

# Study of FPGA – Based Spiking Neural Networks: Classification Implementation

Mehmet Sinan Çınaroğlu<sup>1\*</sup> and Erol Seke<sup>2</sup>

<sup>1</sup>Graduate School of Natural and Applied Sciences, Eskişehir Osmangazi University, Eskişehir, Türkiye (sinanmcinaroglu@gmail.com)

<sup>2</sup>Electrical and Electronics Engineering, Eskişehir Osmangazi University, Eskişehir, Türkiye (eseke@ogu.edu.tr)

\*corresponding author

**Abstract** – With the help of ability to imitate the biological nervous system, Spiking Neural Networks (SNNs) architectures have become popular recently. In order to benefit at the maximum level from the low power consumption, event-based structure and parallelization features of SNN, it is more potent method to implement SNN with hardware-based applications. On the other hand, Field Programmable Gate Arrays (FPGAs) which are frequently preferred in the literature for hardware solutions, are accepted as platforms with low power requirement and parallelization capabilities. For these reasons it was performed to realize a low power and efficient design by combining the capabilities of the SNN with the skills of FPGA. In the study, training was performed using the STDP learning rule on the FPGA platform with MNIST dataset. In addition, the Integrate and Fire (IF) Neuron Model, which is preferred by researchers for low resource required hardware-based applications, was used as the neuron model in the SNN architecture and design. The model was tested with IF neurons trained using the MNIST dataset with STDP. As a result of the test, the correct prediction rate of the model was determined as 92.5%.

**Keywords** – SNN, STDP, FPGA, MNIST, IF Neuron

## I. INTRODUCTION

Artificial Neural Networks (ANNs) have become popular with developing technology and have begun to be widely used in many areas such as image classification, speech recognition, image segmentation, robotic systems and automatic control [1]. Achieved successes in these areas, the development of approaches to Neural Networks (NNs) has become attractive for researchers [2]. Recently, Spiking Neural Networks (SNNs), known as third generation NNs, have been developed to mimic biological systems. Unlike traditional ANNs, in SNNs, a spike-based method that works with membrane potential variation is used in processing and transmitting information. In these structures, an event-based logic is adopted, that allows to become active only segments that exceed the predefined threshold level thus both processing and communication costs significantly reduced [3].

As the complexity of SNN algorithms rises, traditional software-based implementations face disadvantages such as performance limitations and high computational costs. These deficits increase the processing time and reduce efficiency. In order to overcome the inherit limitations of SNN, researchers have been enthusiastic to develop hardware-based solution, especially using Field Programmable Gate Arrays (FPGAs) thanks to parallel processing capabilities [4]. FPGAs perform many operations simultaneously owing to parallel processing structures, and this feature increases the speed of SNN calculations and significantly reduces the latency and cost. Therefore, the parallel processing capability of FPGAs has surpassed traditional software methods and enabled complex SNN architectures to be built more efficiently [5]. The combination of FPGA and SNN offers impressive advantages for certain applications due to well adaptation to hardware event-driven nature of spike-based NNs and their parallel processing capability, leading to improved computational efficiency compared to traditional processors [6].

## II. THEORETICAL BACKGROUND

### A. Spiking Neural Network (SNN)

Spiking Neural Networks (SNN), also known as 3rd generation NN, have the most biological similarity among other NNs. Unlike traditional ANNs and CNNs that require continuous computation, SNNs work with potential calculations that vary by discrete events or spikes. Thus, SNNs do not work continuously, neurons in the SNN are activated only when an event or spike occurs, which causes SNNs to be known as models with high energy efficiency [7].

Hardware with SNN architecture will increasingly take a significant place in various areas of our lives such as sensors, internet of things (IoB), robotics, communication, autonomous driving, real time data analyze. Made SNN applications prominent is that they are able to handle the same operation faster than classical NNs with low energy without sacrificing performance [8]. Neurons in the mammalian brain encode information with chemical changes and transmit it to the next neuron. To mimic this procedure, researchers have developed mathematical models of neuron that can respond and react stimuli in inputs [9].

In academic search, one of the frequently preferred models Integrate and Fire (IF) neuron. When a spike is received by a neuron, the membrane potential of neuron is adjusted depending on whether the spike is excitatory or inhibitory. If the membrane potential exceeds a predefined threshold value, the neuron fires and spike spreads to connected neurons and the firing neuron resets its own membrane potential to the initial set value [10].

There are SNN neuron models available, each specialized for a specific task, including Leaky Integrate and Fire (LIF), Hodgkin-Huxley, Integrate and Fire (IF), and Izhikevich models. Each model offers different levels of complexity and

biological realism. They also allow for exclusive applications in neuromorphic engineering to create hardware that imitate neural networks of brain. SNNs are becoming popular for advancing applications due to their potential for low power, high throughput performance [11].

ANNs and CNNs have achieved fascinating success in various applications including handwriting recognition, speech recognition, and automatic control [12]. However, these networks require a lot of hardware resources. Real-time use in edge devices is very difficult as all the neurons in a layer are activated simultaneously and information is processed in a fixed order from one layer to the next. This simultaneous processing needs high computation and data transfer [13]. In contrast, since SNNs work on an event-driven basis, only the neurons that reach a certain threshold of membrane potential are activated. Spikes are propagated to connected others in the next layer as shown in Fig 1. [14].

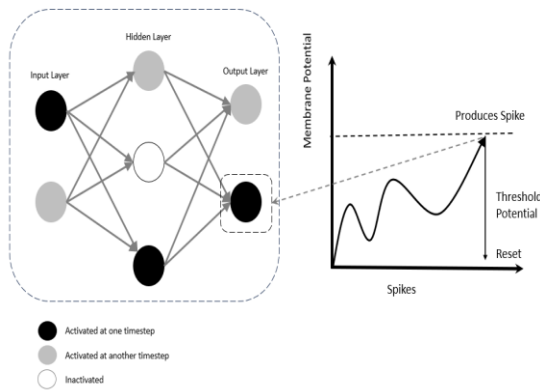


Fig. 1 Operational Process of SNN

Essentially, in SNNs, neurons are not activated sequentially in layers and from one layer to another when firing. As depicted in Fig 1, at each time step, neurons in each layer that exceed the potential threshold fire. As a result during the coding period, a large number of neurons remain passive and are not computed. This asynchronous structure of SNNs allows all layers to operate in parallel, making them more energy-efficient and faster than other NNs [15].

### B. Neuron Models

Primarily, the basic and essential unit of the NN is the neuron and it is major to accept how we characterize it. When a current or stimulus  $I(t) = 0$  arrives at the input of neuron, the membrane potential increases. In the lack of input,  $I(t) = 0$ , the cell returns to resting potential,  $V_{rest}$ . If the potential reaches or exceeds a threshold value,  $V_{thr}$ , the neuron fires and produces a spike. During this firing process, the neuron falls below resting potential after a sudden increase in the membrane potential. This indicates that the neuron is hyperpolarized, and this is considered as refractory period [16].

On account of the balance differences that occur in the direction of the chemical substances found in the neuron, spikes move from axons of then neuron to the axon terminals. When the spike reaches of neurotransmitters in chemical synapses and binds to the receptors on the dendrites of the postsynaptic neurons. Depending on type of neurotransmitter, it can initiate a positive current by stimulating the postsynaptic neuron, or it can initiate a negative current by inhibiting the

postsynaptic neuron [17]. In addition to all these, it has another regulatory effect and the strength of such effects varies between synapses. This whole process is generally known with a parameter called time-dependent synaptic weight, which varies on short and long-time scales. Although the training or transmission data adventure in biological neuron is much more complex, the neuron models that need to be designed should be simplified so that this process can be modeled in software or hardware applications [18].

Moreover, there are many neuron models have been developed and researched [19]. The models that are frequently preferred in studies are Integrate and Fire (IF), Hodgkin-Huxley, Izhikevich, Adaptive exponential integrate and fire (AdEx), Leaky Integrate and Fire (LIF) neuron models. Each model exhibits its own unique features. Although the Hodgkin-Huxley model is the most biologically realistic, it causes high complexity, particularly for hardware applications. As the biological similarity of other model decreases, their applicability increases [20].

When designing SNN applications, especially having large network by using FPGA, the first problem to be solved is to effectively utilize hardware with limited resources. In order to use hardware sources optimally in SNN applications, it is necessary to choose the component that is in large amount in the design [21]. Neurons are responsible major work in the hardware where the SNN application run. Therefore, it would be appropriate to choose a neuron that takes up little space, and not being complex but has biological reality. Considering all these reasons, one of the simplest and most frequently used neuron models in this field was developed by French neuroscientist Louis Édouard Lapicque in 1907 Integrate and Fire neuron model. A neuron model was created using a basic RC (resistor-capacitor) circuit. It works on the principle that the membrane potential increases over time in response to a constant current source applied to the input of the model. The membrane potential is compared with threshold value, if the threshold is reached, the circuit is activated and produces a spike [22].

$$I(t) = C_m \frac{dV_m(t)}{dt}$$

In this RC circuit-based neuron model, that can be seen in Fig 2., the current applied to the neuron’s input is represented  $I(t)$ , the membrane capacitance by  $C_m$ , the membrane potential by  $V_m$ , and the threshold value for the membrane potential by  $V_{th}$ . As the current  $I(t)$  is applied, the membrane potential  $V_m$  increases progressively. Once  $V_m$  reaches  $V_{th}$ , a spike is generated, modeled as a Dirac delta function [23].

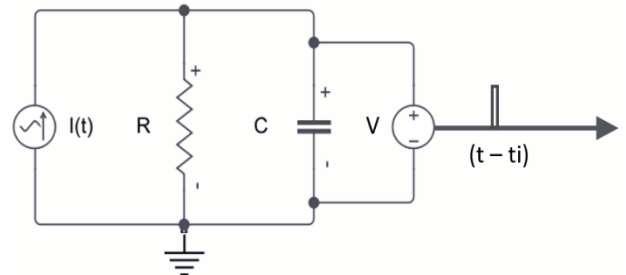


Fig. 2 The Illustration of the RC Circuit Design for IF Neuron Model

### C. Spike Timing Dependent Plasticity (STDP) Learning Rule

Spike Timing Dependent Plasticity (STDP), which is considered as a model of Hebb learning, is a type of long-term synaptic changes and is suitable for plasticity. In the nature of the operation of this mechanism order, frequency and timing of spikes between presynaptic and postsynaptic neurons affect the strengthening (potentiation) or weakening (depression) of synapses, forming the learning mechanism. Studies have shown that STDP plays a crucial role in the training and formation of new structures, and hardware-friendly architectures have been developed with different approaches [24].

To briefly explain the STDP mechanism, when a presynaptic spike precedes a postsynaptic spike, the casual situation occurs and the synapse is strengthened (LTP – Long Term Potentiation). Conversely, once a postsynaptic spike, which is a non-casual situation, takes place first, the synapse is weakened (LTD – Long Term Depression) [25].

Since calculation of synaptic alterations is performed only with the information on the relevant synapse, STDP is a local learning method. The closer positioning of memory and processing units to each other is achieved thanks to locality. Synaptic weights are stored in the memory units, but once the operations are performed in the neurons, there is no need for distant data transfer, while providing energy-efficient on-chip training systems [26].

In general, STDP is a mechanism that allows strong synaptic connections to be strengthened, thus increasing the probability of producing spikes when the neuron sees a pattern it has encountered before. In addition, since STDP has plasticity similar to the learning system of the biological brain, that is, a previously trained model could be restricted and relearned for several investigation [27].

The mathematical notation for a classical STDP mechanism is presented below. In the formula, two presynaptic and one postsynaptic spike (in the scenario where there is one presynaptic and two postsynaptic spikes) are updated with the synapse weights depending on the timing of three spikes. The equation can be used to mathematically express how this mechanism works as follows [28].

$$\Delta W = \begin{cases} \Delta W^+ = e^{-\frac{\Delta t_1}{\tau^+}} \left( A_2^+ + A_3^+ e^{-\frac{\Delta t_2}{\tau_y}} \right) \\ \Delta W^- = -e^{-\frac{\Delta t_1}{\tau^-}} \left( A_2^- + A_3^- e^{-\frac{\Delta t_3}{\tau_x}} \right) \end{cases}$$

Where,  $\Delta W^+$  represents synapse strengthening (potentiation) and  $\Delta W^-$  symbolizes synapse weakening (depression). The terms  $A_2^+$ ,  $A_3^+$ ,  $A_2^-$ , and  $A_3^-$  are parameters that determine the magnitude of strengthening and weakening.  $\tau^+$ ,  $\tau^-$ ,  $\tau_x$ , and  $\tau_y$  are time constants that control the time interval of STDP curve.

### D. FPGA – Driven SNN Implementations

Neuromorphic systems can be classified in three various group: analog, digital, and mixed mode, which assembles both digital and analog. Although the origin story of neuromorphic computing is to mimic the analog nature of biological neural systems, Spiking Neural Networks designed in digital world

have much more benefits in terms of being integrated into large-scale systems [29].

Researchers and technology companies have turned to Neuromorphic Systems to overcome disadvantages of traditional Von Neuman architecture. Neuromorphic Systems have been developed using diverse ASIC chips with special digital designs focused on imitating biological brain functioning [30]. There are architectures that have been made in this direction and are prominent. SpiNNaker developed by Manchester University, TrueNorth chip offered by IBM, and Loihi improved by Intel are the most well-known hardware. Although all these hardware are respectable pavement in Neuromorphic information processing, due to the nature of FPGA, there are limiting factors such as programmability restrictions and access complexities [31].

With convenient optimizations, FPGAs offer higher performance than CPUs and GPUs in certain tasks, and are effective platforms that enable fast, low-cost, and programmable implementation of models [32]. These utilities are largely compatible with SNNs, especially those that have a parallel processing structure and include branching, are largely compatible. Studies shows that implementing SNN applications in FPGA environments has great advantages in terms of speed, energy, and memory. It is clear that FPGAs are a powerful and suitable platform for SNNs [33].

First of the prestigious issues is that if efficiency is at the forefront, many approaches are simplified due to the limited hardware resources of FPGAs. When designing hardware on FPGA, there are differences between the results obtained in software and the implementation in hardware because a direct hardware transfer is generally not possible [34]. Thus, simplifications such as performing digital quantization or representing complex mathematical operations with look up tables (LUT) are essential [35].

## III. MATERIALS AND METHOD

In order to benefit from the biological similarity of the 3rd generation artificial intelligence structure and to use advantages such as speed, parallelism and low power consumption offered by FPGA, an SNN classification implementation created using MNIST dataset [36]. In order to adopt SNN structure and observe compatibility with FPGA platform, two-class design was planned using MNIST dataset. The study was carried out with Very High-Speed Integrated Circuit Hardware Description Language - VHDL in the ISE Design Suite 14.7 hardware design program.

First of all, the suitability of neuron models used in SNN structure in FPGA hardware was evaluated. Although there are many neuron models in the literature, the Integrate and Fire neuron model (IF) was chosen in the work due to hardware limitations of FPGAs.

The headstone of study IF neuron model was designed and simulated in hardware. A threshold value was determined for neuron and when accumulated membrane potential value of the neuron exceed the threshold value, the neuron produce a spike. This hardware was made with accumulator circuit in the neuron. The spikes incoming to the neuron input are collected via the accumulator circuit and final membrane potential value is found. The weight value is also used when calculating the membrane potential. The calculation result of membrane potential is defined as 16 bits. After the neuron produced spike, the membrane potential of neuron is reset to the initial value. The hardware design of IF neuron is given in Fig 3.

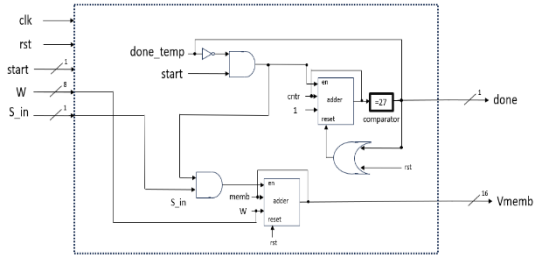


Fig. 3 IF Neuron

STDP training model was selected for learning SNN architecture. 1024 samples of the 0 number and 1024 samples of the 1 number were as training data. The weight values obtained after learning were stored in internal RAM on the FPGA board. Later, when data to be estimated was given to the model, weight values in RAM are used. The weight values updated with STDP are determined as 8 bits. Thus, the memory in which weights are stored is 8 bits. Since the MNIST image has 28 rows pixels, the RAM size is selected as 28 lines. The calculation of weight values was made in accordance with STDP learning mechanism. The hardware design of STDP is presented in Fig 4.

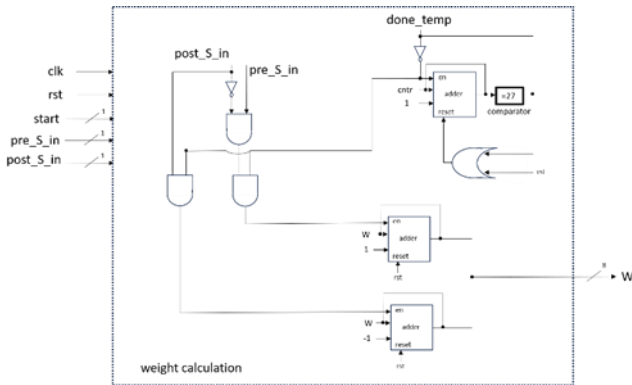


Fig. 4 STDP Learning Mechanism

MNIST dataset consists of 784 pixels with size of 28x28. Each pixel carries values between 0-255 according to color of pixel [37]. Binarized before MNIST data values was given to model for training. 200 was selected as the threshold value and

binarization process was performed for the image. The representation of a MNIST data of number 0 and after binarization is given Fig 5.

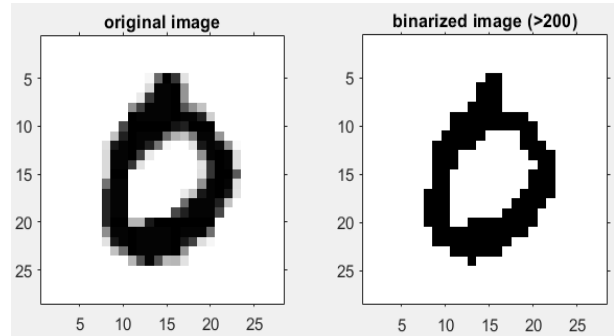


Fig. 5 Original and Binarized Images

When the Fig 5. is examined, it is seen that noise in the original image is removed after binarization. In addition, thanks to binarized data, only binary numbers (black: 1 and white: 0) are used for the learning so making the hardware simpler. Thus, spikes are prepared as input data. Then, 28 neuron components working simultaneously are designed and the first row of binarized image is given to 28 neurons working simultaneously in one clock duration. In the following clocks, the spike values of the other rows are given to 28 neurons again. Thus, the weight calculation of an image is completed and the next sample is passed to update calculation of weight values.

#### IV. RESULTS

Training process was completed in 1024 (number of samples) x 28 (process time for one sample) = 28.672 clock period to update the weight values for number 0. Thanks to parallel structure of the FPGA, the learning for 0 and 1 were received by model at the same time and process was carried out simultaneously and updated weight values for two numbers are stored in separate RAMs. The hardware diagram of training process is given in Fig 6.

After the model training was completed, the binarized test data for prediction is given to the design. Fig 7. shows the hardware diagram of prediction process.

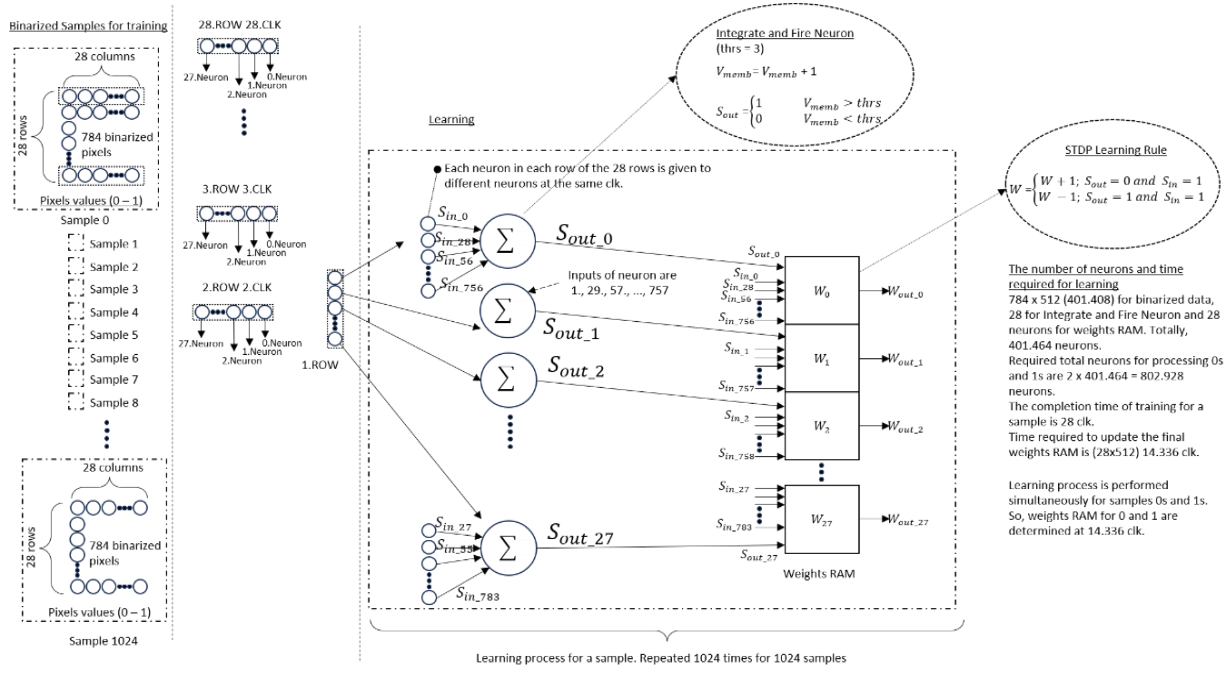


Fig. 6 Hardware Diagram of Training Process

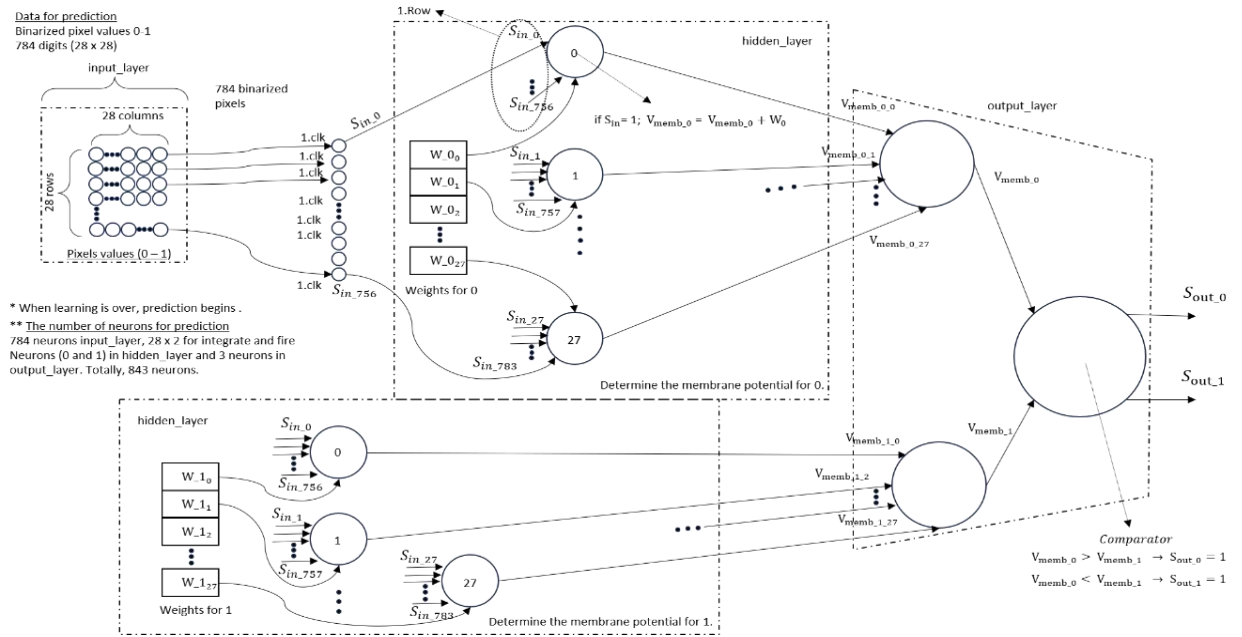


Fig. 7 Hardware Diagram of Prediction Process

With the weight values of the number 0 and 1 in different RAMs, two membrane potential values were calculated for prediction. The neuron that has the highest membrane potential among the neurons created in output layer for number 0 and 1, produces spike. According to various test data given to the design, accuracy of system is 92.5%. The selected board for SNN model in ISE Design Suit 14.7 is Xilinx Spartan 6 XC&SLX75T. Device utilization summary of full model after training and prediction is exhibited in Fig 8.

```

Device utilization summary:
-----
Selected Device : 6slx75tfgg676-3

Slice Logic Utilization:
Number of Slice Registers:    3055 out of 93296    3%
Number of Slice LUTs:        2880 out of 46648    6%
  Number used as Logic:      2860 out of 46648    6%
  Number used as Memory:     20 out of 11072    0%
  Number used as RAM:        20
Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 3651
  Number with an unused Flip Flop: 596 out of 3651    16%
  Number with an unused LUT:      771 out of 3651    21%
  Number of fully used LUT-FF pairs: 2284 out of 3651    62%
  Number of unique control sets: 206

IO Utilization:
Number of IOs:                88
Number of bonded IOBs:        88 out of 348    25%

Specific Feature Utilization:
Number of BUFG/BUFGCTRLs:     1 out of 16    6%

```

Fig. 8 Device Utilization Summary

## V. DISCUSSION

Thanks to ability of the SNN architecture to mimic the biological nervous system and by combining this structure with the efficient hardware platform FPGA, it will be possible to obtain instant results. By increasing the number of training data and the number of hidden layers, the obtained with the study 92.5% correct prediction rate will increase. the number of classes will be increased as much as hardware restriction allows and deep neural network architectures will be employed to design to increase accuracy of hardware. In addition, the necessary optimization techniques will be carried out to reduce device utilization rate of board. As the model grows, the obstacles will arise and be overcome with seeking solutions for the bottlenecks in the implementation of SNN architecture on FPGA.

## VI. CONCLUSION

Parallelization capability has been utilized at a superior level by combining SNN architecture with FPGA. The board Xilinx Spartan 6 XC6SLX75T used in the study has a 100 MHz oscillator, so the training process obtained with hardware is approximately 0.287 ms. Obtaining this result with a software-based architecture will be more costly than obtaining it with hardware. As a result of these results, it is thought that the study will be a source for research combining artificial intelligence with hardware.

## REFERENCES

- [1] Belatreche, A., Maguire, L. P., and McGinnity, M., *Advances in design and application of spiking neural networks*. Soft Computing, 2007, 11, 239-248.
- [2] Upegui, A., Pena-Reyes, C. A., and Sanchez, E., *An FPGA platform for on-line topology exploration of spiking neural networks*. Microprocessors and microsystems, 2005, 29(5), 211-223.
- [3] Wang, R., Hamilton, T. J., Tapson, J., and van Schaik, A., *An FPGA design framework for large-scale spiking neural networks*. In 2014 IEEE International Symposium on Circuits and Systems (ISCAS), 2014, (pp. 457-460). IEEE.
- [4] Roggen, D., Hofmann, S., Thoma, Y., and Floreano, D., *Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot*. In NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings, 2003, (pp. 189-198). IEEE.
- [5] Ambroise, M., Levi, T., Bornat, Y., and Saighi, S., *Biorealistic spiking neural network on FPGA*. In 2013 47th Annual Conference on Information Sciences and Systems (CISS), 2013, (pp. 1-6). IEEE.
- [6] Ju, X., Fang, B., Yan, R., Xu, X., and Tang, H., *An FPGA implementation of deep spiking neural networks for low-power and fast classification*. Neural computation, 2020, 32(1), 182-204.
- [7] Yamazaki, K., Vo-Ho, V. K., Bulsara, D., and Le, N., *Spiking neural networks and their applications: A review*. Brain sciences, 2022, 12(7), 863.
- [8] Yi, Y., Liao, Y., Wang, B., Fu, X., Shen, F., Hou, H., and Liu, L., *FPGA based spike-time dependent encoder and reservoir design in neuromorphic computing processors*. Microprocessors and Microsystems, 2016, 46, 175-183.
- [9] Rathin, N., Chakraborty, I., Kosta, A., Sengupta, A., Ankit, A., Panda, P., and Roy, K., *Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware*. ACM Computing Surveys, 2023, 55(12), 1-49.
- [10] Yang, S., Wang, J., Deng, B., Liu, C., Li, H., Fietkiewicz, C., and Loparo, K. A., *Real-time neuromorphic system for large-scale conductance-based spiking neural networks*. IEEE transactions on cybernetics, 2018, 49(7), 2490-2503.
- [11] Shen, J., Ma, D., Gu, Z., Zhang, M., Zhu, X., Xu, X., and Pan, G., *Darwin: a neuromorphic hardware co-processor based on Spiking Neural Networks*. Sci. China Inf. Sci., 2016 59(2), 1-5.
- [12] Fidjeland, A. K., and Shanahan, M. P., *Accelerated simulation of spiking neural networks using GPUs*. In The 2010 International Joint Conference on Neural Networks (IJCNN), 2010, (pp. 1-8). IEEE.
- [13] Courtois, J., Novac, P. E., Lemaire, E., Pegatoquet, A., and Miramond, B., *Embedded event based object detection with spiking neural network*. In 2024 International Joint Conference on Neural Networks (IJCNN), 2024, (pp. 1-8). IEEE.
- [14] Kim, Y., and Panda, P., *Visual explanations from spiking neural networks using inter-spike intervals*. Scientific reports, 2021, 11(1), 19037.
- [15] Jiménez-Fernández, A., Cerezuela-Escudero, E., Miró-Amarante, L., Domínguez-Morales, M. J., de Asís Gómez-Rodríguez, F., Linares-Barranco, A., and Jiménez-Moreno, G., *A binaural neuromorphic auditory sensor for FPGA: a spike signal processing approach*. IEEE transactions on neural networks and learning systems, 2016, 28(4), 804-818.
- [16] Wu, J., Zhan, Y., Peng, Z., Ji, X., Yu, G., Zhao, R., and Wang, C., *Efficient design of spiking neural network with STDP learning based on fast CORDIC*. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(6), 2522-2534.
- [17] Abderrahmane, N., Lemaire, E., and Miramond, B., *Design space exploration of hardware spiking neurons for embedded artificial intelligence*. Neural Networks, 2020, 121, 366-386.
- [18] Stuijt, J., Sifalakis, M., Yousefzadeh, A., and Corradi, F., *µBrain: An event-driven and fully synthesizable architecture for spiking neural networks*. Frontiers in neuroscience, 2021, 15, 664208.
- [19] Işık, M., *A survey of spiking neural network accelerator on fpga*, 2023, arXiv preprint arXiv:2307.03910.
- [20] Diehl, P. U., Zarella, G., Cassidy, A., Pedroni, B. U., and Neftci, E., *Conversion of artificial neural networks to spiking neural networks for low-power neuromorphic hardware*. In 2016 IEEE International Conference on Rebooting Computing (ICRC), 2016, (pp. 1-8). IEEE.
- [21] Guo, W., Fouda, M. E., Eltawil, A. M., and Salama, K. N., *Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems*. Frontiers in Neuroscience, 2021, 15, 638474.
- [22] Schrauwen, B., D'Haene, M., Verstraeten, D., and Van Campenhout, J., *Compact hardware liquid state machines on FPGA for real-time speech recognition*. Neural networks, 2008, 21(2-3), 511-523.
- [23] Ali, A. H., Navardi, M., and Mohsenin, T., *Energy-Aware FPGA Implementation of Spiking Neural Network with LIF Neurons*, 2024, arXiv preprint arXiv:2411.01628.
- [24] Dan, Y., and Poo, M. M., *Spike timing-dependent plasticity of neural circuits*. Neuron, 2004, 44(1), 23-30.
- [25] Diehl, P. U., and Cook, M., *Unsupervised learning of digit recognition using spike-timing-dependent plasticity*. Frontiers in computational neuroscience, 2015, 9, 99.
- [26] Zamarreño-Ramos, C., Camuñas-Mesa, L. A., Pérez-Carrasco, J. A., Masquelier, T., Serrano-Gotarredona, T., and Linares-Barranco, B., *On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex*. Frontiers in neuroscience, 2011, 5, 26.
- [27] Frenkel, C., Bol, D., and Indiveri, G., *Bottom-up and top-down approaches for the design of neuromorphic processing systems:*

- Tradeoffs and synergies between natural and artificial intelligence*. Proceedings of the IEEE, 2023, 111(6), 623-652.
- [28] Markram, H., Gerstner, W., and Sjöström, P. J., *A history of spike-timing-dependent plasticity*. Frontiers in synaptic neuroscience, 2011, 3, 4.
- [29] Gupta, S., Vyas, A., and Trivedi, G., *FPGA implementation of simplified spiking neural network*. In 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2020, (pp. 1-4). IEEE.
- [30] Yan, Y., Chu, H., Chen, X., Jin, Y., Huan, Y., Zheng, L., and Zou, Z., *Graph-based spatio-temporal backpropagation for training spiking neural networks*. In 2021 IEEE 3rd international conference on artificial intelligence circuits and systems (AICAS), 2021, (pp. 1-4). IEEE.
- [31] Engelen, M., Betting, R., and Strydis, C., *SimHH: A Versatile, Multi-GPU Simulator for Extended Hodgkin-Huxley Networks*. IEEE Access, 2025.
- [32] Cao, Y., Chen, Y., and Khosla, D., *Spiking deep convolutional neural networks for energy-efficient object recognition*. International Journal of Computer Vision, 2015, 113, 54-66.
- [33] Zambelli, C., and Ranhel, J., *Half-precision floating point on spiking neural networks simulations in FPGA*. In 2018 International Joint Conference on Neural Networks (IJCNN), 2018, (pp. 1-6). IEEE.
- [34] Rathi, N., Chakraborty, I., Kosta, A., Sengupta, A., Ankit, A., Panda, P., and Roy, K., *Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware*. ACM Computing Surveys, 2023, 55(12), 1-49.
- [35] Yi, Y., Liao, Y., Wang, B., Fu, X., Shen, F., Hou, H., and Liu, L., *FPGA based spike-time dependent encoder and reservoir design in neuromorphic computing processors*. Microprocessors and Microsystems, 2016, 46, 175-183.
- [36] Carpegna, A., Savino, A., and Di Carlo, S., *Spiker: an fpga-optimized hardware accelerator for spiking neural networks*. In 2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2022, (pp. 14-19). IEEE.
- [37] Xiao, H., Rasul, K., and Vollgraf, R., *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*. arXiv preprint, 2017, arXiv:1708.07747.