**Documentation: Programmable Spiking Neuron Array ASIC (Synthesizable Verilog Design)**

**Objective**

The goal of this project is to design a synthesizable and functional Verilog-based hardware description of a **Programmable Spiking Neuron Array** ASIC. This array utilizes **Leaky Integrate-and-Fire (LIF)** neurons with programmable synapses and parameters, controlled via an SPI interface.

**Design Components**

**1. Spiking LIF Neuron Module**

This module models a biologically-inspired neuron that accumulates input current over time. The key features are:

* **Multibit input current** support.
* **Leaky behavior**: subtracts a fixed leak rate each cycle.
* **Threshold-based spiking**: fires an output spike when the accumulated potential exceeds the threshold.
* **Refractory period**: after a spike, the neuron enters an inactive period.

**Parameters Used:**

* THRESHOLD = 256
* LEAK\_RATE = 1
* REFRACTORY\_PERIOD = 32

**2. Spiking Neural Network (SNN) Module**

This module consists of two layers of neurons:

* **Fully connected** structure: each neuron in Layer 1 connects to all neurons in Layer 2.
* **Programmable synaptic weights**.
* **Input currents** are propagated through weights to the next layer.

This module instantiates multiple LIF neurons and connects them with weight multipliers and summing logic.

**3. Programmable Register File**

This module stores all configuration parameters used by the SNN:

* Threshold
* Leak Rate
* Refractory Period
* Input-to-Hidden weights
* Hidden-to-Output weights

The register file is **write-accessible** and can be programmed externally via SPI.

**4. SPI Interface**

An SPI Master-Slave interface is added for external control:

* SPI transactions are used to write configuration values to the register file.
* Address-based decoding routes incoming data to appropriate registers.

**5. Top-Level Module**

This module instantiates:

* The SPI Slave
* The Register File
* The SNN (neural network)

It connects them using internal wires, enabling the system to be programmed and functionally run in hardware.

**6. Testbench**

A Verilog testbench was created to:

* Simulate SPI communication to program the register file.
* Apply input current to the network.
* Monitor the output spike signal to confirm correct behavior.

**Development Flow Using ChatGPT**

The following iterative instructions were given to ChatGPT to develop and validate the design:

1. **Initial Neuron Design:**

Can you write a verilog module for a spiking leaky integrate and fire neuron? Please include a multibit current input and a refractory period. I want the threshold to be 256 and refractory to be 8. I want my leaky rate to be 1 and refractory period duration to be 32.

1. **Validation and Feature Matching:**

Does the code provided implement all the functionalities as the final code provided in the paper? If not, modify my code to implement all the features.

1. **Network Construction:**

Create a new module that instantiates a network of neurons with 2 layers in a fully connected fashion. Please connect them with programmable synapses.

1. **Array Fixing:**

I think there is an issue with the 2D array while connecting neurons. Please rectify it and give the correct code.

1. **Configuration Module:**

Can you provide a programmable register file that stores all the parameters for our network? Include weights as parameters and let’s say both layers use the same parameters.

1. **SPI Interface:**

Create an SPI interface to communicate with the network module above.

1. **Top-Level Integration:**

Create a top file to connect this SPI module with the network module. The top module needs to instantiate both the SPI module and the network module, appropriately connecting internal signals and declaring input and output ports.

1. **Testbench Creation:**

Write a simple testbench to check if my design is working.

**Conclusion**

This design represents a complete digital hardware model of a programmable SNN ASIC using LIF neurons, suitable for FPGA prototyping or further ASIC synthesis. The design is modular, scalable, and testable, with parameter programmability enabled via SPI.