JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

SPIKING NEURAL NETWORK ACCELERATOR: IN THE FIELD OF BRAIN MACHINE INTERFACE

[1]Aaditya Balakrishna, [2]Harshitha Jampala, [3]Harshith P, [4]Hriday Sachdev, [5]Sujatha K

[1]UG Student, [2]UG Student, [3]UG Student, [4]UG Student,[5]Associate Professor
Department of ECE,
BMSCE, Bengaluru, India

Abstract: Brain-machine interfaces (BMIs) represent a transformative technology enabling direct interaction between the human brain and external devices. Among various BMI approaches, Spiking Neural Networks (SNN) are prominent for their efficiency in mimicking the brain's spiking behavior. This paper presents the design and implementation of an advanced hardware architecture capable of executing SNN computations integrated with Electroencephalogram (EEG) acquisition systems on a Field Programmable Gate Array (FPGA). Data is first pre-processed into arrays for feature extraction using four layers. The model is trained in software, storing weights and parameters, which are then used to create a hardware model and generate a bitstream file. A Python overlay connects the software and hardware, allowing output simulations for accuracy calculations.

Index Terms **– Brain Machine Interface, Electroencephalogram, Field Programmable Gate Array, Spiking Neural Network.**

I. INTRODUCTION

Disability, as described by the WHO World Report on Disability, presents multifaceted challenges that impact millions worldwide. Among those significantly affected are individuals reliant on wheelchairs for mobility. In this paper, we delve into the realm of Spiking Neural Network (SNN) accelerators, exploring their potential applications in the field of Brain-Machine Interfaces (BMI) to enhance the lives of individuals with mobility impairments. Epidemiological data reveals the staggering prevalence of disability globally, with estimates suggesting that approximately 15% of the world's population grapples with various forms of impairment. Within this demographic, a substantial portion—around 131.8 million individuals—requires wheelchairs for mobility, emphasizing the urgent need for innovative solutions to improve their quality of life.

Our focus extends to those affected by conditions such as cerebral palsy, multiple sclerosis, spinal cord injury (SCI), and stroke, which are among the leading causes of mobility impairment. Tetraplegia, characterized by paralysis of the arms, legs, and torso, exemplifies the profound challenges faced by individuals in need of constant care and assistance with daily tasks. Moreover, sudden-onset SCI presents tragic consequences, with an estimated 1.5 million individuals in India alone grappling with its long-term impact. Despite advancements in medical care, rehabilitation efforts often fall short, leaving many without adequate support systems.[1]

In parallel, individuals with severe physical impairments resulting from traumatic brain injury (TBI), muscular dystrophy, or amyotrophic lateral sclerosis (ALS) require continuous care, posing significant burdens on caregivers and financial constraints on affected individuals. Navigating the complexities of disability leads us to explore assistive technology (AT), which aims to enhance the functional abilities and independence of individuals with disabilities. In the context of BMI, AT holds the promise of facilitating direct communication between the brain and external devices, offering newfound avenues for mobility and participation in society.

Decoding electroencephalography (EEG) signals accurately and reliably is pivotal for the functionality of brain-computer interfaces (BCI), which find diverse applications from neurorehabilitation to controlling industrial and mobile robots. However, for practical real-world deployment, BCI systems must be portable and operate on limited energy resources.

Deep neural networks (DNNs) have proven effective in decoding EEG signals for various tasks due to their ability to learn features from raw representations with strong generalization. Yet, the high energy cost associated with DNN inference limits their suitability for portable BCIs. Enter spiking neural networks (SNNs), an emerging brain-inspired architecture known for its energy efficiency. SNNs compute asynchronously through discrete events, or spikes, and their hardware realization on neuromorphic chips has bolstered their applicability in energy-efficient learning tasks.

Moreover, evidence suggests that SNNs may outperform DNNs for signals with non-stationary characteristics, owing to their ability to represent information at multiple timescales through dynamic spike-based computations. However, current SNN-based EEG classification methods often fall short of matching the performance of state-of-the-art DNN methods, limiting their practical applications.[2]

This performance gap can be attributed to several factors. First, common methods for encoding EEG into spikes rely on heuristics and require extensive task-independent tuning of encoding parameters, affecting classification performance. Second, current approaches often utilize reservoirs of randomly connected recurrent neurons, lacking consideration of important EEG priors

such as spatial and temporal dependencies. Third, while local correlation-based learning rules are popular in training SNNs, they lack the expressiveness of error-driven training algorithms like backpropagation, restricting network topologies to shallow architectures and impacting interpretability.[3]

Addressing these challenges necessitates a comprehensive rethinking of SNN design, encompassing input encoding, network topology, training methods, and interpretability. Developing an accurate and reliable SNN solution for EEG classification requires innovative approaches that bridge the gap between neuroscience-inspired computing and practical application domains.

II. THERORETICAL FRAMEWORK

In this work, we present the hardware implementation of SNN on a Python Productivity for Zynq (PYNQ) FPGA board [4][5] for the classification of motion tasks. Figure 1 shows the SNN implementation, where the software model is first trained, and then the weights and parameters are prepared for deployment on the FPGA board. The PYNQ board, chosen for its ease of use with Python codes and high-level abstraction, allows for straightforward validation and testing. After the code is deployed, we propose a novel approach to visualize the output classes, aiding in result interpretation. Time, power, and energy consumption metrics are recorded for evaluation. Subsequently, hardware acceleration techniques are employed to optimize the board's performance. Through a comparison table, we demonstrate the efficacy of the accelerator in enhancing neural network computations.

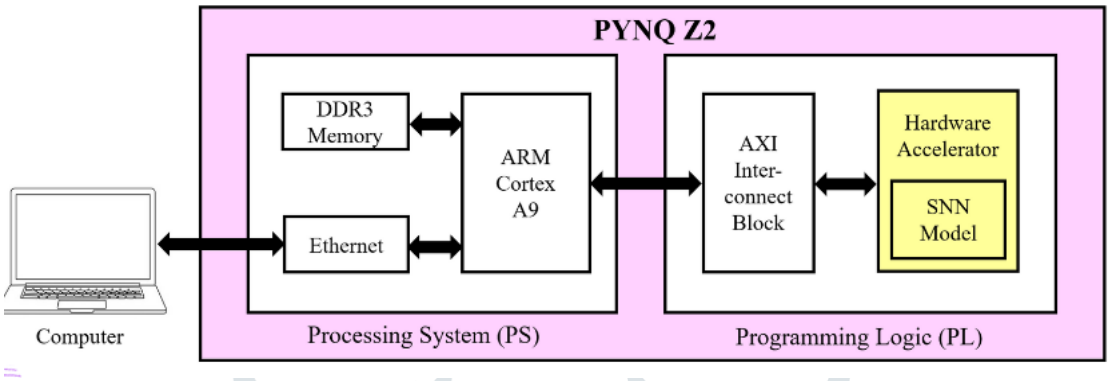

Fig1. Block diagram of proposed solution.

2.1 Leaky Integrate-and-fire Model

The leaky integrate-and-fire (LIF) model [5][6] is a simplified representation of a biological neuron used in computational neuroscience. It describes how a neuron integrates incoming signals and generates spikes. In this model, the neuron's membrane potential increases as it receives input currents. Figure 2 shows the potential decay over time due to a "leaky" term, simulating the natural loss of charge in a biological membrane. The operation of the LIF neuron model with a constant leak is described by five basic operations: 1. synaptic integration, 2. leak integration, 3. Threshold, 4. spike firing, and 5. Reset.

$$\mathrm{Vj(t)} = \mathrm{Vj(t-1)} + \sum_{i=0}^{N-1} xi(t)\, si \qquad (1)$$

$$Vj(t) = Vj(t) - \lambda j \qquad (2)$$

$$if \quad Vj(t) \geq \alpha \mathrm{j} \qquad (3)$$

spike

$$\mathrm{Vj(t)} = \mathrm{Rj} \qquad (4)$$

endif

For the j^{th} neuron in the i^{th} timestep of the Eq.1, the membrane potential $V_j(t)$ is the sum of the membrane potential in the previous timestep Vj(t - 1) and the synaptic input. For each of the N synapses, the synaptic input is the sum of the spike input to the synapse $x_i(t)$ at the current timestep, multiplied by the signed synaptic weight s_i. Following integration, the LIF neuron model subtracts the leak value j from the membrane potential as mentioned in Eq.2. With a linear leak, this constant is subtracted every timestep, regardless of membrane potential or synaptic activity. This operation serves as a constant bias on the neural dynamics. Then, the LIF neuron model compares the membrane potential at the current timestep $V_j(t)$ in the Eq.3 and Eq.4 with the neuron threshold j. If the membrane potential is greater than or equal to the threshold voltage, the neuron fires a spike and resets its membrane potential. In the typical case, the reset voltage R_j is zero.

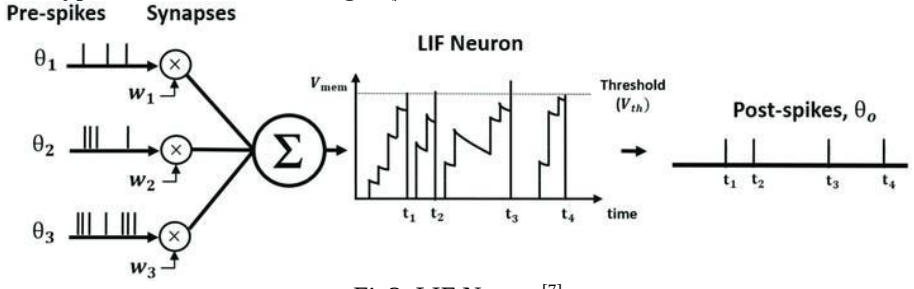

Fig2. LIF Neuron[7]

When the membrane potential reaches a certain threshold, the neuron "fires" a spike, and the potential is reset to a lower value or resting state. The LIF model is particularly valued for its simplicity and computational efficiency, making it suitable for large-scale neural network simulations and hardware implementations. This model captures essential features of neural dynamics while being less complex than more detailed biophysical models.

III. METHODOLOGY

3.1 Software Implementation

3.1.1 Pre-processing

Pre-processing is a critical step in the analysis of electroencephalogram (EEG) data. Figure 3 is the proposed process, where raw EEG signals undergo subsequent analysis by removing noise, artifacts, and irrelevant data.

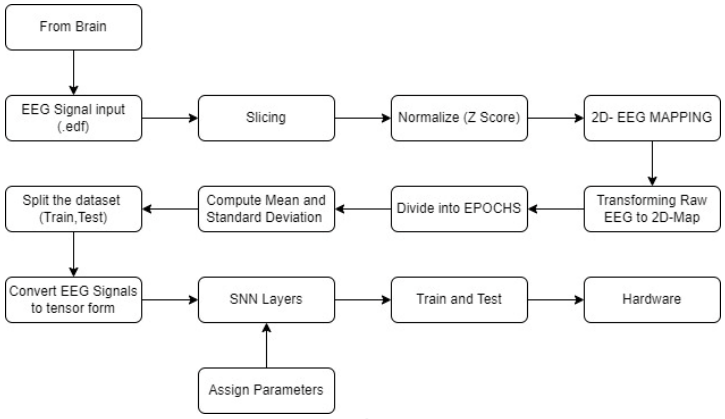

Fig3. Preprocessing Flowchart

The EEGMMIDB dataset [8] consists of raw EEG recordings stored in EDF (European Data Format) files. Each file contains data from various motor imagery tasks performed by different subjects. The raw data is first loaded using the MNE library, and to remove low-frequency drifts and high- frequency noise, a band-pass filter with cut-off frequencies set at 0.1 Hz and 79 Hz is applied. This filtering step ensures that only the frequency components associated with motor imagery are retained.

The next step involves detecting events within the EEG recordings. Events correspond to specific experimental stimuli or actions are segmenting the continuous EEG signal into distinct trials. The events are extracted using annotations embedded in the EDF files. Once the events are identified, the raw EEG data is segmented into epochs based on the time intervals between consecutive events. Each epoch represents a single trial of the motor imagery task, capturing the brain activity corresponding to the initiation and execution of the imagined movement. Next, normalization of EEG signals are done to compare sugnals across different trials. The mean and standard deviation for each EEG channel is computed across the entire dataset. Subsequently, a z-score normalization is performed on the segmented epochs. This step transforms the data such that each channel has a mean of 0 and a standard deviation of 1.

To leverage the spatial relationships between EEG channels, the normalized 1D EEG data is transformed into a 2D format. A pre-defined mapping of EEG channels to a 2D grid is employed, simulating the physical arrangement of electrodes on the scalp. This 2D transformation is particularly beneficial for convolutional spiking neural networks, which can exploit spatial patterns in the data for improved classification performance.

The segmented epochs are further divided into smaller overlapping windows to enhance temporal resolution. Each epoch is subdivided into windows of fixed length, with a certain degree of overlap between consecutive windows. This approach not only increases the amount of training data but also captures the dynamic aspects of motor imagery, thereby improving the temporal precision of the classification model.

In the EEG IMDB dataset, certain motor imagery tasks may be underrepresented, leading to class imbalance. To address this issue, epochs corresponding to different classes are carefully balanced during the data pre-processing stage. Techniques such as data augmentation, and over-sampling of minority classes are considered. Data augmentation techniques such as adding Gaussian noise or randomly shifting the data are applied. Additionally, artifacts resulting from eye blinks, muscle movements, and other non-neural sources are identified and removed using Independent Component Analysis (ICA) or other artifact correction methods.

3.1.2 SNN Architecture

The architecture of the spiking neural network (SNN) implemented in this work is shown in the Fig.4 which consists of multiple specialized layers, each designed to process and propagate spiking activity through the network. A fundamental component of the SNN is the mechanism that simulates the behavior of biological neurons by determining whether a neuron should "spike" based on its membrane potential surpassing a defined threshold. During the forward pass, this mechanism outputs a spike if the input exceeds the threshold. In the backward pass, it computes a pseudo-gradient within a specified gradient window, facilitating the training of the network through backpropagation. The amplitude (AMP) for the gradient is set to 0.3. An extension of this mechanism incorporates dropout—a regularization technique that randomly sets a fraction of inputs to zero during training. The dropout rates used are 0.3 for temporal convolutional layer, recurrent layers, and fully connected layers. This dropout mechanism helps prevent overfitting by ensuring that the network does not become overly reliant on specific neurons.[9]

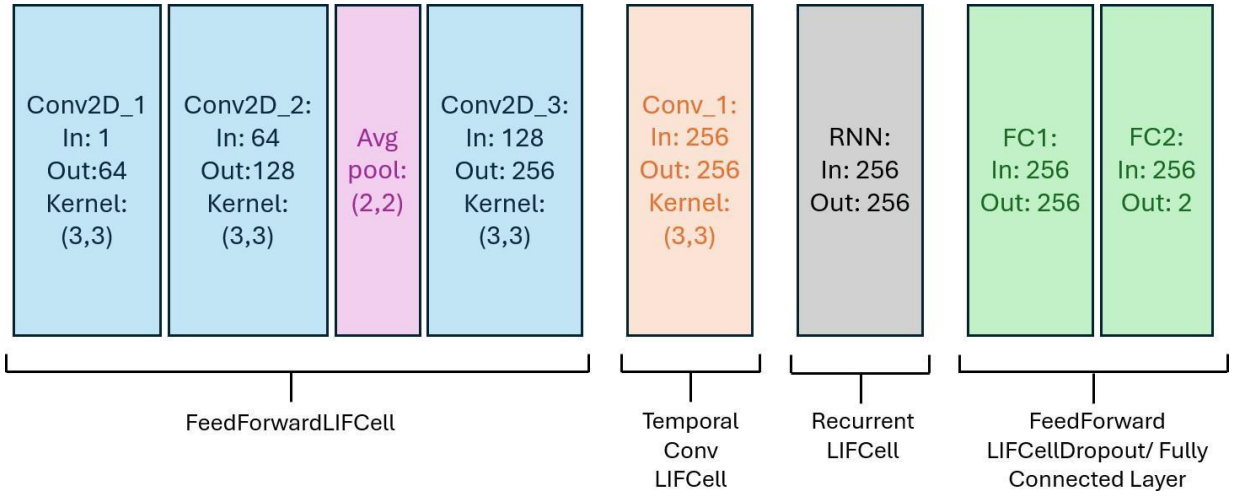

Fig4. SNN Architecture

The network includes several feed forward LIF cells. These cells integrate incoming spikes over time, gradually leaking potential unless reset by a spike. Each cell receives an input, updates its membrane potential, and generates an output spike based on the threshold mechanism. Key parameters include decay constants for current and voltage, spike threshold, and gradient window. The threshold voltage is set to a specific value for each layer, and the gradient window is used to define the range for pseudo-gradient calculation.

Incorporating temporal dependencies, the recurrent LIF cells extend the functionality of feed- forward LIF cells with recurrent connections. These connections allow the cell to maintain a memory of past spikes, enabling the network to capture temporal patterns in the input data. The recurrent cell integrates spikes from both the current input and previous time steps, updating its state accordingly. The output spike is generated using the threshold and dropout mechanisms, ensuring regularization during training. The recurrent layer includes parameters for threshold adaptation, which dynamically adjusts the spike threshold based on recent activity. Parameters specific to this layer include the threshold adaptation amplitude and decay, with a base threshold.

Temporal convolutional cells perform temporal convolution operations across multiple time steps. These cells utilize a kernel size to convolve input sequences, integrating information over a specified temporal window. Each temporal convolutional cell is equipped with multiple synaptic pathways, each modeled by a linear transformation, which collectively contributes to the current integration. The pseudo-gradient and dropout mechanisms are also applied in these cells, ensuring robust learning and mitigating overfitting. Temporal convolutional cells are particularly effective in capturing short-term temporal dependencies in the EEG signals.

The architecture includes several convolutional layers implemented using feed forward LIF cells. These layers apply spatial convolutions to the input data, extracting hierarchical features. For example, the first convolutional layer uses a decay constant for voltage of 1 for 64 channels over an 8x9 spatial grid. Each convolutional layer is followed by an average pooling operation to reduce the spatial dimensions and aggregate information. This hierarchical processing is essential for capturing spatial patterns in the EEG data.

Towards the output, the network employs fully connected layers to integrate the features extracted by previous layers. The feed forward LIF cells with dropout in this stage ensure comprehensive integration of information while applying dropout for regularization. The final layer produces the network's output, which is a classification decision based on the integrated spiking activity. For instance, the first fully connected layer uses a dropout rate of 0.3.

Each component in the SNN architecture is designed to emulate key aspects of biological neural processing, enhancing the network's ability to learn and generalize from complex temporal and spatial patterns in the input data. The use of dropout and pseudo-gradient functions ensures that the network remains trainable and generalizable, ultimately leading to improved performance in motor imagery classification tasks. By using trained weights and parameters, the hardware implementation is employed. The dataset used in our project includes samples with the following distribution across classes: initial samples per class were [2324, 2311, 0, 0, 0] and [2097, 2088, 0, 0, 0] for training, with a training sample distribution of [2097, 2088, 0, 0, 0]. For validation, the samples per class were [227, 223, 0, 0, 0].

The initial sample distribution was heavily skewed, requiring careful monitoring of the model's performance across epochs. Throughout the training, we observed a gradual reduction in loss and an overall improvement in validation accuracy. Initially, the model exhibited significant variability in class-specific accuracy, indicating challenges in learning balanced representations. However, as training progressed, the model adapted, resulting in more consistent accuracy rates across both classes. By the end of the training period, the neural network achieved a substantial increase in validation accuracy and a notable decrease in loss, demonstrating effective learning and robust classification capabilities. This iterative training process highlighted the model's ability to refine its predictions and achieve a balanced performance despite the initial data imbalance.

3.2 Hardware Implementation

For the hardware implementation of the proposed solution is mentioned in the below Fig.5, the Vivado 19.1 software environment was utilized. Vivado 19.1 provides a comprehensive suite of tools for FPGA (Field-Programmable Gate Array) design, including simulation, synthesis, implementation, and bitstream generation functionalities. Leveraging the capabilities of Vivado 19.1 facilitated the development and optimization of the hardware architecture to meet the requirements of the intended application.

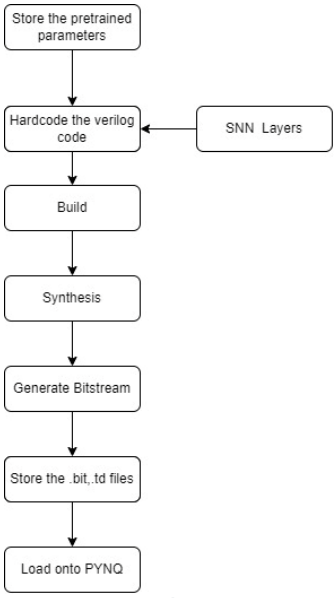

Fig5. Hardware Implementation Flowchart

3.2.1 RTL Designing

The hardware implementation of the Spiking Neural Network (SNN) accelerator for a brain-machine interface is designed using a Zynq-7000 SoC, which integrates both a processing system (PS) and programmable logic (PL) to achieve efficient neural computation and interfacing. The core of the implementation is the ZYNQ7 Processing System, featuring an ARM Cortex-A9 processor responsible for high-level control, configuration, and management of data flow within the system are shown in the Fig.6. Key interfaces of the processing system include the DDR interface for external memory access, the FIXED_IO interface for essential input/output operations, and AXI interfaces that facilitate communication with the programmable logic via an AXI interconnect.

The AXI Interconnect (ps7_0_axi_periph) serves as a bridge between the processing system and various peripherals implemented in the programmable logic, managing multiple AXI interfaces to ensure efficient data transfer and resource sharing. It connects master interfaces (M00_AXI, M01_AXI, etc.) to various peripheral components to initiate transactions and allows peripheral components to respond to AXI transactions through slave interfaces (S00_AXI).

The programmable logic houses several peripheral components critical for the SNN accelerator's functionality. These include IN_SPIKE_1 and IN_SPIKE_2, which are input interfaces for spike data, essential for feeding neural signals into the SNN accelerator, utilizing AXI GPIO interfaces for communication. The OUTPUT_CLASS block is responsible for outputting classification results derived from the SNN accelerator, also using an AXI GPIO interface for data transfer. The SNN_Accelerator_0 is the core neural computation unit, processing input spikes and generating output spikes. It interfaces with both the input spike modules and the output classifier, operating with key signals such as a clock (clk) to synchronize operations, a reset signal (rst_n) for initialization, input spikes (input_spike[1:0]), and output spikes (output_spike[7:0]).

The OUTPUT_CLASSIFIER_0 component classifies the output spikes generated by the SNN accelerator. It directly interfaces with the SNN accelerator’s output and provides classified results via an AXI GPIO interface. The system clock (CLOCK) drives the synchronous operations of the entire system, while reset signals (rst_ps7_0_50M, etc.) ensure proper initialization and reliable operation of all components. The DDR and Fixed I/O interfaces connect to the processing system, enabling external memory access and necessary I/O operations to support data-intensive tasks and external communication.

This hardware implementation leverages the Zynq-7000's versatile architecture to create an efficient and scalable SNN accelerator for brain-machine interface applications. By combining the processing power of the ARM Cortex-A9 with custom neural network processing logic in the programmable logic, the system achieves the real-time performance essential for neural computation and interfacing.

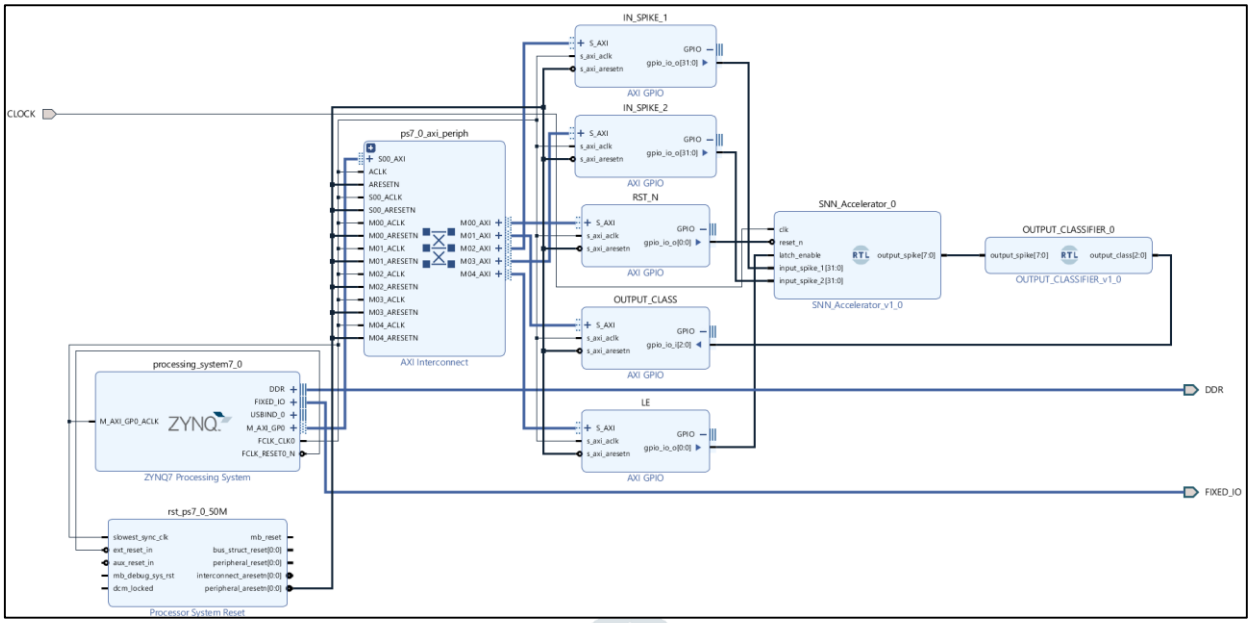

Fig6. The Block Design with the SNN Accelerator RTL Module

In the hardware implementation of the Spiking Neural Network (SNN) accelerator for a brain-machine interface, managing data input and output efficiently is crucial due to the constraints imposed by the GPIO interface and the need to maintain high processing performance. The GPIO can handle only 32 bits at a time, but the SNN accelerator requires 64-bit inputs for effective neural computation. Directly managing 64-bit inputs would significantly increase fabric resource utilization and processor load, making the system less efficient. To address these challenges, the design utilizes a smaller SNN controller that processes 8-bit inputs, employing a parallel pipelining strategy to handle the larger 64-bit data effectively.

The SNN accelerator block leverages the parallel pipelining concept, where eight smaller SNN controllers, each handling 8-bit inputs, operate in parallel. This method breaks down the larger 64-bit input into more manageable 8-bit chunks, which are processed simultaneously by the eight SNN controllers. This approach not only overcomes the limitations of the GPIO interface but also enhances the system's overall processing efficiency. By distributing the workload across multiple smaller units, the system can handle large data inputs more quickly and with reduced strain on resources compared to processing a single 64-bit input directly.

Implementing parallel pipelining in the SNN accelerator offers several advantages. Firstly, it ensures that the data transfer between the GPIO interface and the SNN accelerator remains within the 32-bit constraint, facilitating smooth and efficient communication. Secondly, parallel processing allows the system to achieve higher throughput, as multiple 8-bit data chunks are processed concurrently. This parallelism reduces latency and increases the speed at which neural computations are performed, which is essential for real-time applications in brain-machine interfaces.

In addition to the parallel pipelining strategy, the SNN accelerator incorporates a dropout layer, which is critical for dimensionality reduction. The dropout layer transforms the 64-bit output from the SNN accelerator into an 8-bit format. This reduction in data size simplifies subsequent analysis and classification tasks, making the system more effective for real-time applications. The dropout layer works by randomly "dropping out" a fraction of the neurons during the training phase, which prevents overfitting and improves the generalization of the neural network. In the context of this hardware implementation, the dropout layer reduces the complexity of the output data, facilitating faster and more efficient analysis.

The inclusion of the dropout layer and the parallel pipelining strategy is highly beneficial for the overall performance of the SNN accelerator. By reducing the data size at the output stage, the dropout layer minimizes the load on subsequent processing stages, preventing potential bottlenecks and ensuring smooth data flow throughout the system. This dimensionality reduction also makes the analysis of neural network outputs more manageable, enabling quicker decision-making and response in brain-machine interface applications.

Furthermore, the parallel pipelining approach enhances the scalability of the SNN accelerator. As neural networks grow in complexity and require processing of larger data sets, the system can scale by adding more parallel processing units, maintaining high performance without significantly increasing resource utilization. This scalability is crucial for adapting to the evolving demands of brain-machine interfaces, which may require increasingly sophisticated neural computations.

The OUTPUT_CLASSIFIER layer in the hardware implementation of the SNN accelerator plays a crucial role in categorizing the neural network outputs into five distinct classes. These classes include other non-arm motions, imagination of motion of both hands, actual motion of both hands, actual motion of the right hand, and actual motion of the left hand. By receiving the processed spike data from the SNN accelerator, the OUTPUT_CLASSIFIER utilizes advanced classification algorithms to interpret the neural signals accurately. This layer is designed to handle the reduced 8-bit output data from the dropout layer, ensuring efficient and rapid analysis. The precise classification of neural activity into these specific classes is vital for effective brain-machine interfacing, enabling the system to differentiate between various types of motor intentions and movements with high accuracy. This capability is essential for applications that rely on interpreting complex neural signals to control external devices or provide real-time feedback to users.

3.2.2 PYNQ Overlay

This Python script is integral in demonstrating the application of a spiking neural network accelerator on a PYNQ FPGA board using a Python Overlay as shown in the Fig.7. By utilizing the PYNQ library, the script loads a custom bitstream which implements the neural network. It configures essential GPIO pins for resetting the network, inputting spike signals, enabling latch

functionality, and reading the output classification. This setup allows for real-time interfacing with the neural network, facilitating tasks such as motion detection and classification in applications like brain-computer interfaces and robotic control systems.

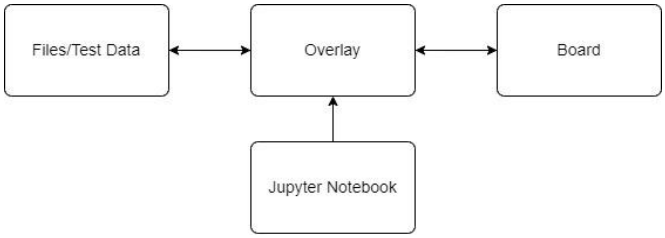

Fig7. PYNQ Overlay Procedure

IV. RESULTS AND DISCUSSION

4.1 Software Results

The software implementation of our EEG signal processing model has undergone extensive validation to assess its performance and accuracy. The results obtained provide a comprehensive understanding of the model's efficacy in classifying EEG signals into distinct categories.

After 19 epochs of training, the model achieved an overall validation accuracy of 86.667%. This high level of accuracy indicates the model's robustness and effectiveness in learning from the training data and making accurate predictions on the validation set. The consistency in performance across multiple epochs demonstrates the stability and reliability of the training process. A detailed analysis of class-specific accuracy reveals the model's capability to distinguish between different categories of EEG signals. For Class 0, the model attained an accuracy of 85.022%, showcasing its proficiency in correctly identifying instances belonging to this class. For Class 1, the accuracy was slightly higher at 88.341%, indicating a strong ability to classify signals associated with this category. The marginal difference in accuracy between the two classes reflects a balanced performance and suggests that the model does not favor one class over the other.

The validation accuracy results provide valuable insights into the model's performance. The high overall accuracy indicates that the model has successfully learned to generalize from the training data to the validation set, making it a reliable tool for EEG signal classification. The class-specific accuracies further affirm the model's capability to handle variations in EEG signal patterns, ensuring accurate classification across different categories.

These results highlight the potential of our model for real-world applications, where accurate and reliable classification of EEG signals is crucial. The insights gained from the validation process will inform future iterations of the model, guiding enhancements to further improve accuracy and performance. Moving forward, these findings lay a strong foundation for deploying the model in practical scenarios, leveraging its robust performance for advanced EEG signal analysis and interpretation.

4.2 Hardware Results

In the post-synthesis resource utilization analysis, it was observed that only 1 out of the available 125 input/output (IO) pins has been utilized in the implemented device. This utilization accounts for approximately 0.80% of the total available IO resources on the device. Such a minimal utilization of IO pins suggests that the design has been judiciously planned to allocate resources efficiently.

Table 4.2: Resource utilization Post-Implementation Report Table

Resources	Utilization	Available	Utilization %
LUT	7598	53200	14.28
LUTRAM	62	17400	0.36
FF	3998	106400	3.76
IO	1	125	0.80
BUFG	2	32	6.25

Table 4.2 shows the post-implementation analysis of fabric resource utilization reveals a highly efficient allocation of resources within the FPGA. The utilization of fabric resources, including LUTs, LUTRAM, flip-flops (FF), I/O, and BUFG, is notably low, with only a fraction of the available resources being utilized. Specifically, a mere 14% of LUTs, 1% of LUTRAM, 4% of FF, 1% of I/O, and 6% of BUFG are utilized, indicating ample room for additional functionality or optimization within the design.

The total power on the chip is measured at 1.472 watts, with a junction temperature of 42°C and a thermal margin of 43.0°C (3.6 watts). Notably, no power is supplied to off-chip devices, highlighting the efficient utilization of on-chip resources. Analysis of power utilization reveals a breakdown of dynamic and static power consumption within the system. Approximately 91% of the total power (1.336 watts) is attributed to dynamic power, while the remaining 9% (0.136 watts) is consumed by static power. Within the dynamic power category, the majority (93%) is consumed by the PS7 component, with minimal contributions from clocks, signals, logic, and I/O. Conversely, static power consumption is entirely attributed to PL Static, emphasizing the negligible impact of static power on overall power consumption. The below Fig.8 shows the pictorial representation of the resources used.

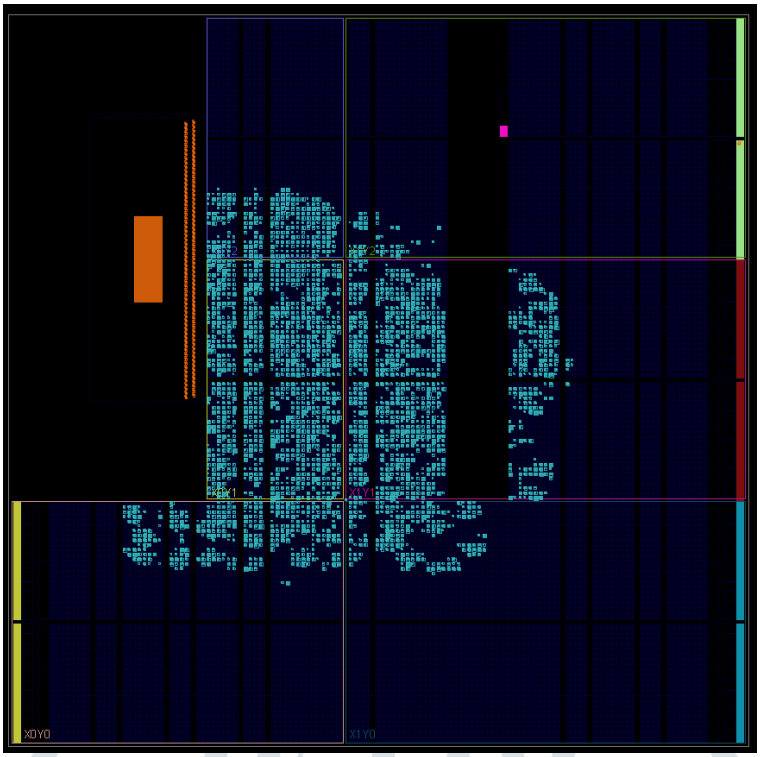

Fig8. Fabric resource utilization post-implementation

The post-implementation design timing analysis confirms the absence of timing violations within the system. With no timing violations detected, the design exhibits robust timing performance, meeting all setup and hold requirements. The worst-case negative slack (WNS) is measured at 3.593 ns, with the worst hold slack at 0.052 ns and the worst pulse width slack at 4.020 ns. These results underscore the design's ability to meet stringent timing constraints and ensure reliable operation under various operating conditions.

4.3 Post Implementation Simulation Results

Two distinct input spike patterns were fed into the system, demonstrating its response through corresponding output spike values and subsequent classification. For the first input spike, labeled as IN_SPIKE_1, the SNN accelerator processed the neural data and generated an output spike value represented as OUTPUT_SPIKE [7:0]. This output was then classified by the OUTPUT_CLASSIFIER layer, yielding a specific class corresponding to one of the predefined motor intentions or states. Similarly, the second input spike, labeled as IN_SPIKE_2, was processed in parallel by the SNN accelerator. The resulting output spike value, also denoted as OUTPUT_SPIKE [7:0], underwent classification, identifying another distinct class from the predefined categories.

The classification results were organized into five distinct classes: other non-arm motions, imagination of motion of both hands, actual motion of both hands, actual motion of the right hand, and actual motion of the left hand. The precise classification of each input spike pattern underscores the system's ability to interpret complex neural signals accurately, paving the way for effective brain-machine interfacing applications. These simulation results as displayed in the Fig.9 and Fig.10, validate the functionality of the SNN accelerator and its classification mechanism, demonstrating its potential for real-time neural signal processing and classification in practical BMI systems.

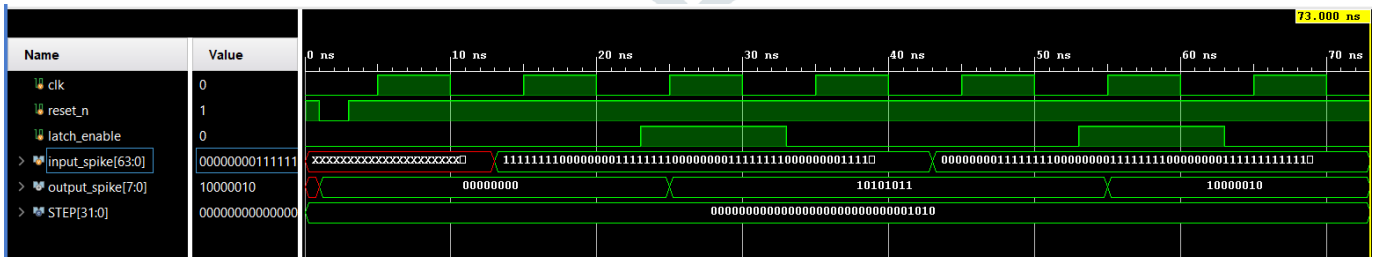

Fig9. Simulation result of two different input spikes and their corresponding output spike value

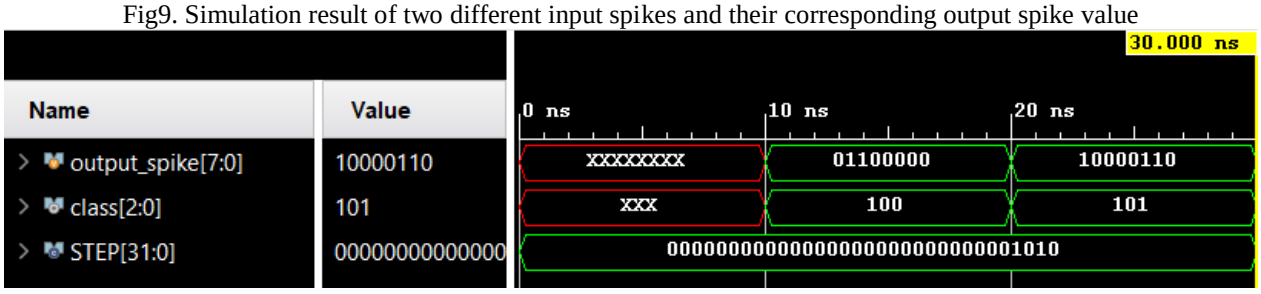

Fig10. Simulation result of two different input spikes and their corresponding classification

4.4 Implementation on PYNQ FPGA

In the pursuit of practical deployment, our FPGA-based EEG signal processing system was ported onto the PYNQ FPGA platform. This implementation involved overlaying the generated bitstream file (.bit) onto the PYNQ platform, providing a seamless integration of our hardware design with the versatility and accessibility of the PYNQ environment. Subsequently, comprehensive testing was conducted using a generalized set of 256 values, representing a subset of the immense input space of 2^64 (18,446,744,073,709,551,616) possible values.

During testing, the output spikes generated by the system were classified into distinct categories based on their patterns and characteristics. Specifically, output spikes corresponding to imagination of motion, motion of both hands, motion of the left hand, and motion of the right hand were labeled as IM_MOTION, MOTION_BOTH, MOTION_LEFT, and MOTION_RIGHT, respectively. Any remaining output spikes were categorized as OTHERS, representing other motion patterns or activities. Figure 11 shows the Right-hand movement and Fig.8 shows the prosthetic arm movement of the same.

The performance of the implemented system was evaluated in terms of calibration time and processing latency. During the calibration phase, the model took 8.152 seconds to calibrate itself, initializing the internal parameters and configurations for subsequent processing. Once calibrated, the system demonstrated impressive efficiency, requiring an average processing time of only 0.004 seconds to classify any input signal and provide the corresponding output label. This low processing latency underscores the real-time capabilities of our FPGA-based system, making it well-suited for applications requiring rapid and accurate analysis of EEG signals.

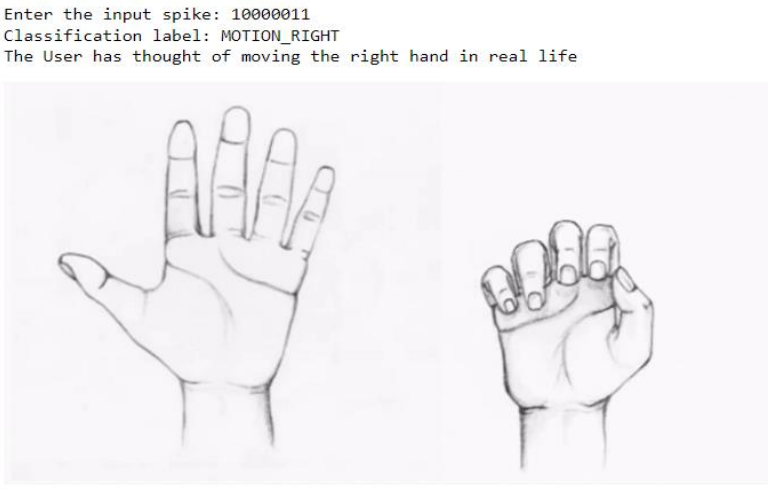

Fig11. The User input is received and processed.

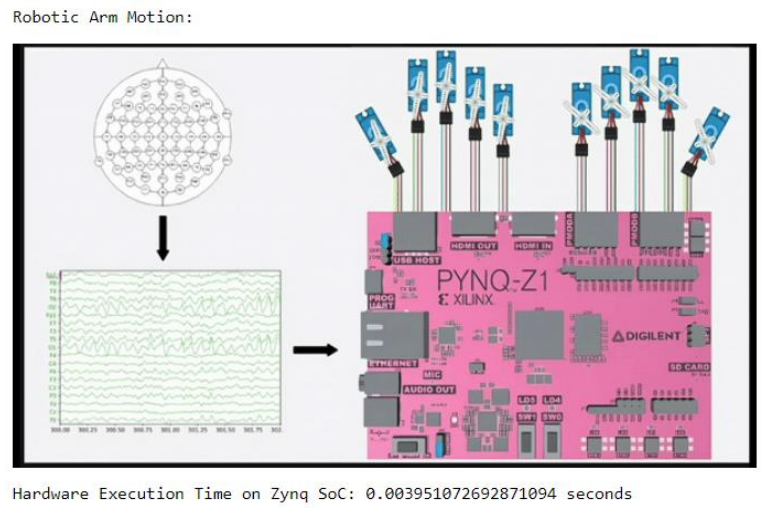

Fig12. The corresponding output LABEL is displayed along with the simulation animation.

V. CONCLUSION AND FUTURE WORK

The trajectory of future advancements in Spiking Neural Network (SNN) algorithms is poised to leverage advanced machine learning paradigms such as reinforcement learning and deep learning to unearth intricate neural activity patterns, thereby refining EEG signal decoding accuracy and efficiency. Envisioning an expanded purview, a multi-modal approach that integrates diverse neural signal modalities like fNIRS and EMG alongside EEG is essential. This approach enriches the Brain-Machine Interface (BMI) system's capacity to discern a broader spectrum of user intentions. Anticipating strides in hardware miniaturization, the focus is on seamlessly embedding the SNN accelerator into wearable devices. This technological progression promises to usher in a new era of portable BMI systems, enabling continuous user interaction with external devices in real-world settings. Furthermore, the goal is to pioneer the development of real-time adaptive systems, empowered by dynamic feedback mechanisms and adaptive learning algorithms that optimize performance and cater to individual user preferences and cognitive dynamics.

In parallel, the vision extends into clinical domains, forging collaborations with healthcare experts to deploy SNN-based BMI systems in therapeutic interventions. This includes tailoring personalized rehabilitation programs for individuals with neurological disorders or motor disabilities. The SNN accelerator's real-time feedback capabilities can significantly augment therapy effectiveness.

Committed to addressing the ethical and privacy ramifications of BMI technology proliferation, future research endeavors will delve into robust methodologies for ensuring data security, user consent protocols, and responsible data handling practices. This commitment aims to fortify user privacy safeguards in BMI applications. Moreover, tackling scalability and accessibility challenges head-on involves developing open-source hardware and software frameworks. This approach fosters interdisciplinary collaboration, explores avenues for commercialization, and promotes knowledge dissemination through educational initiatives. These efforts aim to democratize access to SNN-based BMI systems and amplify their societal impact.

REFRENCES

[1] M. N. Sahadat, S. Dighe, F. Islam and M. Ghovanloo, "An Independent Tongue-Operated Assistive System for Both Access and Mobility," in IEEE Sensors Journal, vol. 18, no. 22, pp. 9401-9409, 15 Nov.15, 2018, doi: 10.1109/JSEN.2018.2870750.

[2] Xu, Qi, et al. "Constructing deep spiking neural networks from artificial neural networks with knowledge distillation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

[3] Neelesh Kumar and Guangzhi Tang and Raymond Yoo and Konstantinos P. Michmizos, Transactions on Machine Learning Research, “Decoding EEG With Spiking Neural Networks on Neuromorphic Hardware”, ISSN=2835-8856, 2022.

[4] YILDIRIM, Oğuzhan, Özden NİYAZ, and Burcu ERKMEN. "FPGA BASED RECONFIGURABLE IMPLEMENTATIONS OF SPIKING NEURAL NETWORKS: A SHORT REVIEW."

[5] Rathi, Nitin, et al. "Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware." ACM Computing Surveys 55.12 (2023): 1-49.

[6] Pietrzak, Paweł, et al. "Overview of spiking neural network learning approaches and their computational complexities." Sensors 23.6 (2023): 3037.

[7] Eshraghian, Jason K., Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. "Training spiking neural networks using lessons from deep learning." Proceedings of the IEEE (2023).

[8] Schalk, G., McFarland, D.J., Hinterberger, T., Birbaumer, N., Wolpaw, J.R. BCI2000: A General-Purpose Brain-Computer Interface (BCI) System. IEEE Transactions on Biomedical Engineering 51(6):1034-1043, 2004.

[9] Plagwitz, Patrick, et al. "SNN vs. CNN Implementations on FPGAs: An Empirical Evaluation." International Symposium on Applied Reconfigurable Computing. Cham: Springer Nature Switzerland, 2024.