INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

EE 605 - Machine Learning Architectures

Project Report:

LIF neuron based on the Euler methods

By:
Ayush Datta Jaiswal - 200102113
Sanskar Kejriwal - 200102082
Soham Karak - 200102107
Vighnesh Deshpande - 200102112

Objective:

To create a two layer synthesizable Spiking Neural Network with Leaky Integrated Fire neurons by using Euler Methods based on this research paper: Toward the Optimal Design and FPGA Implementation of Spiking Neural Networks

Introduction:

This project focuses on creating a two-layer Spiking Neural Network (SNN) using Leaky Integrated Fire (LIF) neurons, based on recent research outlined in "Toward the Optimal Design and FPGA Implementation of Spiking Neural Networks". SNNs emulate the brain's neural firing mechanism through sparse, asynchronous spikes, offering potential advantages in power efficiency and neuromorphic computing.

Our network comprises a Poisson spike generator and LIF neuron modules. The Poisson generator generates spike inputs based on external stimuli (16 bit pixel values between 0-65535), while LIF neurons integrate synaptic inputs that leak over time and emit output spikes when a model specific threshold is reached, mimicking neuron biological behaviour.

We're using Verilog for hardware description and a testbench for simulation to validate the network's functionality across various input scenarios. This project explores the computational capabilities and efficiency of implementing SNNs, with implications for pattern recognition and neuromorphic computing.

Theory:

Spiking Neural Networks (SNNs):

Spiking Neural Networks (SNNs) represent a departure from traditional artificial neural networks by emulating the behaviour of biological neurons more closely. Unlike conventional ANNs, which rely on continuous activations, SNNs communicate using discrete spikes, similar to the action potentials observed in biological brains. This asynchronous and event-driven communication allows for more efficient processing of information and has implications for power efficiency and neuromorphic computing.

Leaky Integrated Fire (LIF) Neurons:

Leaky Integrated Fire (LIF) neurons serve as the basic building blocks of Spiking Neural Networks. These neurons model the behaviour of real neurons by integrating incoming synaptic inputs over time. When the membrane potential of an LIF neuron exceeds a certain threshold, it generates a spike, signalling the neuron's activation. Additionally, LIF neurons incorporate a leak term to simulate the gradual decay of the membrane potential and a refractory period to mimic the recovery time after firing. These characteristics make LIF neurons suitable for modelling biological neurons' dynamics and implementing efficient neural networks.

Euler Method:

The Euler method is a numerical technique used to approximate solutions to ordinary differential equations (ODEs), such as those in Spiking Neural Networks (SNNs). It involves discretizing the equations governing the dynamics of neurons, such as the leaky integrate-and-fire (LIF) model, and iteratively calculating membrane potentials and spike generation. The formula for the Euler method is:

$y\_\{n + 1\} = y\_n + h\, f(t\_n, y\_n)$

Here, h is the step size, and $f(t[n], y[n])$ represents the differential equations describing the neuronal dynamics at time $t[n]$. This method achieves first-order accuracy, meaning the global error is proportional to the step size. This approach enables the implementation and analysis of SNNs in digital hardware environments, crucial for neuromorphic computing and pattern recognition tasks.

We use the following equations for the neuron:

$$I(v[n]) = g_l(v_r - v[n]) - I_s(v[n])$$
$$v[n+1] = v[n] + h/C_m * I(v[n])$$

Here I is synaptic current, v[n] is the membrane potential at nth time step, g is the leaky conductance, Cm is the membrane capacitance and h is the time step size.

Methodology:

Poisson Spike Generator Module:

1. **Input Processing:** The Poisson spike generator module receives inputs, including the clock signal (clk), reset signal (rst), and pixel value (pixel_value), representing the intensity of external stimuli.
2. **Random Number Generation:** Inside the module, a Linear Feedback Shift Register (LFSR) is used to generate pseudo-random numbers. These numbers are crucial for simulating stochastic spiking behavior.

3. **Spike Train Generation:** Based on the pixel value and generated random numbers, the module determines whether a spike should be generated. If the random number is less than the pixel value, a spike is produced, indicating neuronal activation.
4. **Output Formation:** The module outputs the spike train (spike_train), representing the occurrence of spikes over time, as well as a window of spike trains (spike_train_array) for further processing. Additionally, a random number (random_number) is provided for use in subsequent stages of the network.

Leaky Integrated Fire (LIF) Neuron Module:

1. **Input Processing:** The LIF neuron module receives inputs, including the clock signal (clk), reset signal (reset_n), spike input (spike_in) from the Poisson spike generator or another neuron, synaptic weight (weight), threshold (threshold), leak value (leak_value), and refractory period (tref).
2. **Membrane Potential Calculation:** The module integrates incoming synaptic inputs (spike_in multiplied by weight) over time to calculate the membrane potential (voltage) of the neuron.
3. **Leak and Spike Generation:** The module applies a leak term to the membrane potential to simulate gradual decay and determines whether the threshold for spike generation is exceeded. If so, it generates a spike output (spike_out) and enters a refractory period.
4. **Output Formation:** The module provides outputs, including the membrane potential (memb_potential_out) for monitoring neural activity and the refractory period counter (tr) to track the recovery time of the neuron.

SNN System Integration:

1. **Module Instantiation:** The main SNN system module integrates the Poisson spike generator and LIF neuron modules by instantiating them with appropriate parameters and interconnections.
2. **Parameter Configuration:** Parameters such as synaptic weights, thresholds, leak values, and refractory periods are configured based on the requirements of the network and the specific task being performed.
3. **Interconnection:** Spike outputs from one neuron may be fed as inputs to subsequent neurons, creating the layered structure of the network.
4. **Output Formation:** The SNN system module provides outputs, including membrane potentials (memb_potential_out) and spike outputs (spike_out) for

each neuron in the network, along with refractory period counters (tr) and spike trains (spike_train) for monitoring and further analysis.

By systematically implementing and interconnecting these modules, we construct a functional two-layer Spiking Neural Network capable of processing input stimuli and generating appropriate output spikes.

Implementation in Python:

We have done the above implementation in Python in the following notebook: https://colab.research.google.com/drive/1Tuvb7OWddhg7rjMwrugVxZ-Hw2BEAmS-?usp=sharing

Testing and Results:

Testbench Setup and Simulation:

The testbench environment is designed to simulate the behavior of the Spiking Neural Network (SNN) system implemented in Verilog.
It includes stimulus generation, module instantiation, and result monitoring to validate the functionality of the SNN system under various conditions.

Test Scenarios:

1. **Initialization:** The simulation begins with initializing inputs such as clock signal (clk), reset signal (rst), pixel value, synaptic weights, thresholds, leak values, and refractory periods.
2. **Stimulus Variation:** Test scenarios involve varying the pixel value input to observe the network's response to different levels of external stimuli.
3. **Temporal Dynamics:** The simulation tests the temporal dynamics of the network by monitoring membrane potentials, spike outputs, and refractory periods over time.
4. **Edge Cases:** Special cases, such as extreme input values or unusual spike patterns, are tested to ensure robustness and correctness of the network's behavior.

Parameter Values:

WIDTH = 16,
WINDOW_SIZE = 5
Neuron 1 Parameters:

- Weight: weight1 = 16'h0003
- Threshold: threshold1 = 16'h0005

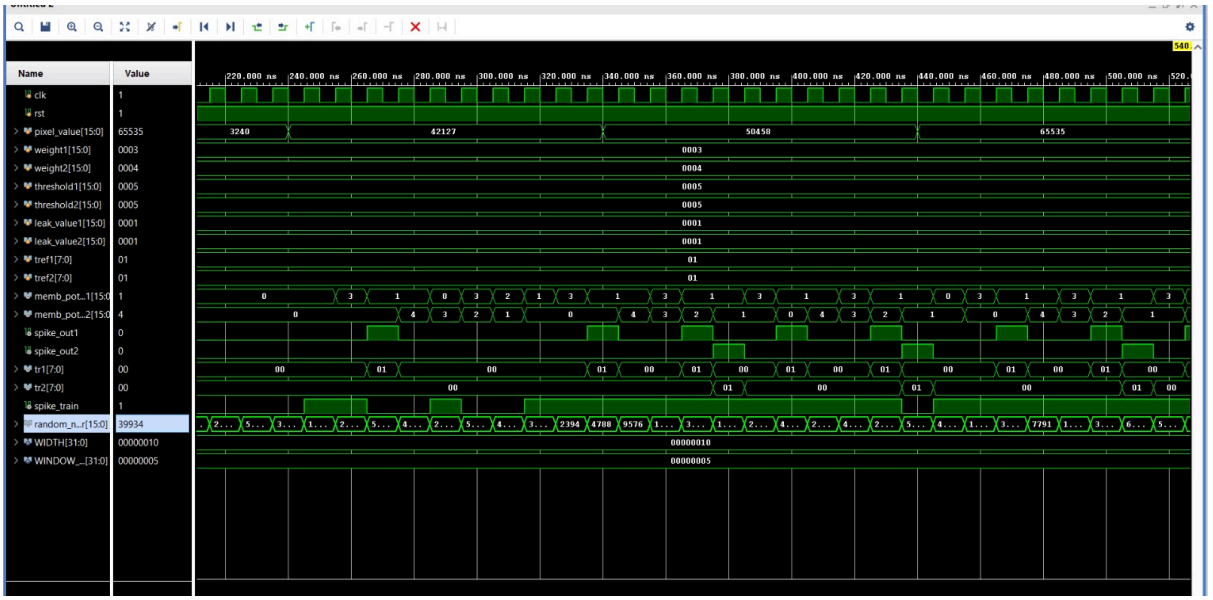
- Leak Value: leak_value1 = 16'd1
- Refractory Period: tref1 = 8'd1
- Resting Potential = 1

Neuron 2 Parameters:

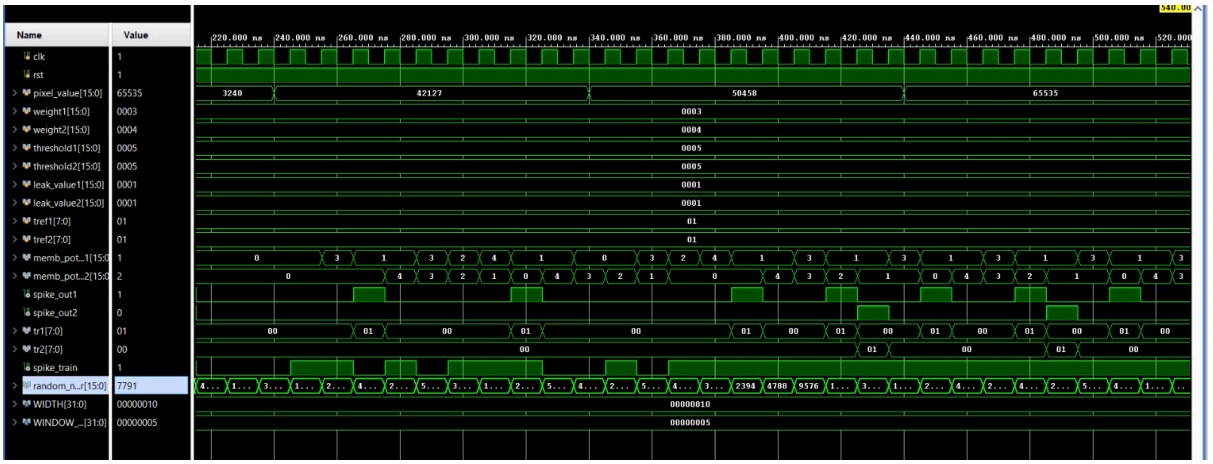
- Weight: weight2 = 16'h0004
- Threshold: threshold2 = 16'h0005
- Leak Value: leak_value2 = 16'd1
- Refractory Period: tref2 = 8'd1
- Resting Potential = 1

Simulation Results:

Pre-Synthesis



Post-Synthesis



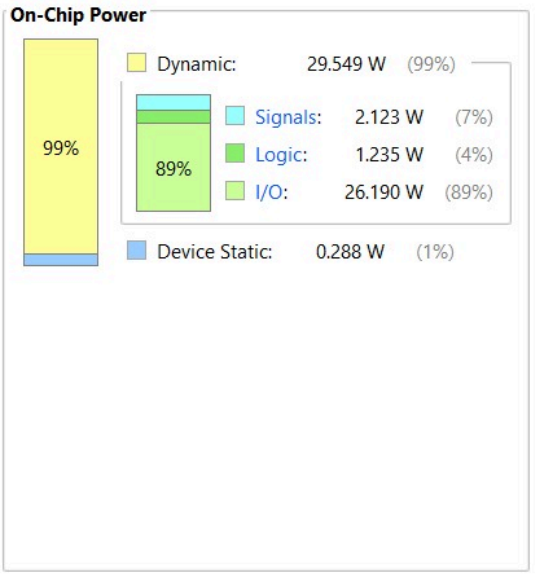
Power Summary

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

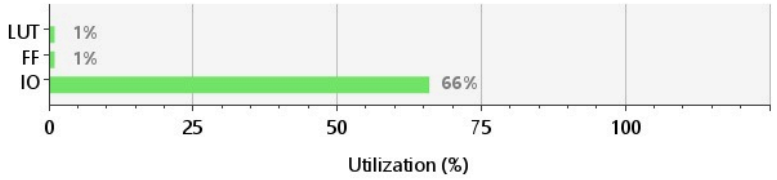
Total On-Chip Power:	**29.836 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**81.2°C**
Thermal Margin:	3.8°C (2.0 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

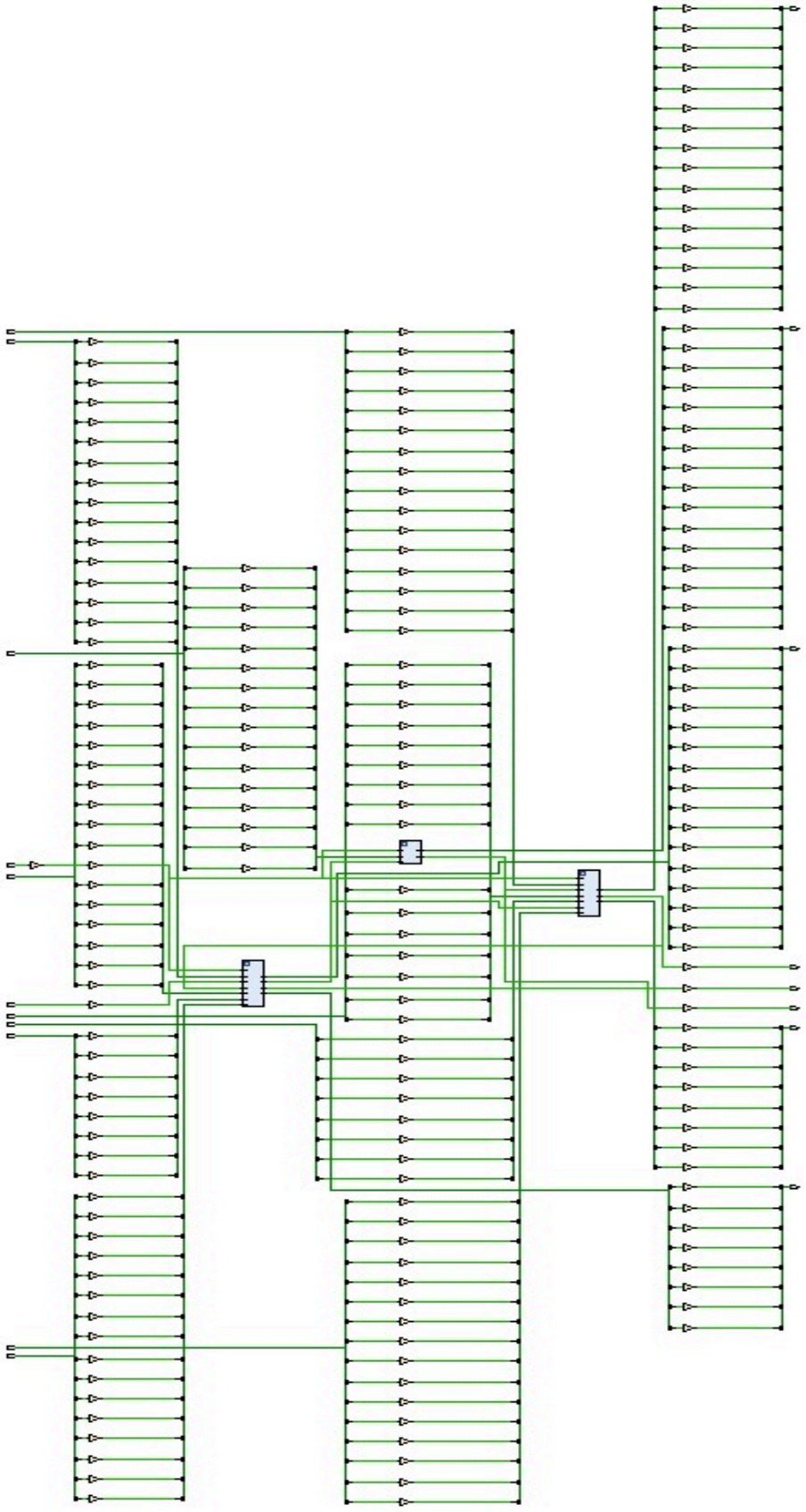


Utilization

Resource	Utilization	Available	Utilization %
LUT	148	41000	0.36
FF	71	82000	0.09
IO	197	300	65.67



Schematic Diagram



IEEE 32 Bit Floating Point Representation:

After implementing integer representation, we added IEEE 32-bit floating point support. We added a Priority Encoder, Adder-Subtractor Module, and Comparator and changed the existing modules to support 32-bit IEEE representation. This also worked very well and was synthesized.

Parameter Values:

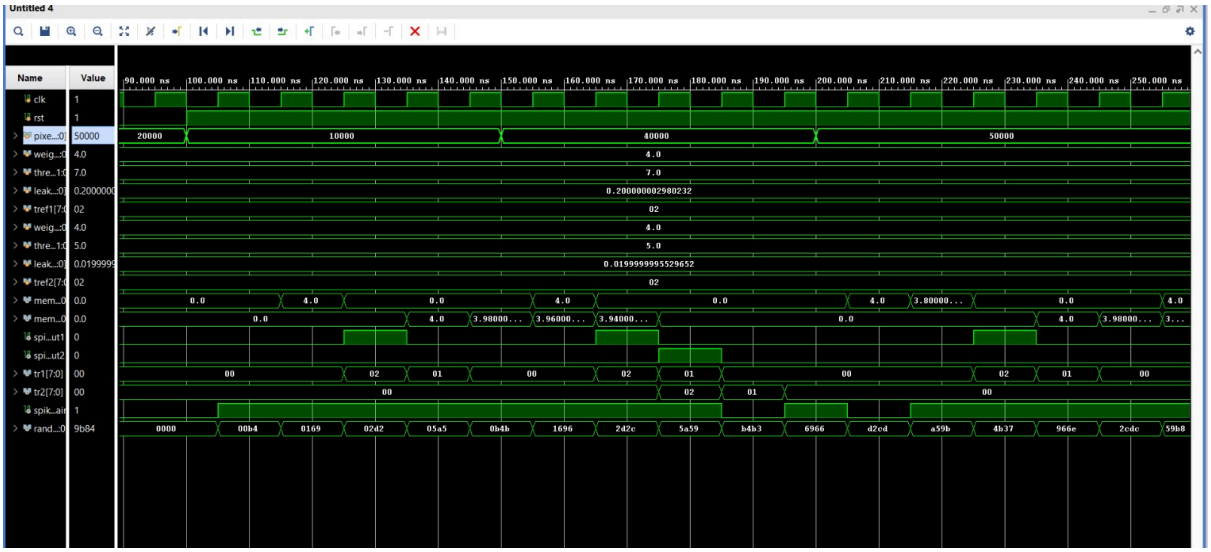
WIDTH = 16,
WINDOW_SIZE = 5
Neuron 1 Parameters:

- Weight: weight1 = 32'h40800000 (4 in decimal)
- Threshold: threshold1 = 32'h40a00000 (7 in decimal)
- Leak Value: leak_value1 = 32'h3f800000 (0.2 in decimal)
- Refractory Period: tref1 = 8'd2
- Resting Potential = 16’d0

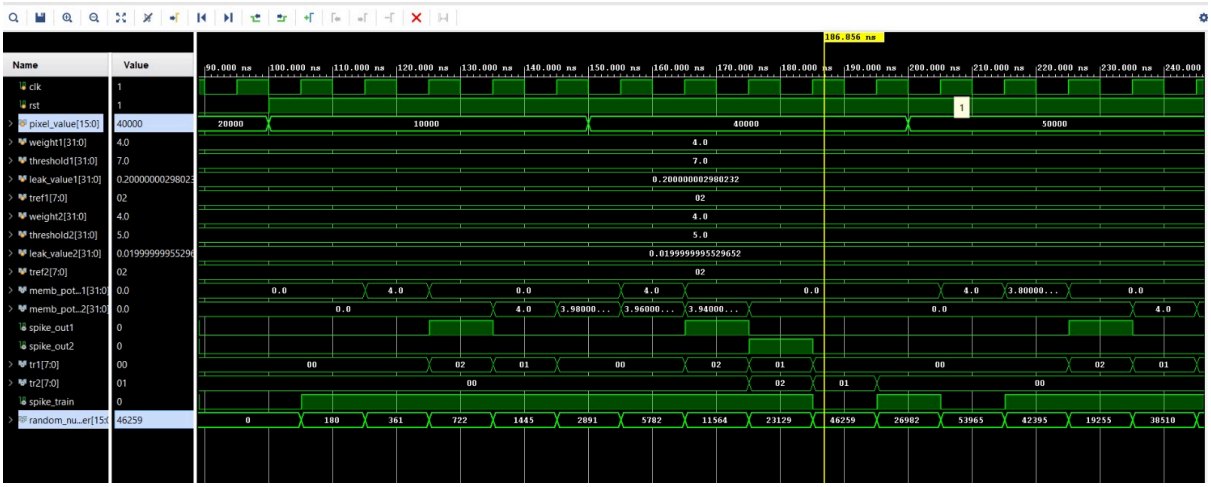
Neuron 2 Parameters:

- Weight: weight2 = 32'h40800000 (4 in decimal)
- Threshold: threshold2 = 32'h40a00000 (5 in decimal)
- Leak Value: leak_value2 = 32'h3f800000 (0.02 in decimal)
- Refractory Period: tref2 = 8'd2
- Resting Potential = 16’d0

Pre-Synthesis



Post-Synthesis



Conclusion:

The simulation results demonstrate the functionality and effectiveness of the implemented SNN system in processing input stimuli and generating appropriate output spikes.
Observations of membrane potentials, spike outputs, and refractory periods provide insights into the network's temporal dynamics and computational capabilities.
Analysis of performance metrics helps assess the network's robustness, accuracy, and suitability for specific applications.
Any discrepancies or unexpected behaviors observed during simulation are analyzed to identify potential issues and areas for improvement in the SNN system.