Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 04 (2023)

FPGA Realization of a High-Speed Spiking Neural Network with Modified LIF Neurons for Pattern Recognition

K. Venkateswara Reddy[1], Dr.N. Balaji[2]

1 Research Scholor, Department of ECE, JNTU Kakinada, A.P, India,

2 Professor of ECE, Department of ECE, JNTU Kakinada, A.P, India,

Abstract:-This paper presents the multiplier-less Modified Leaky Integrate and Fire Neuron Unit (MLIFNU) and proposed a modified Spiking Neural Network (SNN) architecture for Pattern Recognition application. Synthesis results of implementation on Field-Programmable Gate Array are presented. Accordingly, the maximum frequency of the MLIFNU and SNN are 361.991 MHz and 281.442 MHz, respectively.

Keywords*: Spiking Neural Network (SNN), Artificial Intelligence, Neuromorphic system, Modified Leaky Integrate and Fire Neuron Unit (MLIFNU).*

1. Introduction

Most healthcare environments need portable devices that can handle data efficiently. The memory requirements and analysis time are high in all the earlier biomedical systems. Nevertheless, nowadays, methods like neural networks make systems easily predict and diagnose diseases. Artificial neural networks (ANNs) have been used to mimic many of the most sophisticated areas of science and technology in recent years, posing numerous daunting challenges [1]. The advancement of artificial neural networks has resulted in advancements in a variety of cognitive functions and tools for object recognition, language identification, and other areas.

Furthermore, specific ANN models and numerical methods produce results comparable to human performance. Biological neurons in the human brain generate spikes, which transfer information. Similarly, spiking neural networks (SNNs), the third generation of ANNs were designed to increase the biological plausibility of current ANNs by functioning just like biological neurons. Because SNNs more faithfully imitate the activity of actual neurons by combining neurons and synapses, they serve a significant role in the modeling of essential systems in neuroscience [2]. Neurons in SNNs, in particular, communicate information only when their membrane potential, or an inherent characteristic of the neuron related to its membrane electrical charge, reaches a particular value. When the neuron's membrane potential exceeds its threshold, the neuron spikes and creates a signal that passes to neighboring neurons. As a result, a spiking neuron is a neuron that fires in a membrane potential model at the moment of threshold crossing. In various settings, several models have been developed to characterize the spiking actions of neurons. A leaky integrate-and-fire (LIF) neuron model is one of the most basic models, and it serves as the foundation for many biomedical and neuromorphic applications [3,4,5].

The relevance of the LIF model, which has become one of the most common neuron models in neuromorphic computing, has been established in recent research [6,7]. When applied to large neural networks, the LIF model replicates the movements of the cell membrane in a biological system [8] and provides a good balance of complexities and analytical controllability. However, ANNs are computationally costly and frequently face a variety of problems, including considerable accuracy decline if the testing data is contaminated with noise, which may not be seen during training. Furthermore, uncertainties from other sources, such as inputs, devices, chemical processes, etc., would need to be taken into consideration.

The multiplier-less Modified Leaky Integrate and Fire Neuron Unit (MLIFNU) is proposed in this work. Furthermore, this paper presents a modified Spiking Neural Network (SNN) architecture for pattern recognition applications. Accordingly, the maximum frequency of the MLIFNU and SNN are 361.991 MHz and 281.442 MHz, respectively, and the synthesis results of implementation on a field-programmable gate array are presented.

The following is a breakdown of the structure of the paper's content. We present some background literature on several neuromorphic VLSI implementations in Section 2 and describe the representation of the basic LIF neuron model in Section 3. The learning algorithm employed in this paper is shown in Section 4. Section 5 describes the SNN architecture used for pattern recognition problems and section 6 illustrates the Design and implementation of the proposed Modified Leaky Integrate and Fire Neuron Unit (MLIFNU). Section 7 shows the hardware implementations and summarizes the experimental findings of the research, whereas section 8 gives the path for future research, which brings the study to a conclusion.

Motivation and Problem Statement

The concept of employing specialized hardware to construct neurologically based systems is as old as other fundamental fields such as computer science. For a long time, computer scientists have wished to imitate organic brain networks in their machines. Machine learning, artificial neural networks, and artificial intelligence have all benefited from this research. However, the focus of this research is not only on neural networks but also on building a modified neuron model and modeling the suggested SNN using FPGA implementation. The creation of hardware that could conduct parallel operations, inspired by observable parallelism in human brains, but on a single chip, sparked a lot of early work in neuromorphic computing [9], [10]. Despite the availability of parallel processing, neuromorphic systems emphasized a large number of simple processing units (usually in the form of neurons) with dense interconnections between them (usually in the form of synapses), distinguishing them from other parallel computing systems at the time. The intrinsic parallelism of neuromorphic systems was the most prominent rationale for bespoke hardware implementations in early efforts on neuromorphic computing.

The emphasis on neuromorphic systems' capability for considerably lower power operations is the most common motive in current literature and discussions of neuromorphic systems in the mentioned publications. The human brain, which is our main source of inspiration, uses roughly 20 watts of electricity and can execute incredibly complicated computations and jobs on that tiny amount of power. The goal to construct neuromorphic devices that use comparably low power has been a driving factor for neuromorphic computing from its inception [11], [12], although it only became a significant incentive a decade into the field's history.

Motivated by biological discoveries, many different neural network models have been presented to reproduce their behavior dynamically [13]. Mostly those are described using differential equations. Even though they have more accuracy, due to their computational complexity, most of them are rarely used in large-scale SNNs. Therefore, varieties of simplified models are presented for studies in the field of Neuromorphic computing. There is a tradeoff between the biological accuracy of the model and its complexity.

The speed of computing was another prominent motivation for early neuromorphic and neural network hardware implementations. Therefore, the synthesis and physical implementation of an SNN on FPGA with very high speed is a challenging task and this is chosen as the problem statement. The existing FPGA implementation of the SNN has shown considerable performance with low hardware overhead costs. The work is planned to extend if any feature detection application is possible.

2. Literature Survey on Neuro-Morphic VLSI Architecture

Edris Zaman Farsa et al. [17] proposed an FPGA implementation of SNN without using any multiplier. Spiking Timing Dependent Plasticity (STDP) off-chip learning is used to generate a weight matrix for Pattern Recognition application. Hardware synthesis results show that the proposed SNN has provided more speed and lesser area usage. Eugene M. Izhikevich [18] proposed many models of spiking neurons andpresented a detailed

mathematical analysis that can be useful for choosing suitable SNNs for the specific application. Also discussed was the bio-realistic nature and efficiency of computation of some of the SNN models. F. Christophe et.al. [19] proposed a training method for SNNs for pattern recognitionbased on the STDP of connections. STDP is a rule for neurons to strengthen or weaken their connections according to their degree of synchronous firing. Hamid et.al. [20] presented a set of non-multiplier biologically inspired neuron models based on popular model of Izhikevich and developed an architecture for a network of the proposed neuron models which can be implemented in a more speed and less area. The maximum clock rate of piecewise linear (PWL) models has reached up to 200 MHZ. Moslem Heidarpur et.al. [21] proposed a neuromorphic platform for on-FPGA online STDP learning, based on the COordinate Rotation DIgital Computer algorithms. STDP algorithm was adopted for online learning. AyoubAdineh-vand et.al. [22] presented an architecture for spiking convolutional neural networks and is implemented in an embedded system and their potential is evaluated in terms of utilization of hardware and consumption of power in complex applications such as the detection of tumors. Izhikevich neuron model is used with the STDP learning rule.

Frank L. Maldonado Huayaney et.al [23] proposed an analog CMOS VLSI circuit that implements integrate-and-fire SNNs with STDP. M. Nouri et.al. [24] presented a set of PWL approximations in order to produce a digital neuromorphic realization of the STDP rules. J. Seo et al. [25] presented a new architecture by combining innovations in computation, SRAM array, and inter-neuron communication. Luis A. Camunas-Mesa et.al.[26] developed hybrid memristor-CMOS approaches to implement complex SNNs with learning abilities, offering a scalable and reduced cost. P. U. Diehl et.al. [27] presented a train-and-compel strategy that is viable with the abilities of recent and near-future neural network frameworks.

3. Basic LIF neuron model

As per the survey presented by Catherine D. Schuman et.al. on Neuromorphic Hardware [28] the following Fig. 1. shows the comparison of neural network models in terms of Biological Inspiration and Complexity of the neuron model.

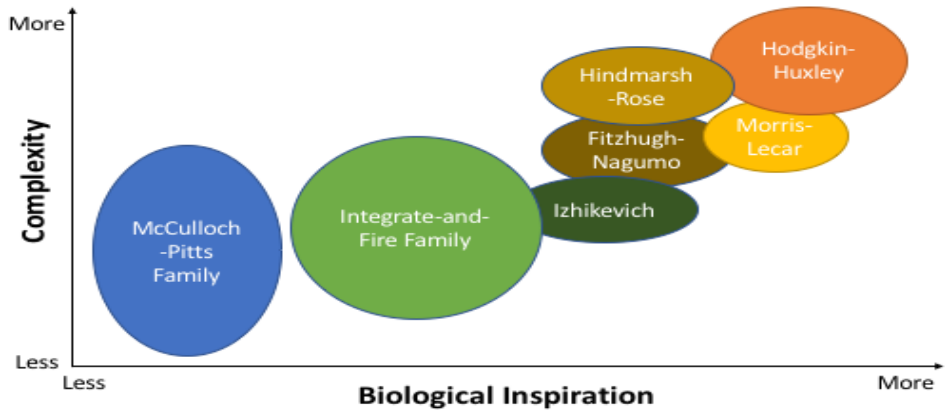

Fig. 1. The Comparison of neural network models in terms of biological inspiration and complexity of the neuron model [28].

Based on Fig.1 LIF is one of the simple biologically inspired SNN models in computational neuroscience. These are less biologically realistic with enough computational complexity providing good tradeoffs.
The basic LIF model is proposed as

$$\frac{\partial v}{\partial t} = I + a - bv, if\ v \geq v_{thresh}\ then\ v \leftarrow c \quad \text{..... (1)}$$

where

v = membrane potential
I = input current (internal parameter)
a = external parameter

b = resting potential

v_{thresh} = threshold value of membrane potential.

The schematic circuit diagram of the LIF model is shown in the Fig.2.

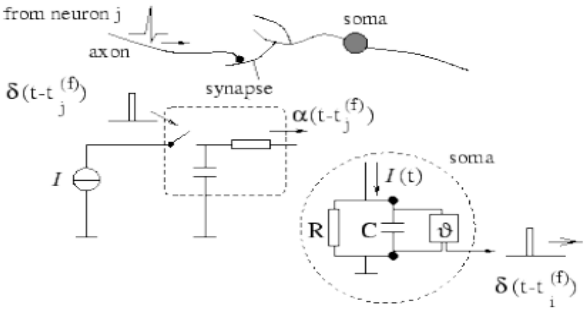

Fig. 2. A Schematic circuit diagram of the LIF model [29].

The LIF model can be described by one differential equation eq. (2) as

$$\tau_m \frac{dv}{dt} = -v + RI \quad \text{..... (2)}$$

The LIF model is discretized by applying the Euler method as

$$v(n+1) = v(n) + \frac{h}{\tau_m}(-v(n) + RI) \quad \text{..... (3)}$$

Here h represents the discretizing step whose value is 0.001 with τ_m = 10ms and by choosing R and I values as proposed in [29] and step $\frac{h}{\tau_m}$ =0.1.

$$v(n+1) = v(n) + 0.1\,(-v(n) + RI) \quad \text{..... (4)}$$

This can be approximated as below to reduce hardware

$$v(n+1) = v(n) + 0.125\,(-v(n) + RI) \quad \text{..... (5)}$$

The equation (5) is proposed by Zaman Farsa et.al [13] as modified LIF model and its scheduling diagram is shown in Fig. 3. below. Here 0.125 can be obtained by three arithmetic right shifts (>>3).

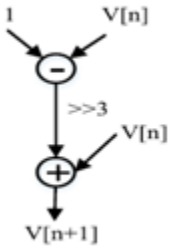

Fig.3. The tree diagram of the simplified LIF model [17].

4. Spiking neural network model and pattern recognition

The Spiking Neural Networks (SNNs) are faster, more accurate, and computationally powerful. The SNN models accurately describe the nervous system and other machine learning algorithms can also be incorporated efficiently. This makes the architecture better when compared to the conventional Artificial Neural Network (ANN) architectures. In the existing work by Farsa et. al. [17] The reported investigation shows these are less biologically realistic with enough computational complexity providing a good tradeoff. An SNN with the LIF neurons for pattern recognition is structured [14] in Fig. 4. The network consists of 25 dummy neurons in the input layer, 5 MLIF neurons in the hidden layer, and 1 MLIF neuron in the output layer. Fig.5 shows the noise-less patterns of the 'O' (circle) and 'X' (cross) that are used as inputs for pattern recognition.

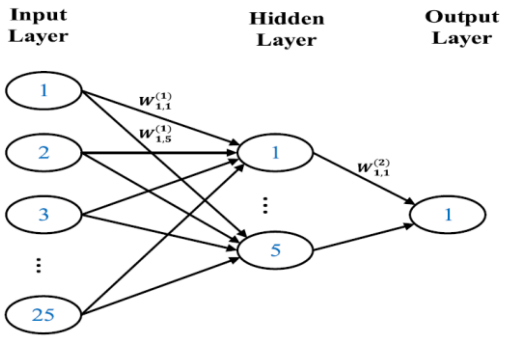

Fig. 4 The structured SNN [14]

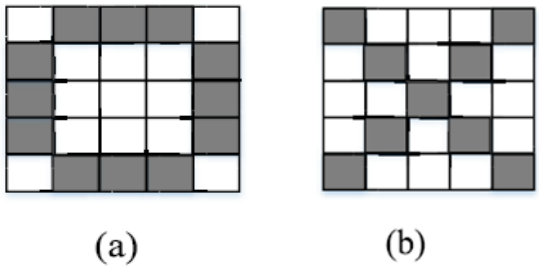

Fig. 5 (a) O pattern (b) X pattern

5. Design and implementation of Modified Leaky Integrate and Fire Neuron Unit (MLIFNU)

The module depicted in Fig 6 is designed to exhibit the behavior outlined in the MLIFNU proposed in [17]. In this module, an input voltage V[n] is utilized in calculations to produce another voltage, V[n+1], as illustrated. The approach in [17] considered V[n] as an internal signal with an initial value of zero, but this proved impractical for system integration. Modifications were thus implemented, designating V[n] as an input and incorporating a Finite State Machine (FSM) for the process.

The FSM involves several steps:

1. In Step S0, the value of V[n] is captured and assigned to a register named C.
2. Step S1 entails subtracting 1 - V[n], and the result is denoted as A.
3. In Step S2, the discretization step A/8 is executed by employing three arithmetic right shifts, implemented through concatenation to minimize area increase, as opposed to using an ordinary shift right operator.
4. Step S3 involves computing V[n] + A/8.
5. Finally, in Step S4, the resulting value is compared with a threshold value of 0.98, as proposed in [29].

If the final value exceeds the threshold, it is considered a "fire," and this value becomes the output. If not, the process proceeds to Step S5, and the output remains zero, indicating no fire (reset to zero). Notably, the specifics of Step 4, outlining how the MLIFNU should operate, were not detailed in [17].

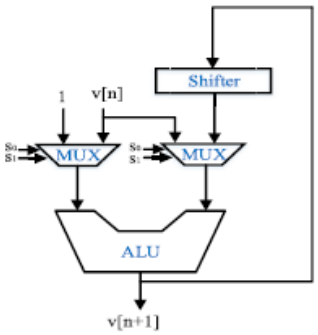

Fig. 6 The modified LIF neuron unit (MLIFNU)

The device utilization summary, presented in Fig 7, provides an overview of the resources consumed by the MLIFNU. This includes details on how various components and features of the device are allocated within the hardware.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	138	301440	0%
Number of Slice LUTs	180	150720	0%
Number of fully used LUT-FF pairs	104	214	48%
Number of bonded IOBs	67	600	11%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig. 7. Device utilization summary of MLIFNU

Moving on to the timing report illustrated in Fig. 8 outlines the performance metrics of the MLIFNU. The Max Speed indicated as 361.991 MHz, represents the maximum clock frequency achievable by the system. This metric is crucial in assessing the efficiency and speed of the MLIFNU in processing tasks.

```
Timing Summary:
---------------
Speed Grade: -1

   Minimum period: 2.762ns (Maximum Frequency: 361.991MHz)
   Minimum input arrival time before clock: 1.504ns
   Maximum output required time after clock: 0.777ns
   Maximum combinational path delay: No path found
```

Fig. 8. Timing report of MLIFNU

In comparison to the NCHU proposed in [17], the modified MLIFNU exhibits a slightly lower maximum speed. This reduction in speed is attributed to the introduced modifications aimed at enhancing the device's practical applicability. While there is a trade-off between speed and functionality, the modifications are justified by the improved integration into broader systems.

Fig. 9 presents the behavioral simulation of the MLIFNU, offering a dynamic view of its functionality. This simulation showcases how the device responds to different inputs and processes information over time. It serves as a valuable tool for validating the performance and behavior of the MLIFNU under varying conditions, providing insights into its operational characteristics.A Spiking Neural Network (SNN) with MLIFNU is trained to recognize O and X patterns from binary pixel representations. Dark pixels represent logic 0, and white pixels represent logic 1. The SNN, during simulation, fires spikes when accurately identifying a pattern. The simulation waveform reflects the spiking activity over time. When the exact pattern (O or X) is recognized, the

output becomes logic 1. MLIFNU integrates machine learning and fuzzy logic for sophisticated pattern recognition. The network learns associations between spiking patterns and specific input patterns. This fusion enables nuanced and accurate identification in the presence of noise or variations. The SNN's firing behavior signifies successful recognition, evident in the simulation waveform. In essence, the SNN with MLIFNU provides a robust and adaptive solution for precise pattern recognition.

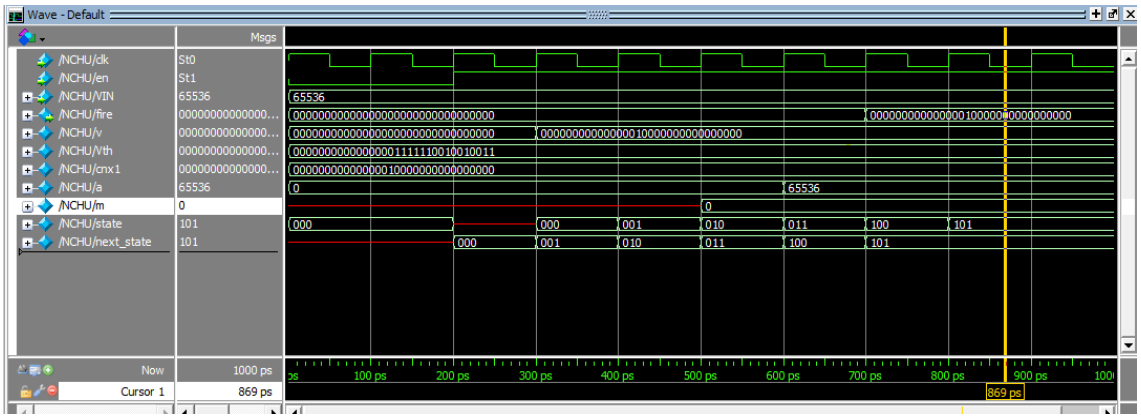

Fig. 9. Behavioral simulation of MLIFNU

6.Hardware implementation and Experimental results

The neuromorphic system architecture, as outlined in literature [17] and depicted in Fig. 10, encompasses key components such as the Input Provider Unit, SNN Controller Unit, and a dedicated Synaptic Weights Memory. The Input Provider Unit is responsible for supplying inputs, while the SNN Controller Unit manages the timing and coordination of all system units. The Synaptic Weights Memory stores essential synaptic weights for neural computations. Building upon this architecture, Fig. 11 details the spiking neural network (SNN) hardware design, featuring input pixels, six Modified Leaky Integrate-and-Fire Neuron Units (MLIFNUs), a single Static Random-Access Memory (SRAM) block, five Adder_25 blocks for 25-bit arithmetic operations, an Adder_5 block for 5-bit arithmetic, and buffer blocks (FIFO_5 and FIFO_25). The final output, represented by Output Spikes, reflects the culmination of the SNN model's computations. This comprehensive design ensures effective neural processing, precise timing control, and efficient storage and retrieval of synaptic weights, facilitating the implementation of a functional spiking neural network.

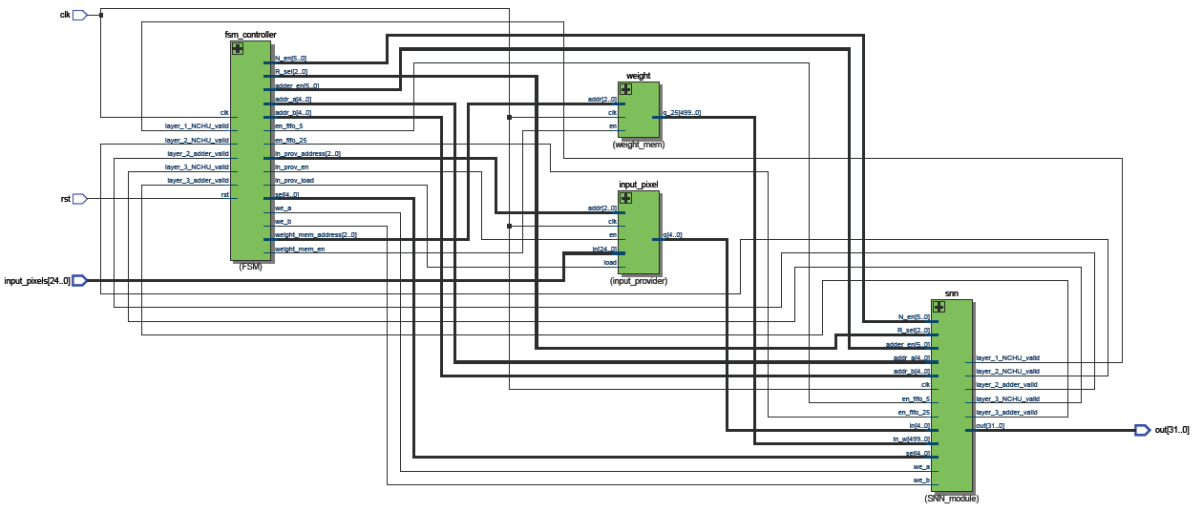

Fig. 10 The neuromorphic system architecture

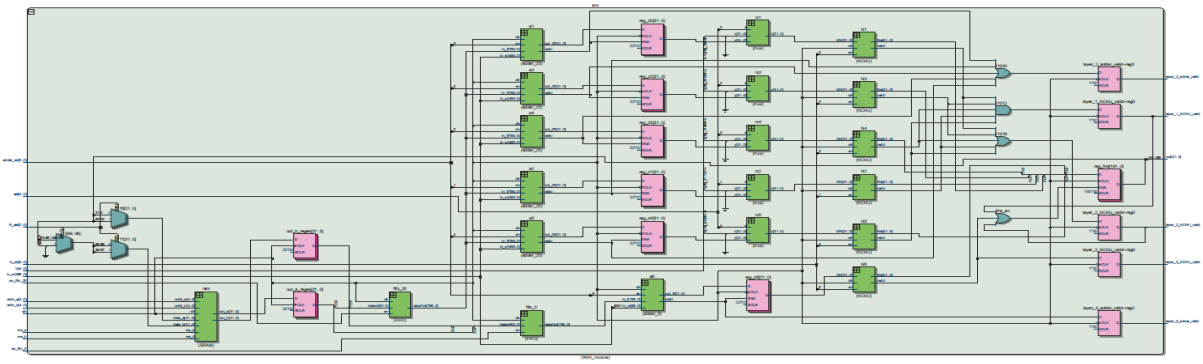

Fig. 11 Proposed SNN architecture

1.Adder_25

In the depicted hardware design, 25 fire inputs and weights undergo multiplication in a system featuring a 5-level adder tree. Registers strategically placed between stages allow for frequency increase, enhancing processing speed. The system efficiently utilizes two key blocks: an ordinary adder block and a pipeline DSP multiplier. The adder tree hierarchically accumulates the multiplied results. The pipeline DSP multiplier employs a sequential processing approach to optimize multiplication speed. The Register-Transfer Level (RTL) schematic, illustrated in Fig. 12, likely showcases the interconnection of add and mult blocks. This design is both scalable and flexible, accommodating 25 inputs and weights. The integration of a DSP multiplier emphasizes the system's proficiency in digital signal processing tasks. The use of registers between stages enhances concurrent processing, contributing to overall system efficiency. The 5-level adder tree strikes a balance between performance and complexity, optimizing computation efficiency. The system's structure suggests adaptability for diverse applications, providing a well-organized solution for high-speed and scalable computations.

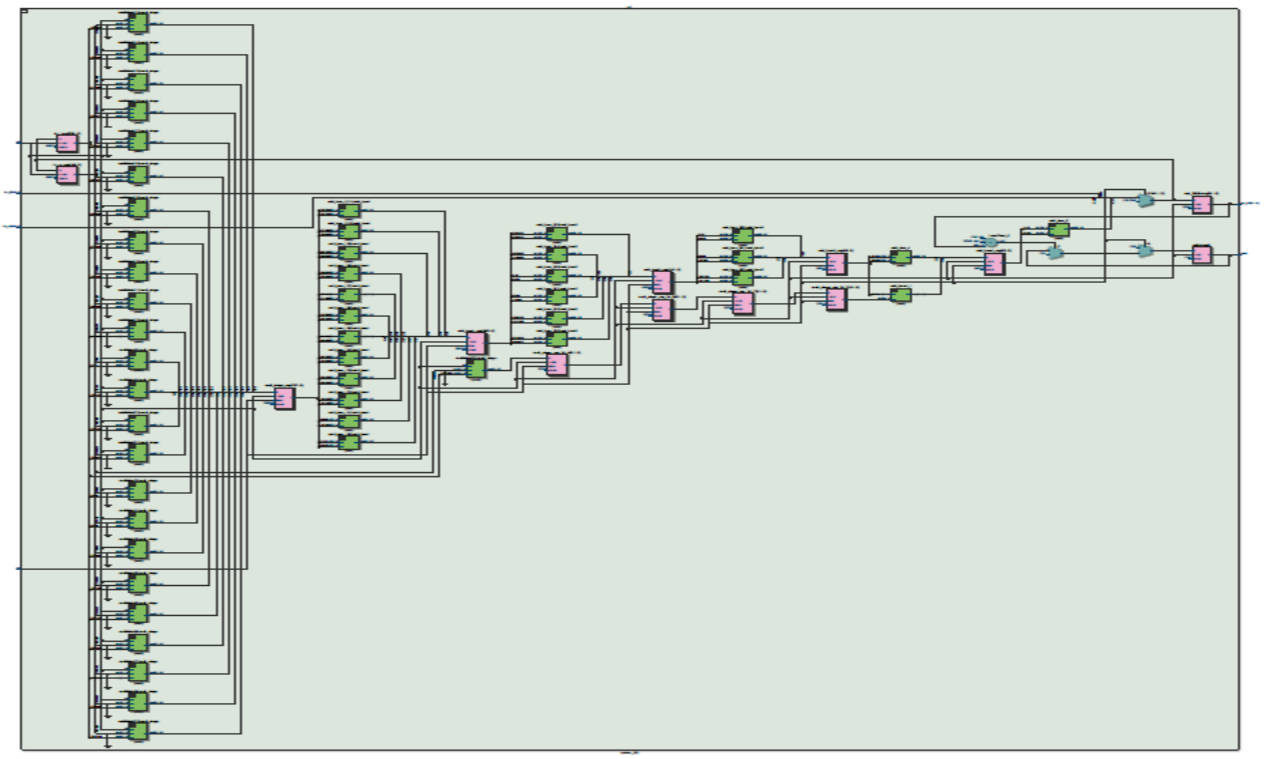

Fig. 12. The RTL schematic of adder_25 block

2. Adder_5

Adder_5 is a component in the hardware design with a structure similar to adder_25 but designed for a smaller set of inputs. It utilizes add and multsub-blocks for efficient arithmetic operations. The distinction in size implies its capacity to process five inputs. The design follows a modular and scalable approach, allowing for flexibility in adapting to different computational requirements. The shared structure with adder_25 suggests a cohesive system design for varying scales of computations. Likely sharing an RTL schematic, Adder_5 contributes to overall efficiency in arithmetic operations within the larger system. Its use of subblocks aligns with a focus on optimizing computational speed and resource utilization. Adder_5 and adder_25 work together cohesively within the system, each serving specific computational tasks. The design showcases adaptability and purposeful modularity, catering to diverse use cases. In summary, Adder_5 is a specialized and efficient component designed for processing smaller sets of inputs within the broader hardware framework.

3. SRAM

The SRAM (Static Random-Access Memory) described is a standard 2-port read and write memory unit, sourced from reference [16]. This SRAM is capable of simultaneously reading two values from distinct locations and performing write operations. However, a challenge arises when attempting to read a large volume of data at once, prompting the introduction of a First In First Out (FIFO) solution. The FIFO serves to manage data flow, preventing errors associated with reading extensive datasets. The RTL schematic of the SRAM memory block, depicted in Fig. 13, likely illustrates the intricate interconnections and data pathways within the memory unit. This SRAM design addresses the need for efficient data retrieval and storage in a multi-port system, showcasing adaptability and problem-solving capabilities in the context of large-scale data operations. The integration of a FIFO mechanism reflects a thoughtful approach to enhancing the memory unit's functionality, ensuring robust performance in handling diverse data access patterns.

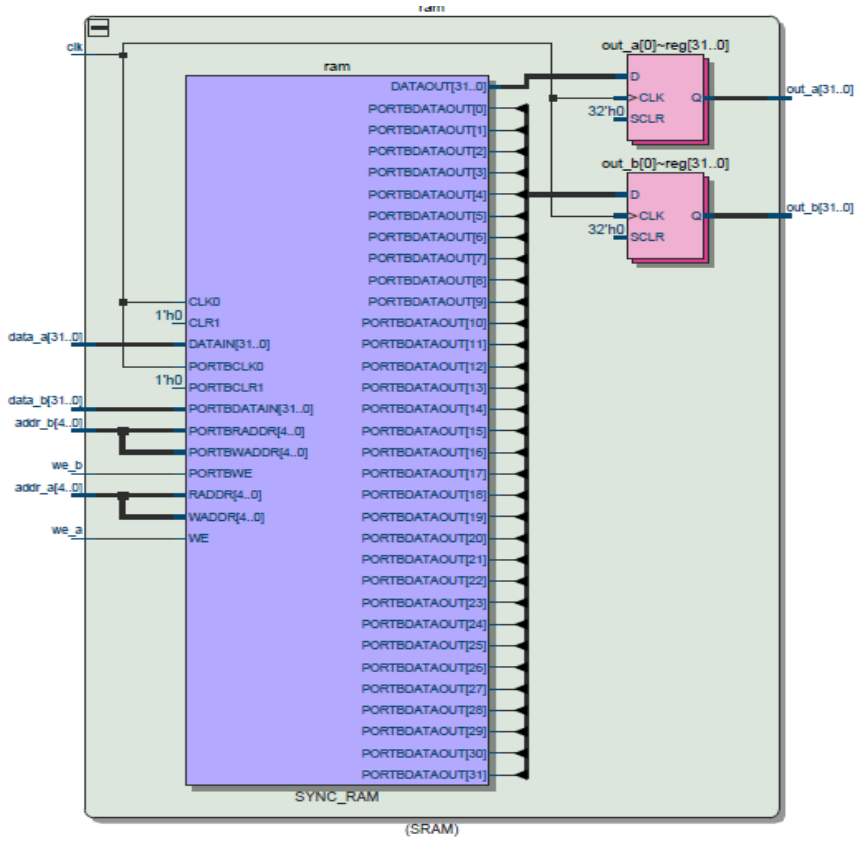

Fig. 13. The RTL schematic of SRAM memory block.

4. FIFO

The FIFO (First In, First Out) block, a widely used digital component, operates as a stack, storing data in a specific sequence and retrieving it in the same order but at different speeds. This functionality proves crucial in handling 25 pixels of data read from memory, organizing them two by two. The FIFO's inherent sequencing capability ensures the orderly retrieval of data, accommodating various processing rates. Interfacing with memory, it efficiently manages the flow of data, preventing errors associated with reading a large number of pixels at once. The FIFO's adaptability allows it to handle diverse data access patterns within a digital system. The RTL schematic, likely depicted in Fig. 14, showcases the intricate connections and logic governing how data is stacked, sequenced, and transferred to upcoming modules. The stacked data, comprising 25 pixels, can be simultaneously used as input for subsequent processing modules, optimizing overall system performance. Acting as a buffer, the FIFO regulates data flow between components, preventing bottlenecks and enhancing data processing efficiency. Its design addresses the need for organized data retrieval and storage, reflecting a thoughtful approach to digital system architecture.

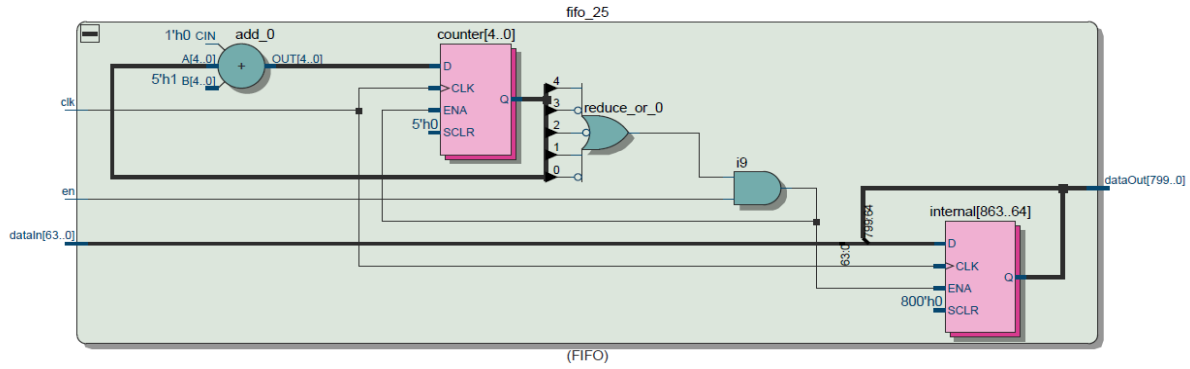

Fig. 14. The RTL schematic of FIFO block

5. Input provider

The input provider block processes input pixels, presenting them to the system in a 5 x 5 format. The block likely organizes and formats the input data for optimal utilization within the system. The behavioral simulation, illustrated in Fig. 15, provides a dynamic representation of the block's functionality during operation. The RTL schematic, depicted in Fig. 16, is a visual representation of the internal logic and connections governing the input provider block's operation. The block plays a crucial role in structuring and delivering input data in a coherent 5 x 5 arrangement. The simulation waveform in Fig. 15 captures the dynamic behavior of the input provider block during its operation. The RTL schematic in Fig. 16 likely details the interconnections and data pathways within the block, offering insight into its internal structure. Overall, the input provider block serves as a key component in the system, ensuring the efficient presentation of input pixels in a 5 x 5 format.

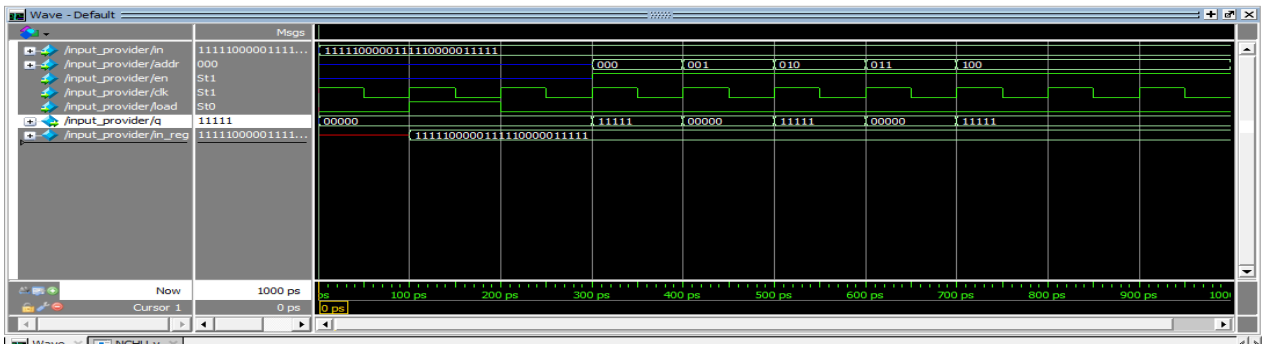

Fig. 15. Behavioral simulation of input

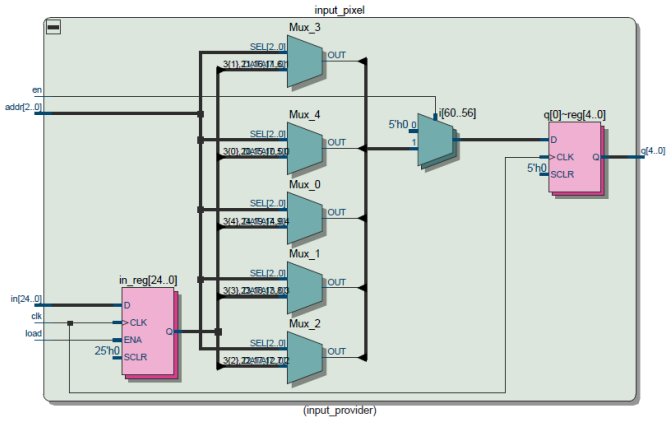

Fig. 16. The RTL schematic of input provider block.

6.Weight memory

The weight provider module reads weights from a file, supplying them to the system in a structured 25 x 25 format for each layer of the neural network. These manually generated weights are tailored for a pattern recognition application, aiming to trigger the Spiking Neural Network (SNN) to fire when the final output surpasses a specified threshold of 0.98. The careful manual crafting of weights suggests a deliberate approach to enhance the network's accuracy in pattern recognition tasks. The 25 x 25 arrangement reflects the connectivity and specificity of the weights within each neural network layer. By setting a threshold of 0.98, the weight provider module establishes a criterion for network activation, contributing to fine-tuned and precise pattern recognition capabilities. The use of a file for weight storage allows for flexibility in updating and modifying weights as needed for evolving recognition requirements. Overall, this weight provider module plays a pivotal role in configuring the network for effective pattern recognition based on the specified threshold and manually crafted weights.

1	0000_0000_0000_0000_0000_0001_0100_1000	w11	0	0.005
2	0000_0000_0000_0000_0001_1001_1001_1010	w21	1	0.1
3	0000_0000_0000_0000_0001_1001_1001_1010	w31	1	0.1
4	0000_0000_0000_0000_0001_1001_1001_1010	w41	1	0.1
5	0000_0000_0000_0000_0000_0001_0100_1000	w51	0	0.005
6	0000_0000_0000_0000_0001_1001_1001_1010	w61	1	0.1
7	0000_0000_0000_0000_0000_0001_0100_1000	w71	0	0.005
8	0000_0000_0000_0000_0000_0001_0100_1000	w81	0	0.005
9	0000_0000_0000_0000_0000_0001_0100_1000	w91	0	0.005
10	0000_0000_0000_0000_0001_1001_1001_1010	w101	1	0.1
11	0000_0000_0000_0000_0001_1001_1001_1010	w111	1	0.1
12	0000_0000_0000_0000_0000_0001_0100_1000	w121	0	0.005
13	0000_0000_0000_0000_0000_0001_0100_1000	131	0	0.005
14	0000_0000_0000_0000_0000_0001_0100_1000	w141	0	0.005
15	0000_0000_0000_0000_0001_1001_1001_1010	151	1	0.1
16	0000_0000_0000_0000_0001_1001_1001_1010	161	1	0.1
17	0000_0000_0000_0000_0000_0001_0100_1000	171	0	0.005
18	0000_0000_0000_0000_0000_0001_0100_1000	181	0	0.005
19	0000_0000_0000_0000_0000_0001_0100_1000	191	0	0.005
20	0000_0000_0000_0000_0001_0111_0000_1010	201	1	0.09
21	0000_0000_0000_0000_0000_0001_0100_1000	211	0	0.005
22	0000_0000_0000_0000_0000_0111_1010_1110	221	1	0.03
23	0000_0000_0000_0000_0000_0111_1010_1110	231	1	0.03
24	0000_0000_0000_0000_0000_1010_0011_1101	24	1	0.04
25	0000_0000_0000_0000_0000_0001_0100_1000	251	0	0.005

Fig. 17 weights used for the pattern recognition

7.SNN module

The SNN module, depicted in Fig. 11, is designed for pattern recognition applications according to the paper. The Table I likely provides a comprehensive overview of device utilization, detailing how resources are allocated within the proposed SNN. This utilization summary offers insights into the efficiency and resource optimization of the neural network implementation. Additionally, the timing summary in Fig. 18 outlines the

temporal aspects of the overall architecture, showcasing the speed and synchronization of different components. The proposed SNN demonstrates a balance between functionality and resource efficiency, as indicated by the device utilization summary. The timing summary in Fig. 18 provides a visual representation of the system's temporal characteristics, ensuring proper coordination and synchronization in processing. The architectural design reflects a thoughtful consideration of both computational and timing aspects, essential for effective pattern recognition applications.

Device Utilization Summary (estimated values)			[-1]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	24763	301440	8%
Number of Slice LUTs	8108	150720	5%
Number of fully used LUT-FF pairs	3268	29603	11%
Number of bonded IOBs	59	600	9%
Number of Block RAM/FIFO	1	416	0%
Number of BUFG/BUFGCTRLs	1	32	3%
Number of DSP48E1s	520	768	67%

Table. I Device Utilization Summary of the

```
Timing Summary:
---------------
Speed Grade: -1

   Minimum period: 3.553ns (Maximum Frequency: 281.442MHz)
   Minimum input arrival time before clock: 1.915ns
   Maximum output required time after clock: 0.777ns
   Maximum combinational path delay: No path found
```

Fig. 18 Timing summary of the SNN proposed

8.FSM

The control unit, implemented as a Finite State Machine (FSM) with 67 states, orchestrates the entire process for recognizing the 'O' pattern. The main flow begins with the initiation of the code, followed by fetching input from the input provider. Layer one computations unfold across five main stages, each comprising six states. Subsequently, data is read from memory to the FIFO buffer, involving 16 states. Layer two computations then proceed over 17 states, followed by another data read from RAM to FIFO in six states. Finally, layer three computations conclude with three states. The simulation waveform in Figure 19 captures the dynamic behavior of the system during the recognition of the correct 'O' pattern. This includes the intricate interplay of states within the FSM, transitions between stages, and the flow of data through the system. The waveform offers a visual representation of the temporal aspects of the control unit's operation, providing insights into the synchronization and coordination of various computational stages during the recognition process.

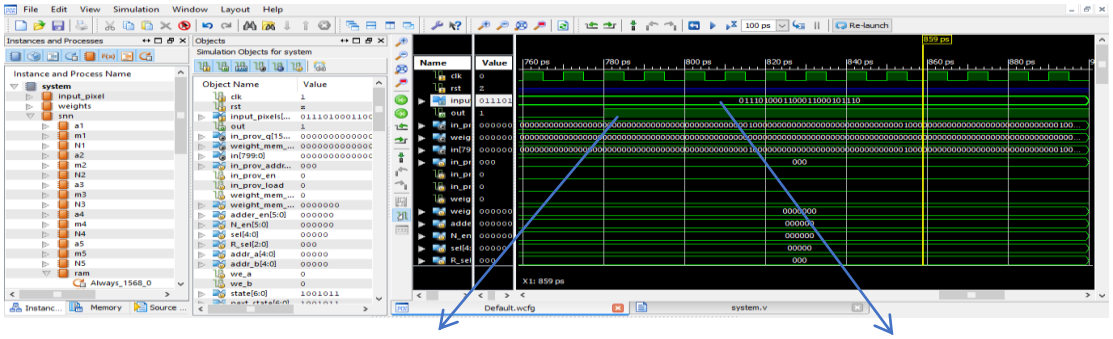

Fig. 19. The simulation waveform for recognition of correct input 'O' pattern.

The following Fig. 20.(a),(b) shows simulation waveforms for recognition of incorrect input 'O' patterns.

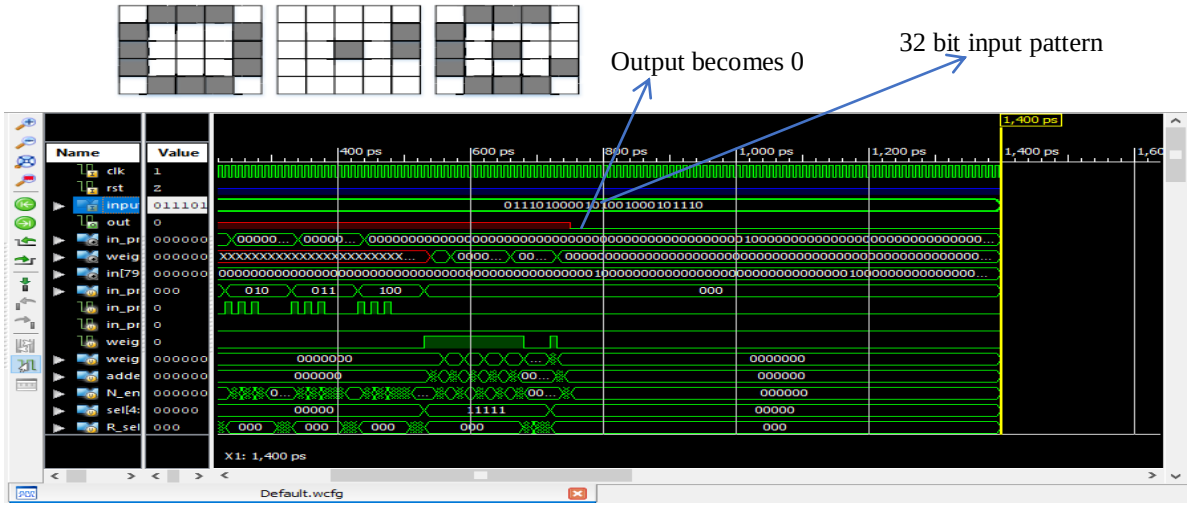

Fig. 20. (a) The simulation waveforms for recognition of incorrect input 'O' Pattern-I.

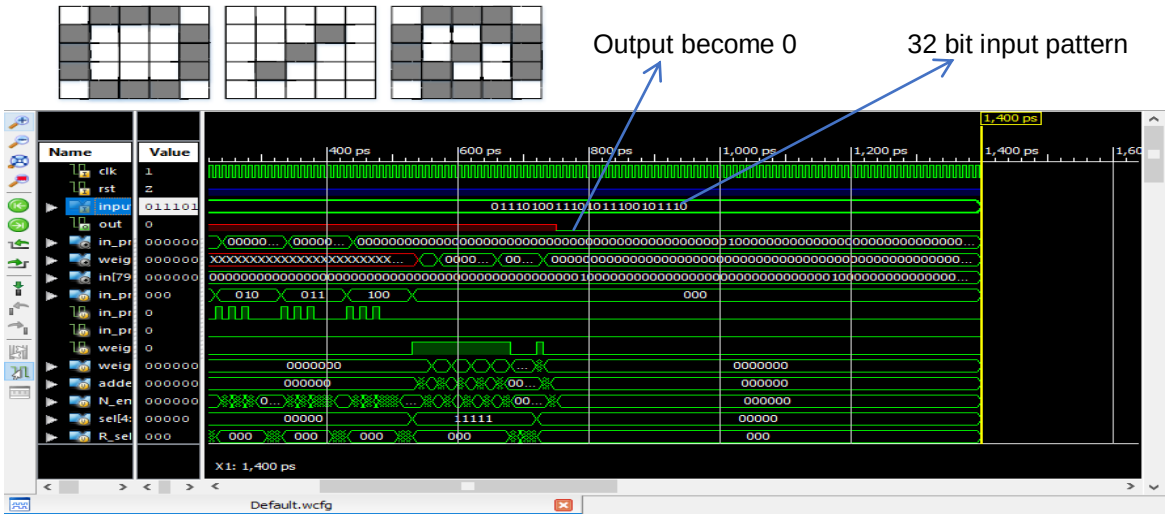

Fig. 20. (b). The simulation waveforms for recognition of incorrect input 'O' pattern-II.

The following Table. II presents the comparison between the synthesis results of the suggested SNN and previously published works.

Reference	Slice Registers		Slice LUTs		Max. speed (MHz)	Device
	Number	Utilization	Number	Utilization		
[17]	1023	0%	11339	7%	189.071	Virtex-6 ML605
This work	24763	8%	8108	5%	281.442	Virtex-6 ML605

Table. II Comparison between the synthesis results of the suggested SNN and previously published works.

7.Conclusion

This research introduces a multiplier-less Modified Leaky Integrate and Fire Neuron Unit (MLIFNU) and an adapted Spiking Neural Network (SNN) architecture tailored for Pattern Recognition. The FPGA synthesis

results reveal a significant improvement, particularly in terms of slice registers and slice LUTs, compared to a previous study. The MLIFNU and SNN achieve impressive maximum frequencies of 361.991 MHz and 281.442 MHz, respectively, highlighting the substantial speed enhancement offered by the proposed model.

This research underscores the potential of the MLIFNU and the optimized SNN architecture, demonstrating their efficiency in achieving higher processing speeds. The comparison with prior work emphasizes the success of the proposed modifications, with a notable increase in maximum frequencies. The multiplier-less approach in the MLIFNU contributes to computational efficiency, a critical factor in neural network implementations.

However, the discussion also acknowledges the need for further research to address current limitations in SNNs. Future efforts should focus on developing more effective models, algorithms, and architectures to unlock the full potential of neuromorphic implementations. This research sets the stage for the creation of innovative systems across scientific sectors, emphasizing the broader impact of advancing SNN technologies.

In summary, the research presents a novel MLIFNU and an optimized SNN architecture, showcasing remarkable improvements in FPGA synthesis results. The study highlights the need for ongoing research to overcome current limitations and foster the development of more effective neuromorphic models and systems across scientific domains.

8. References

[1]. S. S. Chowdhury, C. Lee, and K. Roy, “Towards Understanding the Effect of Leak in Spiking Neural Networks”. arXiv, 2020. doi: 10.48550/ARXIV.2006.08761.

[2]. X. Chen, T. Yajima, I. H. Inoue, and T. Iizuka, “An ultra-compact leaky integrate-and-fire neuron with long and tunable time constant utilizing pseudo resistors for spiking neural networks,” Japanese Journal of Applied Physics, vol. 61, no. SC, p. SC1051, Feb. 2022, doi: 10.35848/1347-4065/ac43e4.

[3]. I. Jaras, T. Harada, M. E. Orchard, P. E. Maldonado, and R. C. Vergara, “Extending the integrate- and- fire model to account for metabolic dependencies,” European Journal of Neuroscience, vol. 54, no. 4, pp. 5249–5260, 2021.

[4]. S. R. Nandakumar, I. Boybat, M. Le Gallo, E. Eleftheriou, A. Sebastian, and B. Rajendran, “Experimental demonstration of supervised learning in spiking neural networks with phase-change memory synapses,” Scientific Reports, vol. 10, no. 1, 2020.

[5]. T. Van Pottelbergh, G. Drion, and R. Sepulchre, “From Biophysical to Integrate-and-Fire Modeling,” Neural Computation, vol. 33, no. 3, pp. 563–589, Mar. 2021, doi: 10.1162/neco_a_01353.

[6]. W. H. Brigner et al., “Three Artificial Spintronic Leaky Integrate-and-Fire Neurons,” SPIN, vol. 10, no. 2, p. 2040003, 2020, doi: 10.1142/S2010324720400032.

[7]. S. Dutta, V. Kumar, A. Shukla, N. R. Mohapatra, and U. Ganguly, “Leaky integrate and fire neuron by charge-discharge dynamics in floating-body MOSFET,” Scientific Reports, vol. 7, no. 1, 2017.

[8].S. Cavallari, S. Panzeri, and A. Mazzoni, “Comparison of the dynamics of neural interactions between current-based and conductance-based integrate-and-fire recurrent networks,” Frontiers in Neural Circuits, vol. 8, 2014, doi: 10.3389/fncir.2014.00012.

[9].A. F. Murray and A. V. W. Smith, "Asynchronous VLSI neural networks using pulse-stream arithmetic," in IEEE Journal of Solid-State Circuits, vol. 23, no. 3, pp. 688-697, June 1988, doi: 10.1109/4.307.

[10].F. Blayo and P. Hurat, “A VLSI Systolic Array Dedicated to Hopfield Neural Network,” in VLSI for Artificial Intelligence, J. G. Delgado-Frias and W. R. Moore, Eds. Boston, MA: Springer US, 1989, pp. 255–264. doi: 10.1007/978-1-4613-1619-0_24.

[11].P. H. W. Leong and M. A. Jabri, "A VLSI neural network for morphology classification," [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, 1992, pp. 678-683 vol.2, doi: 10.1109/IJCNN.1992.226909.

[12].G. Cairns and L. Tarassenko, "Learning with analogue VLSP MLPs," Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems, 1994, pp. 67-76, doi: 10.1109/ICMNN.1994.593184.

[13].E. M. Izhikevich, "Which model to use for cortical spiking neurons?," in IEEE Transactions on Neural Networks, vol. 15, no. 5, pp. 1063-1070, Sept. 2004, doi: 10.1109/TNN.2004.832719.

[14].Christophe, F., Mikkonen, T., Andalibi, V., Koskimies, K., &Laukkarinen, T. (2015, October). "Pattern recognition with spiking neural networks: a simple training method". In SPLST (pp. 296-308).

[15].Song, S., Miller, K. & Abbott, L. "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity." Nat Neurosci 3, 919–926 (2000). https://doi.org/10.1038/78829

[16].XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices

[17].E. Z. Farsa, A. Ahmadi, M. A. Maleki, M. Gholami and H. N. Rad, "A Low-Cost High-Speed Neuromorphic Hardware Based on Spiking Neural Network," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 66, no. 9, pp. 1582-1586, Sept. 2019, doi: 10.1109/TCSII.2019.2890846.

[18].E. M. Izhikevich, "Which model to use for cortical spiking neurons?," in IEEE Transactions on Neural Networks, vol. 15, no. 5, pp. 1063-1070, Sept. 2004, doi: 10.1109/TNN.2004.832719.

[19].Christophe, François &Mikkonen, Tommi&Andalibi, Vafa&Koskimies, Kai &Laukkarinen, Teemu. (2015)."Pattern recognition with Spiking Neural Networks: a simple training method," in Proc. 14th Symp. Program. Lang. Softw.Tools. (SPLST), Tampere, Finland, Oct. 2015, pp. 296–308.

[20]. H. Soleimani, A. Ahmadi and M. Bavandpour, "Biologically Inspired Spiking Neurons: Piecewise Linear Models and Digital Implementation," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 59, no. 12, pp. 2991-3004, Dec. 2012, doi: 10.1109/TCSI.2012.2206463.

[21].M. Heidarpur, A. Ahmadi, M. Ahmadi and M. RahimiAzghadi, "CORDIC-SNN: On-FPGA STDP Learning WithIzhikevich Neurons," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 7, pp. 2651-2661, July 2019, doi: 10.1109/TCSI.2019.2899356.

[22].Adineh-vand, A., G. Karimi and M. Khazaei. "Digital Implementation of a Spiking Convolutional Neural Network for Tumor Detection." (2020) in Journal of Microelectronics, Electronic Components and Materials Vol. 49, No. 4(2019), 193 – 201

[23].Huayaney F.L.M., Tanaka H., Matsuo T., Morie T., Aihara K. (2011) "A VLSI Spiking Neural Network with Symmetric STDP and Associative Memory Operation." In: Lu BL., Zhang L., Kwok J. (eds) Neural Information Processing. ICONIP 2011. Lecture Notes in Computer Science, vol 7064. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24965-5_43

[24].M. Nouri, M. Jalilian, M. Hayati and D. Abbott, "A Digital Neuromorphic Realization of Pair-Based and Triplet-Based Spike-Timing-Dependent Synaptic Plasticity," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 6, pp. 804-808, June 2018, doi: 10.1109/TCSII.2017.2750214.

[25].J. -s. Seo et al., "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," 2011 IEEE Custom Integrated Circuits Conference (CICC), 2011, pp. 1-4, doi: 10.1109/CICC.2011.6055293.

[26].Camuñas-Mesa LA, Linares-Barranco B, Serrano-Gotarredona T. "Neuromorphic Spiking Neural Networks and Their Memristor-CMOS Hardware Implementations." Materials (Basel). 2019;12(17):2745. Published 2019 Aug 27. doi:10.3390/ma12172745

[27].Diehl, Peter &Zarrella, Guido & Cassidy, Andrew &Pedroni, Bruno &Neftci, Emre. (2016). "Conversion of Artificial Recurrent Neural Networks to Spiking Neural Networks for Low-power Neuromorphic Hardware." 1-8. 10.1109/ICRC.2016.7738691.

[28].Schuman, Catherine &Potok, Thomas & Patton, Robert & Birdwell, J. & Dean, Mark & Rose, Garrett & Plank, James. (2017). "A Survey of Neuromorphic Computing and Neural Networks in Hardware."

[29].W. Gerstner and W. M. Kistler, "Spiking Neuron Models: Single Neurons,Populations, Plasticity." Cambridge, U.K.: Cambridge Univ. Press, 2002.