milestone was reached in 2003 when Izhikevich introduced an innovative spiking neuron model [2] that significantly streamlined computations while

Address correspondence to Zhen Cao, caozhen@xidian.edu.cn; Biao Hou, avcodec@163.com

有科出版 YOUKE PUBLISHING EXCLUSIVELY AVAILABLE VIA IEEE Xplore DIGITAL LIBRARY

maintaining remarkable fidelity in simulating a wide array of neuronal behaviors. This groundbreaking proposal laid the groundwork for more practical and effective implementations of SNNs.

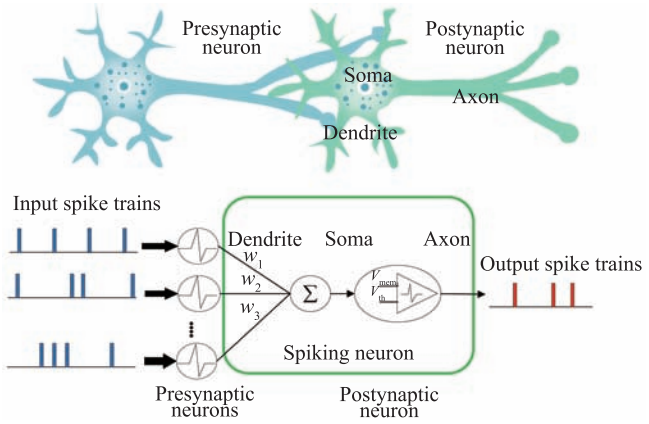

Figure 1 Spiking neural network model.

SNN is celebrated for its advanced capability to simulate biological neural processes and its inherent compatibility with event-driven hardware architectures, which enables a form of computation that parallels the human brain's efficiency in terms of both low power consumption and high-performance processing. By virtue of these attributes, SNNs are uniquely positioned to emulate the brain's energy-efficient information processing mechanisms, thereby fostering an innovative paradigm shift towards neuromorphic computing systems. It represents a crucial area of study in brain-like computing systems [3, 4]. In the 1990s and 2000s, researchers developed specialized neuromorphic chips like SpiNNaker [5], TrueNorth [6], Darwin1,2 [7, 8], Loihi1,2 [9, 10], Tianjic [11], TianjicX [12] to simulate SNN models. These chips, equipped with parallel computing capabilities and low power consumption, enable practical applications of SNN.

Application-Specific integrated circuit (ASICs) offer advantages like smaller size, lower power consumption, and higher integration, but they come with longer design cycles, higher costs, and no post-completion modifications. In contrast, the semi-custom design method shortens development cycles, significantly reduces costs, and allows for convenient design modifications, making it ideal for mass production. Consequently, field programmable gate arrays (FPGAs) are increasingly chosen by researchers for SNN acceleration. Leveraging FPGA flexibility and parallel computing power enables the creation of high-performance, real-time processing, and low-power consumption systems [13]. Notably, Neil et al. introduced Minitaur, an FPGA-based, low-power, high-performance, event-driven SNN accelerator [14]. Zhang et al. proposed an asynchronous SNN accelerator with 1024 neurons and 1 million synapses [15]. Other studies include Huang et al.'s low-power SNN design for radioisotope identification [16], Khodamoradi et al.'s streaming SNN (S2N2) architecture on FPGA [17], and Gao et al.'s application of the MLIF model to SNN with basal dendrites, apical dendrites, and somas [18].

However, the FPGA-based SNN accelerators discussed earlier require spike trains as input, leading to CPU-intensive pre-processing such as spiking encoding for image classification. This CPU-centric approach not only consumes more time and power but also contrasts with the efficiency achieved by encoding directly on the FPGA. To overcome this challenge, we proposes an inventive FPGA-based SNN accelerator implementation with integrated spiking encoding in this paper. Significantly, this method drastically reduces CPU pre-processing time by threefold without adding extra processing time on the FPGA. Distinguishing itself from prior methodologies, our approach excels in judicious resource utilization, diminished power consumption, and expedited image inference. Notably, the pixel-based input structure of the system seamlessly integrates into real-time classification systems, promising efficient and swift classification and detection.

2 Materials and methods

2.1 Spiking encoding

In the SNN, information propagates through spikes, and spiking encoding technology plays a pivotal role in translating information into spike trains. This key technology encompasses common encoding techniques, categorized into rate encoding and temporal encoding [19]. This paper employs a widely used method for generating spike trains known as Poisson distributed spike encoding [20, 21]. The Poisson encoding method falls

under the category of rate encoding and transforms input data into spike trains with an issuance distribution adhering to the Poisson process. Specifically,

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda} \quad (1)$$

where, P denotes the probability of event X occurring k times, and the parameter λ is the average number of occurrences of random events per unit time or unit area.

We performed Poisson encoding on the image over 20 timestep, and Fig. 2(a) represents the spike trains at t = 1, 5, 10, 15, 20, showing that the Poisson encoding is time-independent. Superimposing the spike trains in 20 timesteps, Fig. 2(b) shows the cumulative effect at t = 1, 5, 10, 15, 20. It can be seen that the original image can be efficiently reconstructed through the repeated overlay of Poisson encoding.

In practical implementation, Poisson spike trains are commonly generated by comparing scaled pixels to random numbers. The FPGA's Poisson encoding module utilizes a pseudo-random number generator (PRNG) and a comparer. In Fig. 3, the PRNG is realized through a 16-bit linear shift register employing three XOR gates.

FPGA systems operating with spike trains as input currently necessitate CPU-driven encoding processing and the FPGA input is serialized. In Fig. 4, the introduction of an encoding module to the FPGA system maintains the same input length as the prior system. However, with the addition of parallel encoding processing and neuron calculations, there is minimal extra FPGA processing time. Consequently, incorporating an encoding module into the FPGA system reduces the encoding time in pre-processing without escalating

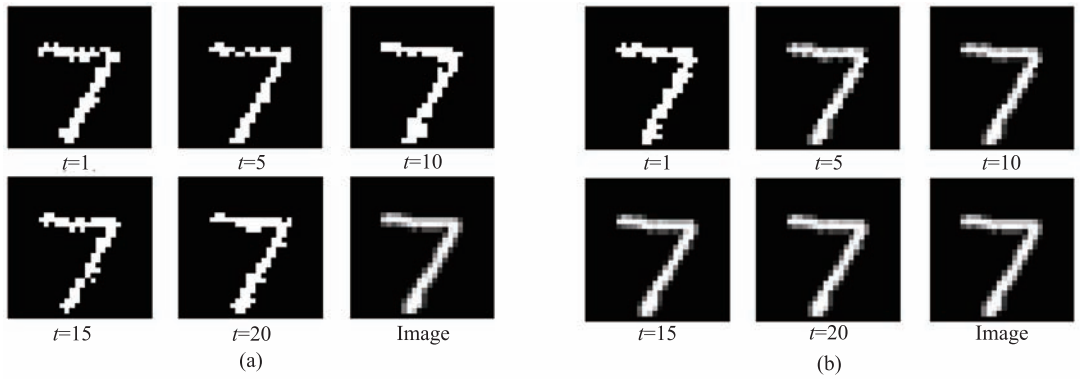

Figure 2 Poisson encoding: (a) spike train encoded image with different timestep; (b) cumulative effect superimposed encoding image in 20 timestep.

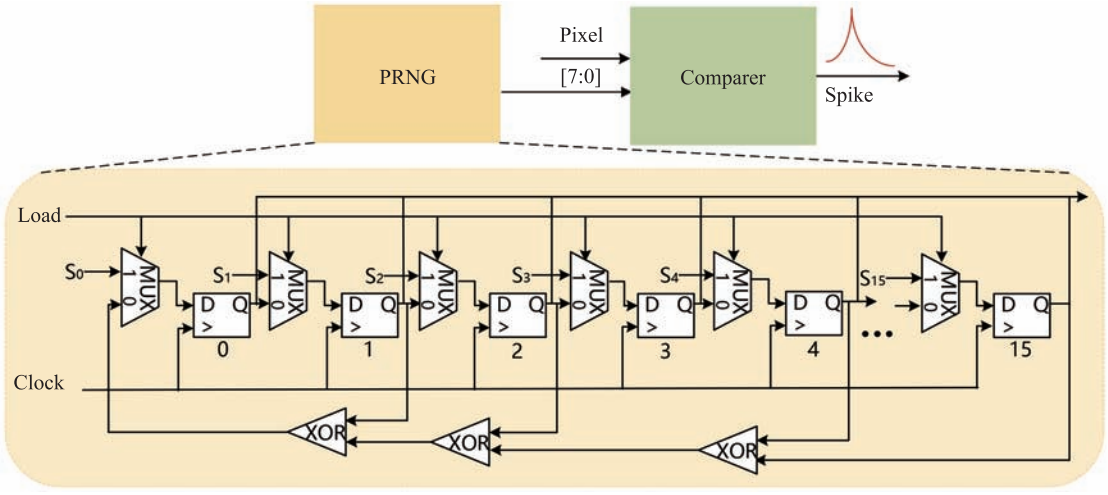

Figure 3 Circuit diagram for Poisson encoding.

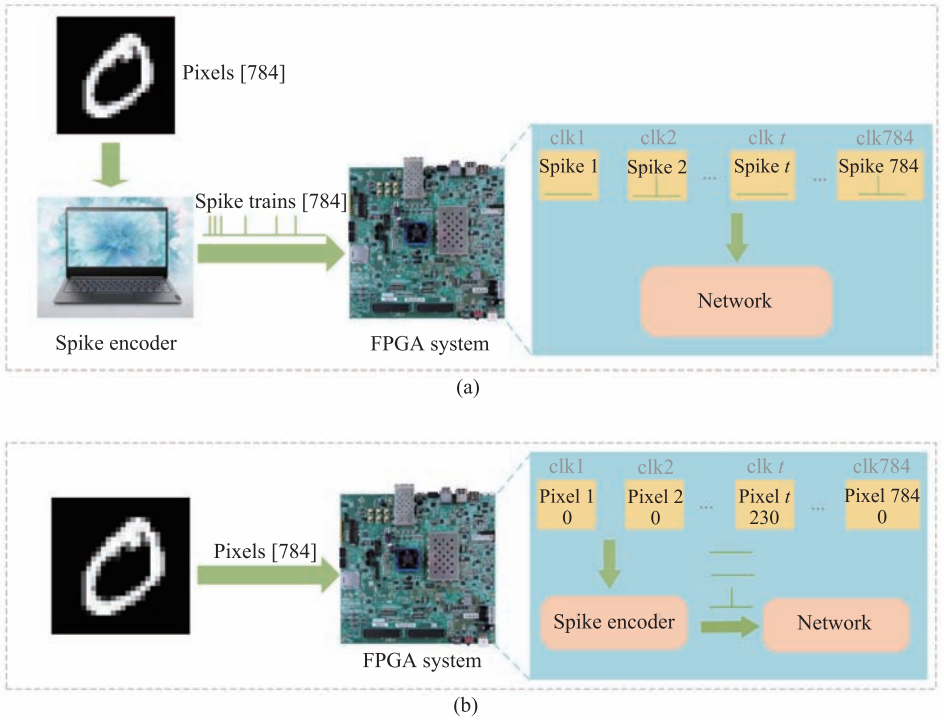

Figure 4 The distinction between CPU-based encoding and FPGA-based encoding is depicted: (a) encoding on the other device; (b) encoding on the FPGA.

FPGA processing time. Moreover, in the second system, pixels could originate from a sensor like a camera, allowing direct integration into real-time systems without the need for pre-processing.

2.2 Spiking neuron

The primary distinction between SNN and artificial neural networks lies in their neurons. Spiking neurons closely mimic human neuron connections. Neurophysiologists have devised various models to capture the dynamic electrical potential of working neurons, serving as the fundamental units of SNN and determining the network's core dynamics. The realization of SNN is intricately tied to the evolution of spiking neurons, with models such as the Hodgkin-Huxley model [22] and the Izhikevich model [2] closely resembling biological neurons. However, their intricate hardware circuitry renders them less suitable for practical hardware deployment. In contrast, the IF model [23, 24] and the integrate-and-fire (LIF) model [25, 26], while not as intricate as their predecessors, offer simplicity in calculations and are more amenable to hardware deployment. This paper employs the LIF neuron model, described by the following equation as

$$\tau_{\mathrm{m}} \frac{\mathrm{d}V(t)}{\mathrm{d}t} = -\left(V(t) - V_{\mathrm{res}}\right) + RI(t) \tag{2}$$

where τ_{m} epresents the delay constant, $V(t)$ signifies the instantaneous value of the neuronal membrane potential at time t, and V_{res} epresents the resting value, the membrane potential when there is no input.

As Eq. (2) is in the differential form, and FPGA processes digital signals exclusively, it is discretized and organized Eq. (2) as follows:

$$V(t+1) = V(t) + \left(RI(t) - \left(V(t) - V_{\mathrm{res}}\right)\right) / \tau_{\mathrm{m}} \tag{3}$$

where $I(t)$ represents the input spikes received by the neuron, correlating to the synapses in the neural network, Eq. (3) can be modified to

$$V(t+1) = V(t) + \left(\sum\nolimits_i W_i S_i - \left(V(t) - V_{\mathrm{res}}\right)\right) / \tau_{\mathrm{m}} \tag{4}$$

where i is the ith synapse connected to the neuron, W_i is the weight corresponding to the ith synapse, and S_i indicates whether the current timestep of the ith synapse has an input spike, which is 1 or 0 otherwise. In this work, the τ_{m} is 2 and the V_{res} is 0.

The circuit diagram for the computation and update of the LIF neuron is illustrated in Fig. 5.

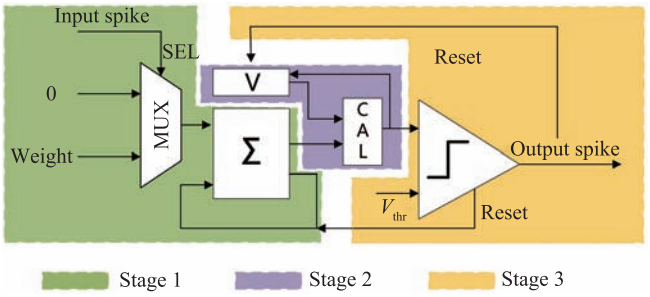

Figure 5 LIF neuron circuit diagram.

The neuron initially aggregates the weights corresponding to spikes at the same timestep t. Subsequently the neuron calculates the current membrane potential by incorporating the previous moment's membrane potential and updates the potential. Following the computation, the neuron compares the membrane potential with the threshold. If the membrane potential exceeds the threshold, the neuron generates a spike and resets the membrane potential to the reset potential.

2.3 Network

In this paper, the MNIST dataset is utilized for testing, comprising 60,000 training sets and 10,000 test sets. The architecture of a 784-10 fully connected layer SNN is depicted in Fig. 6. Spike trains serve as input to the network, and communication between each layer employs the address event representation (AER) protocol [27]. Following computation, each neuron in the output layer determines whether it generates a spike in each timestep. By tallying the spikes, the category associated with the neuron producing the most spikes after T timestep is considered as the classification result. Network training is conducted using the PyTorch and SpikingJelly [28] frameworks. Optimal weights are obtained and quantized into 16-bit fixed-point numbers, with the specific quantization method being $W_i=w_i\cdot 2^{\beta}$. Where β is 15 and the 16th bit is the sign bit.

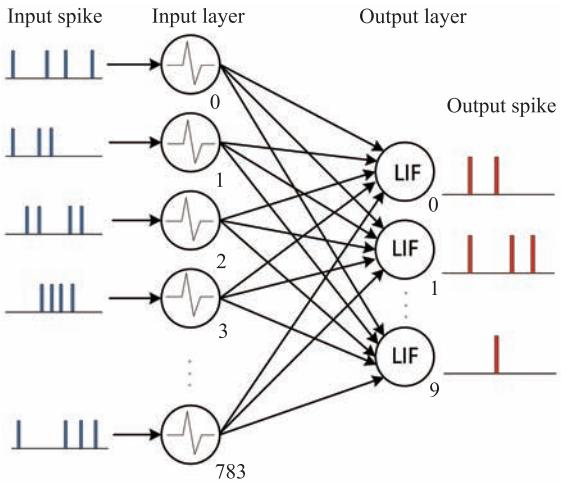

Figure 6 Spiking fully connected network architecture.

2.4 System architecture

The system architecture is shown in Fig. 7. Initially, the weight ROM initializes the weights of each neuron, and data is allowed to entry only after completion of this initialization. Subsequently, as data is entered, the spike_encoder module generates spike trains by encoding pixel value using the Poisson encoding method. The network calculates the spike trains at each timestep,

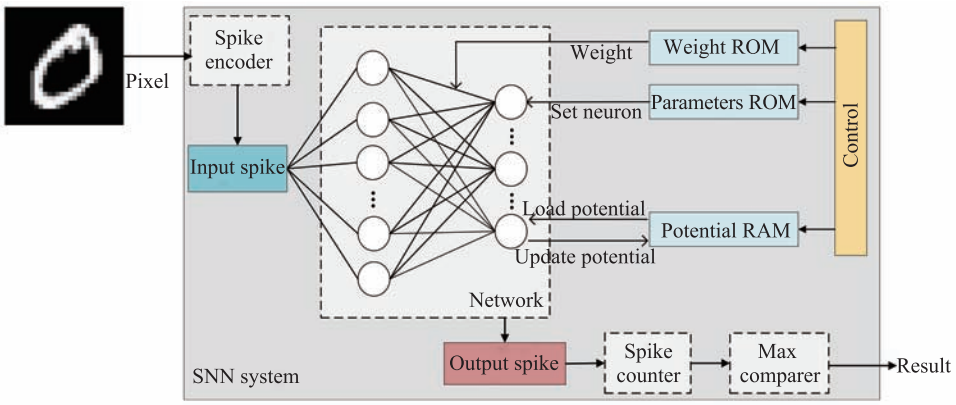

Figure 7 System architecture.

producing corresponding output spikes. Concurrently, the spike_counter module tallies the spikes generated by the respective neurons. After T timestep, the max_comparer module identifies the neuron generating the most spikes as the output. Prior to the arrival of the next set of data, the membrane potential and spike count of a neuron are reset to zero. This iterative procedure persists until the exhaustive evaluation of the entire dataset has been accomplished.

3 Results and discussion

All tests in this section are conducted on the MNIST test dataset, and they are performed at a clock frequency of 100 MHz on the FPGA.

3.1 Test for Poisson encoding

We evaluate the network performance on CPU, GPU, ASIC, and FPGA, and the results are presented in Table 1.

Analyzing the data in Table 1, it is evident that the FPGA system outperforms the CPU implementation by 14 times and the GPU implementation by 2.5 times, with a negligible 0.07% reduction in accuracy. Notably, the power consumption of this work, at 0.675 W, is significantly lower than that of the CPU and GPU implementations. Moreover, in comparison to another FPGA-based accelerator named Minitaur, this work exhibits superior speed and lower power consumption. Upon examining the data presented in Table 2, it becomes evident that regardless of the varying timestep, the FPGA accelerator consistently demonstrates a speed improvement of approximately 14 times over the CPU.

To assess the performance of the PRNG in our design, we conduct a comparison with the {$random} function in Verilog. B. Fang et al. [29] implemented the fixed uniform spike generation algorithm in the spiking convolutional neural network (SCNN) on FPGA, and in another work [30], H. Fang achieved an SCNN architecture with time encoding also based on FPGA. Our design is then compared with these two works.

Analyzing the data in Table 3, it is observed that employing the PRNG as the Poisson encoding module results in a 0.06% higher accuracy and a 0.1% lower power consumption compared to using the {$random} function. While our design, focused on image classification using SCNN, exhibits lower accuracy compared to the two referenced works, it is noteworthy that our system is three times faster and consumes 6–7 times less power than the other two designs. In terms of speed and power efficiency, our design stands out.

A significant limitation of many FPGA-based SNN accelerators is the absence of a spiking encoding module, necessitating encoding processing during pre-processing, typically handled by the CPU. Integrating

Table 1 Comparations with other platforms.

Platform	Type	Time (s)	Accuracy (%)	Power (W)
This work	FPGA	1.5693	92.62	0.675
i7-8565U	CPU	22.3933	92.69	56.1
RTX 3060 (batch_size=64)	GPU	4.0458	92.69	58.2
TrueNorth [6]	ASIC	2.1867	93.00	0.18
Minitaur [14]	FPGA	10.8755	92.00	1.5

Table 2 The accuracy and using time of FPGA system with different timesteps.

Timestep	Accuracy on PC (%)	Time on PC (s)	Accuracy on FPGA (%)	Time on FPGA (s)
100	93.06	102	93.00	7.6354
20	92.69	22.3933	92.62	1.5695
10	91.85	12.4128	91.85	0.7861
5	90.77	7.2795	90.82	0.3941

Table 3 Test for Poisson encoding module.

	Accuracy (%)	Time (s)	Power (W)
Our design	92.62	1.5693	0.675
{$random}	92.56	1.5693	0.676
B. Fang et al. [29]	97.87	4.645	4.775
H. Fang et al. [30]	99.2	4.708 (125 MHz)	4.5

the encoding directly into the FPGA can mitigate this bottleneck. The pre-processing tasks encompass image reading, image-to-spike train encoding, and serial file read/write operations. Upon the introduction of the encoding module to the FPGA system, the CPU's pre-processing time is notably reduced by more than threefold, decreasing from 405 s to 120 s.

3.2 Test for Poisson encoding with the same spike

Given that Poisson encoding represents a form of rate encoding, the spike trains generated at each timestep are time-independent. Therefore, it is plausible to utilize the same spike train for each timestep. Upon the arrival of the initial batch of images, the system encodes them into spike trains and stores the corresponding addresses in RAM. Subsequently, these addresses are read again in the subsequent timesteps, optimizing efficiency in spike train handling.

Analyzing the data in Table 4, it is evident that employing the same spike train for each timestep leads to a 1.69% reduction in accuracy. However, it significantly decreases the testing time on the FPGA by five times. This approach might be a viable consideration for applications prioritizing real-time processing over absolute accuracy, such as video surveillance.

3.3 Resource usage

The model is deployed on the Xilinx ZCU 102, and we conduct a comparison with two other implementations sharing a similar network architecture without the spiking encoding module. The resource usage comparison is presented in Table 5.

In Table 5, it is evident that the SNN system designed in this paper utilizes a minimal number of resources, showcasing efficient resource utilization. In comparison with the other two preceding works, our SNN system demonstrates reduced resource usage with 4342 look-up tables (LUTs), 620 flip-flops (FFs), and 5 block RAMs (BRAMs). Additionally, the total power dissipation is 0.675 W, reflecting lower power consumption. Moreover, our system is 2–3 times faster than the other two accelerators when inferring 10,000 images.

3.4 Embedded application

Given that the system processes pixel inputs, it can be seamlessly integrated into real-time classification systems for effective classification and detection. In

Table 4 The impact of using the same Poisson encoding on accuracy and processing time.

Poisson encoding	Accuracy on CPU (%)	Time on CPU (s)	Accuracy on FPGA (%)	Time on FPGA (s)
Different	92.69	3.2214	92.62	1.5693
Same	90.19	1.5000	90.79	0.2916

Table 5 System resource usage comparations.

	LUT	Register (FF)	BRAM	Time (s) (10,000 images)	Power (W)
This work	4342	620	5	1.5693	0.675
Shen et al. [31]	20,738	6549	100	3.522	1.584
Gupta et al. [32]	56,230	23,238	16	5	—

this section, we specifically apply it to recognize digital numbers, as illustrated in Fig. 8.

Figure 8 Digital numbers recognition system.

The system workflow is illustrated in Fig. 9. Initially, the camera captures the image and transfers RGB pixels to the ZCU 102. Subsequently, the ZCU 102 resizes the image to 28×28 and converts the RGB pixel to a grayscale value. Finally, the SNN system reads the grayscale value, predicts it, and displays the result using LEDs

4 Conclusion

In this paper, we introduce a novel implementation approach for an FPGA-based SNN accelerator with Poisson encoding. Leveraging LIF neurons, our design attains a classification accuracy of 92.62% for the MNIST dataset, concurrently reducing CPU pre-processing time by threefold and being 14× faster than CPU, 2.5× faster than GPU. The system operates at a power consumption of 0.675 W, showcasing efficiency in resource utilization compared to prior designs. Given its pixel-based input, this accelerator seamlessly integrates into real-time classification systems, facilitating efficient classification and detection.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Nos. 62104176 and 62171347), in part by the Fundamental Research Funds for the Central Universities (No. XJSJ23083), in part by the Proof of Concept Foundation of Xidian University Hangzhou Institute of Technology (No. GNYZ2023XJ0409-1), in part by the Shaanxi Higher Education Teaching Reform Research Project (No. 23ZZ014), and in part by the China Postdoctoral Science Foundation (No. 2019M663637).

References

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

[2] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks.* vol. 14, no. 6, pp. 1569–1572, 2003.

[3] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[4] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, "A review of learning in biologically plausible spiking neural networks," *Neural Networks*, vol. 122, pp. 253–272, 2020.

[5] S. B. Furber, F. Galluppi, S. Temple, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. J.

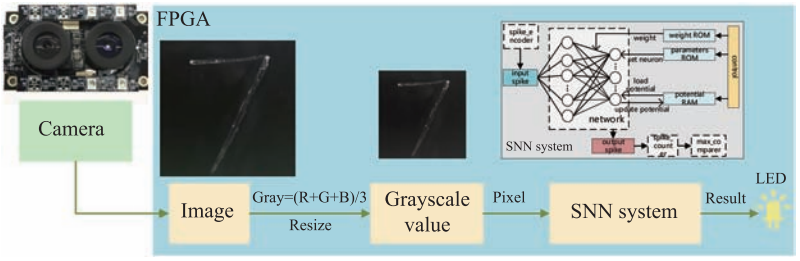

Figure 9 Digital numbers recognition system workflow.

Nam, et al., "The spinnaker project," *Proceedings of the IEEE,* vol. 102, no. 5, pp. 652–665, 2014.

[6] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-lcaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, and G. -J. Nam, "Truenorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. vol. 34, no. 10, pp. 1537–1557, 2015.

[7] J. Shen, D. Ma, Z. Gu, M. Zhang, and G. Pan, "Darwin: A neuromorphic hardware co-processor based on spiking neural networks," *Science China - Information Sciences*, vol. 59, no. 2, pp. 1–5, 2016.

[8] S. Deng, P. Lv, O. Jin, S. Dustdar, Y. Li, D. Ma, Z. Wu, and G. Pan, "Darwin-S: A reference software architecture for brain-inspired computers," *Computer*, vol. 55, no. 5, pp. 51–63, 2022.

[9] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*. vol. 38, no. 1, pp. 82–99, 2018.

[10] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. B. Shrestha, F. T. Sommer, and M. Davies, "Efficient neuromorphic signal processing with Loihi 2," in *Proceedings of the 2021 IEEE Workshop on Signal Processing Systems (SiPS*), Coimbra, Portugal, 2021, pp. 254–259.

[11] J. Pei, L. Deng, S. Song, M. G. Zhao, Y. H. Zhang, S. Wu, G. R. Wang, Z. Zou, Z. Z. Wu, W. He, et al. "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[12] S. Ma, J. Pei, W. Zhang, G. Wang, D. Feng, F. Yu, C. Song, H. Qu, C. Ma, and M. Lu, "Neuromorphic computing chip with spatiotemporal elasticity for multi-intelligent-tasking robots," *Science Robotics*, vol. 7, no. 67, p. 2948, 2022.

[13] L. P. Maguire, T. M. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin, "Challenges for large-scale implementations of spiking neural networks on FPGAs," *Neurocomputing*, vol. 71, no. 1–3, pp. 13–29, 2007.

[14] D. Neil and S. C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2621–2628, 2014.

[15] J. Zhang, H. Wu, J. Wei, S. J. Wei, and H. Chen, "An asynchronous reconfigurable SNN accelerator with event-driven time step update," in *Proceedings of the 2019 IEEE Asian Solid-State Circuits Conference (A-SSCC*), Macau, China, 2019, pp. 213–216.

[16] X. Huang, E. Jones, S. Zhang, S. Y. Xie, S. Furber, Y. Goulermas, E. Marsden, I. Baistow, S. Mitra, and A. Hamilton, "Spiking Neural network based low-power radioisotope identification using FPGA," in *Proceedings of the 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS*)", Glasgow, Scotland, 2020, pp. 1–4.

[17] A. Khodamoradi, K. Denolf, and R. Kastner. "S2N2: A FPGA accelerator for streaming spiking neural networks," in *Proceedings of the 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. Virtual Event, USA, 2021, pp. 194–205.

[18] T. Gao, B. Deng, J. Wang, and G. Yi, "Highly efficient neuromorphic learning system of spiking neural network with multi-compartment leaky integrate-and-fire neurons," *Frontiers in Neuroscience*, vol. 16, p. 929644, 2022.

[19] Y. F. Hu, G. Q. Li, Y. J. Wu, and L. Deng, "Spiking neural networks: A survey on recent advances and new directions," *Control and Decision*, vol. 36, no. 1, pp. 1–26, 2021.

[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, 2015.

[21] P. U. Diehl, D. Neil, J. Binas, M. Cook, and S. C. Liu "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN*), Killarney, Ireland, 2015, pp. 1–8.

[22] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology,* vol. 117, no. 4, pp. 500, 1952.

[23] P. Dayanand and L. F. Abbott, "Theoretical neuroscience: Computational and mathematical modeling of neural systems," *Journal of Cognitive Neuroscience*, vol. 15, no. 1, pp. 154–155, 2003.

[24] P. Dayan and L. Abbott, *Computational Neuroscience: The-*

oretical Neuroscience: Computational and Mathematical Modeling of Neural Systems, Cambridge, UK: MIT Press, 2001, pp. 162–166.

[25] X. Ju, B. Fand, and R. Yan, "An FPGA implementation of deep spiking neural networks for low-power and fast classification," *Neural Computation*, vol. 32, no. 1, pp. 182–204, 2020.

[26] X. Huang, E. Jones, S. Zhang, S. Y. Xie, S. Furber, Y. Goulermas, and E. Marsden, "An FPGA implementation of convolutional spiking neural networks for radioisotope identification," in *Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS*), Daegu, South Korea, 2021, pp. 1–5.

[27] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, 2000.

[28] W. Fang, Y. Chen, J. Ding, D. Chen, Z. Yu, H. Zhou, T. Masquelier, Y. Tian, "Spikingjelly," Accessed: May 1, 2024 [Online]. Available: https://github.com/fangwei123456/spikingjelly, 2020.

[29] B. Fang, Y. Zhang, R. Yan, and H. J. Tang, "Spike trains encoding optimization for spiking neural networks implementation in FPGA," in *Proceedings of the 2020 12th International Conference on Advanced Computational Intelligence (ICACI)*, Dali, China, 2020, pp. 412–48.

[30] H. Fang, Z. Mei, A. Shrestha, Z. Y. Zhao, Y. L. Li, and Q. R. Qiu, "Encoding, model, and architecture: Systematic optimization for spiking neural network in FPGAs," in *Proceedings of the 39th International Conference on Computer-Aided Design. Virtual Event*, USA, 2020, pp. 1–9.

[31] Y. J. Shen J. C. Shen, J. Ye, Q. Ma, "Design of spiking neural network Accelerator based on FPGA," *Electronic Science and Technology*. vol. 30, no. 10, pp. 89–92, 2017.

[32] S. Gupta, A. Vyas, and G. Trivedi, "FPGA implementation of simplified Spiking Neural Network," in *Proceedings of the 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Glasgow, Scotland, 2020, pp. 1–4.