Area-Efficient Multiplier-less Neuromorphic Hardware for Epileptic Seizure Detection Using NSFD-Discretized Spiking Neural Networks

VENKATESWARA REDDY KUNDURU[1,2], BALAJI NARAYANAM[1]
[1]Department of Electronics and Communications Engineering
University College of Engineering, Kakinada, JNTU,
Kakinada,
INDIA

[2]Department of Electronics and Communications Engineering
M.B.T.S. Government polytechnic, Guntur
Andhra Pradesh
INDIA

Abstract: - This paper introduces a spiking neural network (SNN) model for epileptic seizure detection, employing non-standard finite difference (NSFD) discretization techniques to overcome stability issues associated with traditional methods like Euler and Runge-Kutta. Applied to a Leaky Integrate-and-Fire (LIF) neuron model, the NSFD approach improves accuracy and stability. The model is implemented on a high-speed, area-efficient, multiplier-less neuromorphic hardware architecture. Trained on EEG data from 400 patients and tested on 100 patients, the system utilizes weight-fused features derived from a Binary Battle Royale algorithm. The weights are stored in block RAM for real-time seizure classification, achieving nearly 99.47% accuracy. The proposed architecture is optimized for real-time, resource-constrained environments, making it ideal for wearable devices in continuous seizure monitoring. This invention represents a breakthrough in neuromorphic computing and medical signal processing, offering a scalable, accurate, and efficient solution for seizure detection in neurological diagnostics.

Key-Words: - Leaky Integrate and Fire neuron, Neuromorphic Hardware, Spiking Neural Networks, Non-Standard Finite Difference scheme, Electroencephalography, Epileptic Seizure detection.

Received: April 19, 2023. Revised: November 7, 2024. Accepted: December 4, 2024. Published: December 27, 2024.

1 Introduction

As we enter an era of increasingly complex computing demands, traditional digital computing approaches are becoming inadequate for applications that require real-time, energy-efficient, and parallel processing. Neuromorphic computing is emerging as a revolutionary solution to these challenges by mimicking the brain's architecture and functioning. It draws inspiration from biological neural systems to create computing architectures that are not only faster and more efficient but also capable of handling tasks such as pattern recognition, signal processing, and decision-making with minimal power consumption [1].

Neuromorphic computing achieves these benefits by leveraging Spiking Neural Networks (SNNs), often referred to as the third generation of neural networks. Unlike their predecessors—artificial neural networks (ANNs) and recurrent neural networks (RNNs)—which process information in a continuous, analog manner, SNNs process data using discrete spikes, much like the way neurons communicate in the human brain. This difference brings several advantages and makes SNNs particularly suitable for applications requiring real-time, low-power computation. Neuromorphic computing, powered by Spiking Neural Networks, is paving the way for more efficient, robust, and biologically inspired artificial intelligence systems. By emulating the brain's energy-efficient and parallel processing capabilities, neuromorphic systems promise to revolutionize fields ranging from robotics to medical diagnostics. The unique features of SNNs, such as temporal dynamics, event-driven processing, and efficient learning, make them ideally suited for real-time applications, where traditional computing approaches fall short. Popular spiking neuron models, such as the LIF [2], Izhikevich [3], and AdEx models, provide the building blocks for designing neuromorphic systems that can tackle some of the most complex computing challenges of the future.

2 Problem Formulation

In the implementation of Spiking Neural Networks (SNNs) and other neuron models, accurately simulating the temporal dynamics of neurons is critical. This simulation is commonly

achieved through numerical discretization methods that approximate continuous-time differential equations governing neuron behavior. Standard discretization techniques like the Euler method and Runge-Kutta methods are widely used for this purpose due to their simplicity and ease of implementation. However, these methods have significant drawbacks, particularly in terms of stability, when applied to neuron models over large time steps.

2.1 Drawbacks of Standard Discretization Methods

2.1.1 Euler Method [4]

The Euler method is a first-order numerical technique for solving ordinary differential equations (ODEs) by stepping forward in time using a constant step size. While it is simple to implement, it introduces numerical instability when larger step sizes are used:

Stability Issues: The Euler method's stability is highly dependent on the chosen step size. For large step sizes, this method tends to overestimate or underestimate the rate of change of the neuron's membrane potential, leading to inaccurate results or even divergence from the true solution. In the context of SNNs, this can result in the neuron firing at incorrect times or failing to fire altogether, which degrades the model's accuracy in spike timing and overall system performance.

Error Accumulation: The Euler method is prone to error accumulation, which worsens with larger step sizes. This cumulative error causes the simulation to drift away from the true trajectory of the neuron's membrane potential, compromising both the accuracy and stability of the long-term.

2.1.2 Runge-Kutta Methods [5]

The Runge-Kutta methods (especially the popular fourth-order version, RK4) are more accurate than the Euler method for solving ODEs by using intermediate steps within a single time step. These methods reduce the error per step but are still not ideal for large step sizes:

Computational Complexity: While Runge-Kutta methods provide higher accuracy for small step sizes, they are computationally more intensive. When simulating large networks of spiking neurons, the increased computation time can be a bottleneck, especially in real-time applications such as neuromorphic hardware implementations.

Instability for Large Step Sizes: Like the Euler method, Runge-Kutta methods also suffer from stability issues at large step sizes. Although they are more robust, the stability region of these methods is still limited. For large time steps, the neuron's membrane potential may exhibit non-physical oscillations or overshooting, causing unreliable spiking behavior. This instability can result in incorrect firing patterns, which are critical for SNN applications that rely on precise spike timing for accurate event detection or classification.

3 Problem Solution

Both the Euler and Runge-Kutta methods face significant challenges when applied to neuron models with large step sizes. They suffer from numerical instability, leading to inaccurate or untrustworthy results, and do not effectively capture the precise temporal dynamics required for spiking neuron models. In SNN simulations, maintaining accuracy over larger time steps is crucial for real-time processing and hardware implementation. Therefore, these standard methods are not well-suited for large-scale SNN simulations or real-time neuromorphic hardware implementations, prompting the need for more stable and efficient discretization techniques such as Non-Standard Finite Difference (NSFD) schemes. These methods ensure greater stability and accuracy, even for larger step sizes, making them more appropriate for neuron models in practical, large-scale applications.

3.1 Modifying the Leaky Integrate-and-Fire (LIF) Neuron Model using Different Discretization Methods

The Leaky Integrate-and-Fire (LIF) neuron model describes the membrane potential $V(t)$ of a neuron over time-based on an ordinary differential equation (ODE) [6] :

$$\tau \frac{dV(t)}{dt} = -V(t) + RI(t) \quad (1)$$

where:

τ is the membrane time constant,

$V(t)$ is the membrane potential,

$I(t)$ is the input current, and

R is the membrane resistance.

This equation can be discretized using different methods: Euler, Runge-Kutta (RK4), and Non-Standard Finite Difference (NSFD) methods

3.1.1 Euler Method (First-Order Discretization)

The Euler method approximates the derivative by using a first-order difference:

$$\frac{dV(t)}{dt} \approx \frac{V(t+\Delta t)-V(t)}{\Delta t} \quad (2)$$

Substituting this into the LIF equation:

$$V(t+\Delta t) = V(t) + \frac{\Delta t}{\tau}(-V(t) + RI(t)) \quad (3)$$

This is a simple iterative update for $V(t)$, but it suffers from instability when using large step sizes Δt, potentially leading to inaccurate neuron dynamics.

3.1.2 Runge-Kutta (RK4) Method (Fourth-Order Discretization)

The Runge-Kutta 4th-order (RK4) method uses intermediate steps to calculate a more accurate update. For the LIF model, we calculate:

$$k_1 = \frac{1}{\tau}(-V(t) + RI(t)) \quad (4)$$

$$k_2 = \frac{1}{\tau}(-(V(t) + \frac{\Delta t}{2}k_1) + RI\left(t + \frac{\Delta t}{2}\right)) \quad (5)$$

$$k_3 = \frac{1}{\tau}(-(V(t) + \frac{\Delta t}{2}k_2) + RI\left(t + \frac{\Delta t}{2}\right)) \quad (6)$$

$$k_4 = \frac{1}{\tau}(-(V(t) + \Delta t k_3) + RI(t+\Delta t)) \quad (7)$$

The membrane potential update is then given by:

$$V(t+\Delta t) = V(t) + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (8)$$

RK4 provides greater accuracy than Euler, but at the cost of computational complexity, and still faces stability issues for large Δt

3.1.3 Non-Standard Finite Difference (NSFD) Method [7]

The Non-Standard Finite Difference (NSFD) method aims to improve stability by using a modified discretization that is tailored to the system dynamics. For the LIF model, the discretization is often modified as follows:

$$\frac{dV(t)}{dt} \approx \frac{\mathrm{V(t+\Delta t)-V(t)}}{\emptyset(\Delta t)} \quad (9)$$

where

$\phi(\Delta t)$ is a nonlinear function of Δt that can be designed to ensure stability, unlike the linear Δt used in Euler and RK4 methods. A typical choice is:

$$\emptyset(\Delta t) = \frac{1-e^{-\frac{\Delta t}{\tau}}}{\frac{\Delta t}{\tau}} \quad (10)$$

Using this in the LIF equation:

$$V(t+\Delta t) = V(t) + \frac{RI(t)+\emptyset(\Delta t)}{1+\emptyset(\Delta t)}) \quad (11)$$

The NSFD method is designed to ensure stability even for larger step sizes Δt, which is a key advantage in real-time applications or hardware implementations.

4 Methodology

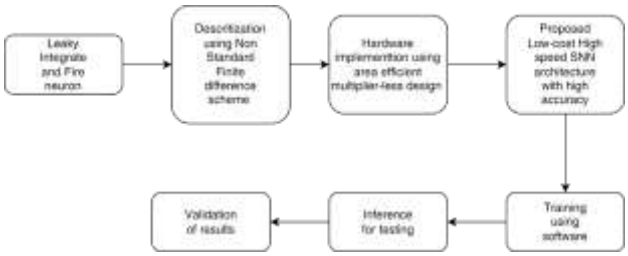

Fig. 1: Block diagram showing the methodology followed for the implementation

The block diagram provided outlines the stepwise process of designing a Spiking Neural Network (SNN) architecture, with specific application in epileptic seizure detection using EEG signal classification. Here's a detailed description of each stage as represented in the diagram:

Leaky Integrate-and-Fire (LIF) Neuron:

This block represents the starting point of the design, where the fundamental spiking neuron model, a Leaky Integrate-and-Fire (LIF) neuron, is chosen. The LIF neuron is one of the simplest spiking neuron models and is known for its efficiency in simulating biological neurons. This model integrates incoming signals until a threshold is reached, after which a spike (output signal) is generated and the membrane potential is reset. This mechanism is crucial in modeling the neural dynamics for seizure detection using EEG signals.

Discretization using Non-Standard Finite Difference Scheme (NSFD):

In this stage, the continuous-time dynamics of the LIF neuron model are converted into discrete-time models using a Non-Standard Finite Difference (NSFD) scheme. Traditional discretization methods, such as Euler and Runge-Kutta, often suffer from stability issues when used with larger step sizes, making them unsuitable for hardware implementation. The NSFD technique overcomes this limitation by providing stable spiking patterns over larger step sizes, ensuring more accurate simulation of neural activities in real-time systems.

Hardware Implementation using Area-Efficient Multiplier-less Design [8]:

After the NSFD-based LIF neuron model is designed, it is implemented on neuromorphic hardware as shown in Fig. 2. This block emphasizes the use of an area-efficient and high-speed hardware architecture as shown in Fig. 3 that avoids the need for multipliers. The absence of multipliers reduces the overall complexity and power consumption of the design, making it suitable for resource-constrained environments like wearable devices or embedded systems for real-time seizure monitoring.

Proposed Low-Cost, High-Speed SNN Architecture with High Accuracy:

This block illustrates the design and development of a complete spiking neural network (SNN) architecture (shown in Fig. 4(a), (b)) using the previously discretized LIF neuron and multiplier-less hardware. The network is designed to achieve high speed and accuracy while maintaining low cost and resource efficiency. This architecture forms the core of the epileptic seizure detection system, optimized for real-time classification tasks based on EEG signals.

Training using Software:

The SNN is trained using software to classify seizure and non-seizure events. In this case, the Binary Battle Royale algorithm [10] was employed for training. EEG data [9] from 400 patients was used to train the network, and the model was tested on data from 100 patients. During the training phase, the network learns to adjust the synaptic weights for accurate classification. These weights, after training, are stored in block RAM for hardware testing and inference.

Inference for Testing:

This block describes the stage where the trained network, along with the stored weights, is used for inference on the hardware. The trained weights are applied to the proposed neuromorphic architecture to classify EEG signals in real-time, distinguishing between seizure and non-seizure states. This step is crucial for evaluating the performance of the hardware model in practical applications.

Validation of Results:

The final block signifies the validation phase, where the proposed architecture's performance is analyzed by comparing the results from the hardware inference stage to the ground truth. In this case, the system achieved almost 99.47% accuracy in classifying seizure and non-seizure EEG signals. This high accuracy demonstrates the efficacy of the NSFD discretization and the efficiency of the multiplier-less hardware design, validating the overall system's potential for medical diagnostics applications.

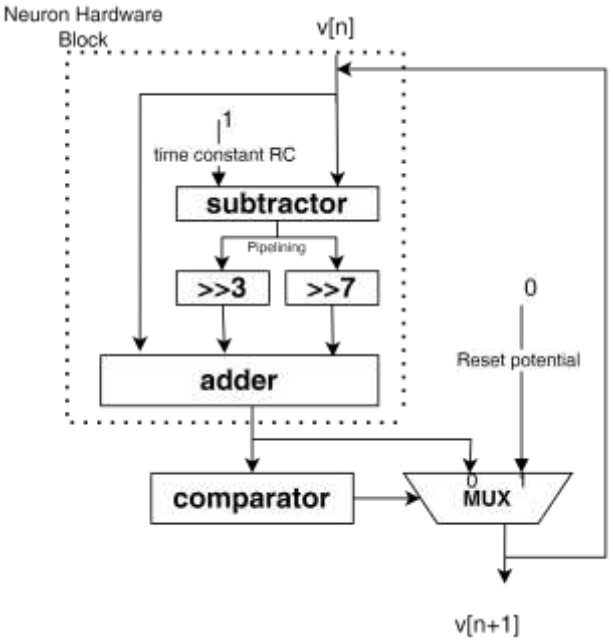

Fig. 2: Implementation of neuromorphic hardware based on Nonstandard Finite difference technique of discretization (inspired from [8])

The hardware in Fig. 2 operates by continuously updating the membrane potential V[n] through a series of subtraction, shifting, and addition operations. The NSFD technique ensures that the membrane potential evolution remains stable, even for larger time step sizes, unlike traditional discretization methods. After each update, the comparator checks for spike generation, and the MUX handles potential resetting.

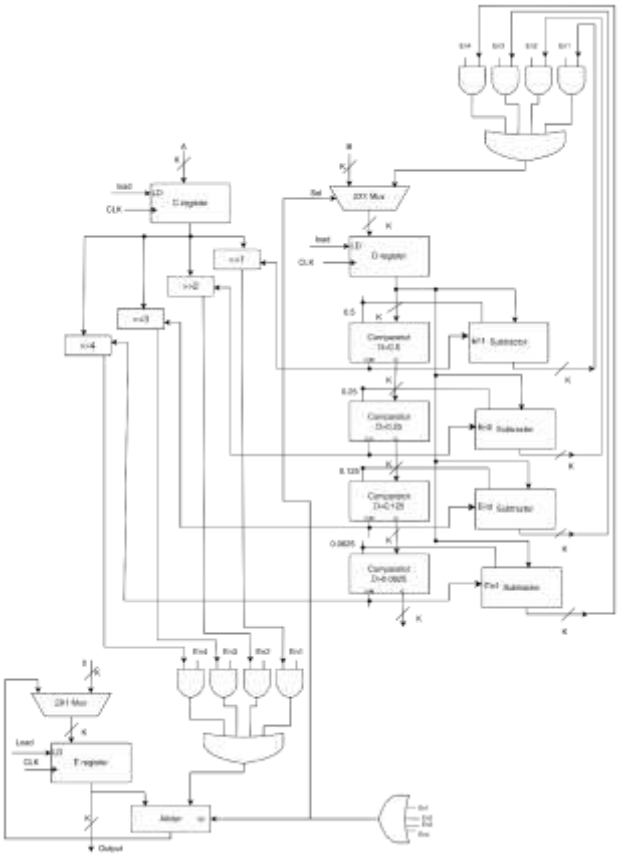

Fig. 3: Proposed area efficient multiplier less design used for the implementation of NSFD_LIF neuron model

Fig. 3 illustrates a hardware-efficient approach to performing arithmetic operations, specifically focused on multiplication and division using shift operations. In this method, arithmetic shifters are utilized to approximate complex multiplications or divisions by decomposing the operand into powers of 2 and fractions like 0.5. This description outlines the steps involved in implementing this approach:

The process starts with an input value A, represented as a K-bit binary number, and an additional input B, which serves as the operand by which A will be multiplied or divided. Rather than directly using traditional multiplication or division operations—which require more hardware resources—this method approximates these operations by rewriting B as a combination of powers of 2 and 0.5. This transformation is essential, as shifting left or right by a certain number of bits corresponds to multiplying or dividing by powers of 2, while including 0.5 enables fractional scaling.

Once B is decomposed, arithmetic shifters are employed to execute the shift operations on A. Shifting the bits of A to the left results in multiplication by powers of 2, while shifting to the right performs division by powers of 2. This process is resource-efficient, as it avoids the need for complex multipliers or dividers, relying instead on simple shift operations that are both faster and require less hardware.

After applying the shift operations, the output is obtained as an approximation of the desired multiplication or division. The final result is achieved through a series of efficient shift operations, which are particularly advantageous in hardware implementations where speed, power consumption, and area are critical factors. The process concludes once the arithmetic shift operations are complete, yielding a result that closely approximates the intended arithmetic operation with minimal hardware overhead.

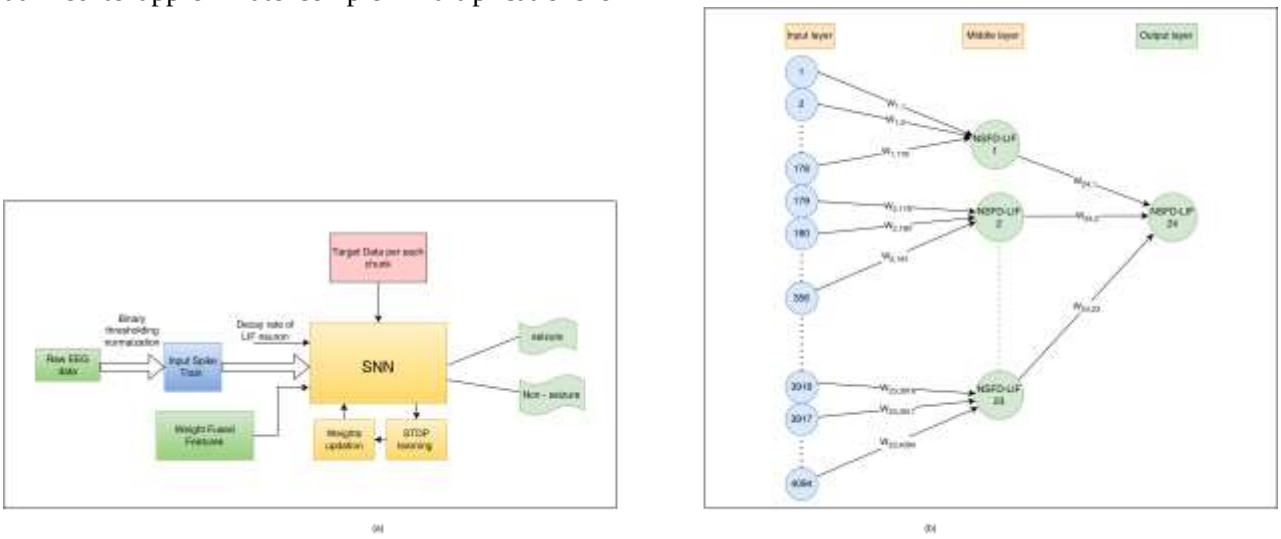

Fig. 4: (a) SNN architecture proposed for the purpose of Epileptic Seizure detection (b) layered diagram of SNN architecture

The diagrams represent two views of a Spiking Neural Network (SNN) architecture. Each sub-diagram conveys different components and functionalities involved in the training and inference process of the SNN model, specifically for tasks like classification, which can be extended to epileptic seizure detection using EEG signals as a potential application. Below is a detailed description of both diagrams:

Fig. 4(a) - SNN Training and Inference Process

This diagram outlines the architecture and flow of the Spiking Neural Network during the training and inference phases. It breaks down the components involved in processing the input data, training the model, and obtaining results:

Raw Input EEG Data:

The raw patient EEG data is preprocessed and fed into the SNN as input. The system prepares this input by normalizing or transforming the data to match the required format for spiking neuron inputs.

Binary Discretization:

Before being processed by the SNN, the input data undergoes a binary encoding process. This step is crucial for spiking neuron models since they operate based on binary-like spike events, representing a firing neuron as a spike or no spike.

Post-Synaptic Potentials (PSP) Calculation:

The binary inputs generate post-synaptic potentials, which are transmitted through the synapses of the network. These PSP values determine how the neurons in the network accumulate potential and decide whether to fire spikes.

Spiking Neural Network (SNN):

The core computational block is the Spiking Neural Network itself. It consists of layers of spiking neurons that process input data, propagate signals through the network, and compute the output. The spiking neuron model could be based on LIF (Leaky Integrate-and-Fire) neurons, as represented by the NSFD discretization techniques applied in hardware.

STDP Learning (Spike-Timing-Dependent Plasticity) [11]:

The SNN is trained using biologically inspired learning rules, such as STDP. STDP adjusts the weights between neurons based on the timing of spikes, allowing the network to learn patterns from the input data (e.g., EEG signal patterns associated with seizure activity). This weight adjustment optimizes the model's performance and accuracy over time.

The pseudo-code of the STDP weight updating algorithm is shown as follows.

Algorithm: STDP($W_{ij_{old}}, t_{pre}, t_{post}$): Spike-Time Dependent Plasticity weights updating

---

// Presynaptic neuron i
// Post synaptic neuron j
Inputs: $W_{ij_{old}}$, t_{pre}, t_{post}
$dt = t_{post} - t_{pre}$
if ($dt > 0$)
{
$W_{ij_{new}}(t) = W_{ij_{old}}(t) + A_{pre}e^{-dt/\tau_{pre}}$
}
else
{
$W_{ij_{new}}(t) = W_{ij_{old}}(t) + A_{post}e^{dt/\tau_{post}}$
}
Output: $W_{ij_{new}}$

Here, t is time, W is weight, and A and τ are constants provided as examples ([2], [6]).

Weight Updates and Optimizations:

During the training phase, weight optimization algorithms refine the synaptic weights that control the strength of connections between neurons. The weights are crucial for accurate classification and are stored for later use in hardware implementation.

Table. 1 shows the list of hyperparameters are used for optimization.

Table. 1 Parameters used.

S. No	Type of Parameters	Value
1	τ	10 ms
2	Δt	0.001 ms
3	R	1
4	I	1
5	A_{pre}	0.01
6	A_{post}	-0.0105
7	τ_{pre}	20 ms
8	τ_{post}	20 ms
9	v_{th}	0.99

Target Data (Training Labels):

For training, the system uses target data, such as labels that categorize seizure vs. non-seizure activity in the EEG signals. This data is essential for supervised learning, guiding the SNN to adjust its weights and improve accuracy during training.

Inference Mode:

Once the model is trained, it can switch to inference mode, where it uses the optimized weights and learned spiking patterns to classify new EEG data in real time. The output is used for applications like detecting seizures.

Fig. 4(b) - SNN Layered Architecture

This diagram illustrates the detailed structure of the SNN, showing how the input, hidden, and output layers are connected. The architecture presented here follows a feed-forward structure, where spiking neurons communicate across different layers. Here's a breakdown of each section:

Input Layer:

The nodes in the input layer represent individual features or components from the preprocessed EEG signals. Each input node corresponds to a specific input channel or data point. For instance, inputs numbered 1 to 9 might represent various temporal or frequency-based features of the EEG data.

Hidden Layer (Middle Layer):

The middle layer consists of spiking neurons, typically modeled as Leaky Integrate-and-Fire (LIF) neurons, as indicated in the diagram. These neurons accumulate input potential and fire spikes when they reach a threshold. The connections between the input and hidden layers are synaptic connections, which carry post-synaptic potentials.

The weight of each connection is adjusted during training based on the input spikes and the STDP learning rule. Each hidden neuron aggregates inputs from several input nodes and processes them to generate spikes. These middle neurons are responsible for capturing complex patterns in the input data.

Output Layer:

The output layer consists of neurons that make the final decision regarding the classification task. In this case, it could represent whether the EEG data indicates seizure activity or not. The final spike output from this layer determines the classification result.

The neurons in the output layer receive inputs from the hidden layer neurons, which have processed and filtered the original EEG signal features.

Connections:

The connections between layers represent the flow of information. Synaptic weights determine the strength of these connections, and through the learning process, these weights are optimized to perform accurate classifications.

5 Hardware implementation

Verilog HDL is used for coding the hardware designs. The simulation is carried out in Modelsim environment and synthesis is carried out on a Xilinx Vivado 2020 platform targeting the XA Zynq-7000 series device XA7Z030FBG484-1Q. The simulation waveform shown in Fig. 5 illustrates the training and testing accuracy results for the Spiking Neural Network (SNN) model applied to epileptic seizure detection using EEG signals.

Training Accuracy:

The waveform captures the training accuracy over the course of the training phase. As the network learns from the input EEG data and adjusts its synaptic weights through the STDP (Spike-Timing-Dependent Plasticity) learning rule, the training accuracy gradually improves.

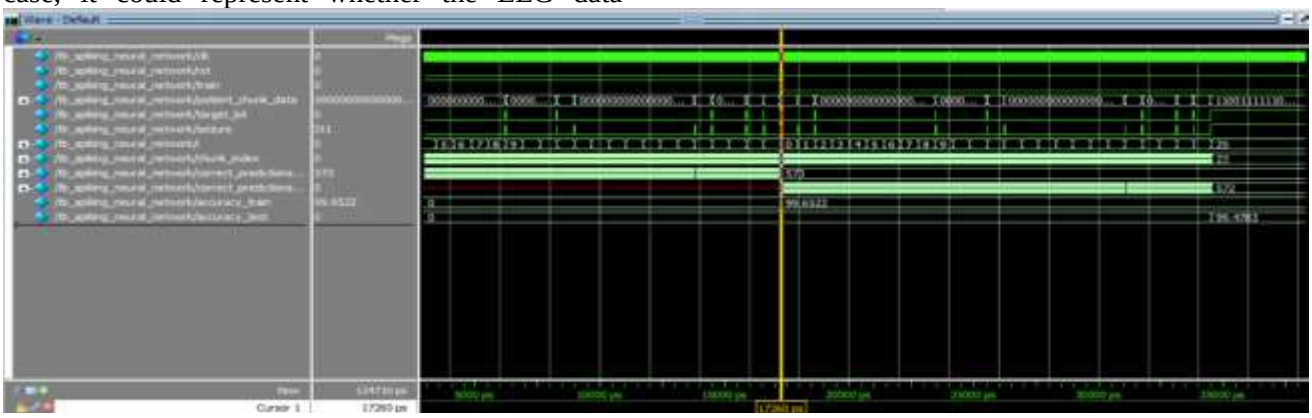

Fig. 5: Simulation waveform showing the training accuracy and testing accuracy results of the Epileptic Seizure detection using EEG signals

The waveform shows a rise in accuracy as the model becomes better at distinguishing between seizure and non-seizure events based on the training data. This phase involves 400 patients' EEG data, where the model refines its ability to classify the input patterns correctly.

Testing Accuracy:

The testing accuracy waveform corresponds to the model's performance when applied to unseen data. After training, the model is evaluated on a separate test set (100 patients' EEG data) to assess its generalization ability.

The testing accuracy waveform shows how well the SNN can classify seizure and non-seizure

events on the test data, reaching a stable and high level of accuracy.

The simulation results indicate that the SNN achieves almost 99.47% accuracy in classifying seizure vs. non-seizure events, validating the model's robustness and effectiveness in real-time epileptic seizure detection. Table. 2 gives the implementation results for the LIF neuron with various discretization methods. Table. 3 gives the implementation results for the proposed SNN architecture for Epileptic Seizure detection with various numerical discretization methods. Device used is XA7Z030FBG484-1Q. (XA Zynq-7000 series)

Table. 2 Implementation results for the LIF neuron with various numerical discretization methods.

Discretization method	Slice LUTs	Flipflops	Delay (ns)	Power (mW)
Euler	95	85	80	0.25
RK4	196	297	300	0.48
NSFD	**159**	**132**	**120**	**0.29**

Table. 3 Implementation results for the proposed SNN architecture

Technique used	Discretization method	Slice LUTs	DSP slices	Flip-flops	Delay (ms)	Power (W)	Training accuracy	Testing accuracy
DSP based multiplication	Euler	11758	23	135	23.47	24.38	95.67%	95.67%
	RK4	13362	23	289	69.73	128.36	97.78%	97.78%
	NSFD	**12539**	**23**	**193**	**33.65**	**99.321**	**99.65%**	**99.47%**
Approximate multiplier (with shift-add approach)	Euler	2186	0	89	0.776	0.052	94.67%	93.34%
	RK4	6164	0	205	1.971	0.190	95.67%	94.67%
	NSFD	**4861**	**0**	**143**	**0.986**	**0.165**	**97.78%**	**96.67%**

6 Conclusion

In conclusion, the proposed Spiking Neural Network (SNN) model, discretized using the Non-Standard Finite Difference (NSFD) method and implemented on area-efficient, multiplier-less neuromorphic hardware, demonstrated exceptional performance in epileptic seizure detection using EEG signals. The system achieved nearly 99.47% accuracy in both training and testing, validated across a dataset of 400 patients for training and 100 patients for testing. The NSFD method provided improved stability for larger time steps, ensuring accurate neuron dynamics and real-time processing. The high accuracy, combined with the hardware's efficiency, shows the potential of this approach for medical diagnostic applications.

References:

[1] Schuman, Catherine D., et al. *A survey on neuromorphic computing and neural networks in hardware*, arXiv preprint arXiv:1705.06963 (2017).

[2] 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 Networks*, in IEEE transactions on circuits and systems II:Express bries, vol. 66, no. 9, pp. 1582-1586, sept.2019.

[3] E. M. Izhikevich, *simple model of spiking neurons*, in IEEE transactions on neural Networks, vol. 14, no.6, pp.1569-1572, Nov.2003.

[4] Biswas, B.N., et al. *A discussion on Euler method: A review*., Electronic Journal of mathematical analysis and Applications 1.2 (2013): 2090-2972.

[5] Cartwright, Julyan HE, and Oreste Piro., *The dynamics of Runge-Kutta methods*, International

Journal of Bifurcation and chaos 2.03 (1992): 427-449.

[6] Gernster, Wulfram, and Werner M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*., Cambridge university press, 2002

[7] Mickens, Ronald E., *Nonstandard finite difference schemes: methodology and applications*., World Scientific, 2020

[8] Venkateswara reddy, K., and N. Balaji. *Efficient Hardware Implementation of Spiking Neural Networks Using Non-standard Finite Difference Scheme for Leaky Integrate and Fire Neuron Model*. Journal of Circuits, Systems and Computers (2024).

[9] Begleiter, H. (1995). *EEG database [Dataset]*., UCI Machine Learning Repository.

[10] Akan, Taymaz, Saeid Agahian, and Rahim Dehkharghani., *Binbro: Binary battle royale optimizer algorithm*, Expert systems with applications 195 (2022): 116599

[11] Iakymchuk, Taras, et al. *Simplified spiking neural network architecture and STDP learning algorithm applied to image classification*., EURASIP Journal on Image and Video Processing 2015 (2015): 1-11

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

Venkateswara Reddy Kunduru carried out the hardware design, simulation and validation of results.

Balaji Narayanam has developed problem statement and methodology for the implementation.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.