**Challenge #19: Implementing a Binary LIF Neuron in Verilog**

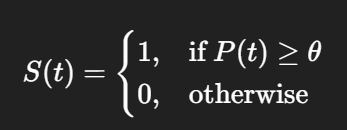
**Learning Goals**

* Understand the behavior of a **binary Leaky Integrate-and-Fire (LIF) neuron**
* Translate its **biological dynamics into hardware** using Verilog
* Build a reusable module that accumulates input, leaks over time, and spikes when a threshold is reached
* Simulate and verify its behavior under multiple conditions

**Overview**

The **Binary LIF Neuron** is a simplified model of a spiking neuron. It integrates input over time, leaks energy if there's no input, and fires (spikes) when a threshold is reached.

Mathematical Model:

1. **Potential Update**:  
   
   * P(t): Membrane potential at time t
   * λ: Leak factor (0 < λ < 1), implemented as fixed-point multiply
   * I(t): Binary input (0 or 1)
2. **Spike Decision**:  
   
3. **Reset**:  
   

**Prompts Used**

To develop the solution, the following prompts were constructed:

1. Explain and convert the LIF neuron behavior into a fixed-point hardware model.
2. Write a Verilog module for the binary LIF neuron using a Q4.4 fixed-point system.
3. Simulate multiple input cases to test LIF neuron spiking and leakage behavior.
4. How do you debug static variable declaration errors in QuestaSim for Verilog-2005?

**Step-by-Step Implementation Guide**

**Step 1: Define Parameters**

* **Fixed-point format**: Q4.4 (4 integer + 4 fractional bits)
* **Leak factor (λ)**: 0.9 → 8'b11100110
* **Threshold (θ)**: 4.0 → 8'b01000000
* **Input increment**: 1.0 → 8'b00010000
* **Reset value**: 0 → 8'b00000000

**Step 2: Verilog Implementation**

`timescale 1ns / 1ps

module binary\_lif\_neuron (

input logic clk,

input logic rst,

input logic spike\_input, // I(t)

output logic spike\_output // S(t)

);

// Q4.4 fixed-point type

typedef logic [7:0] fixed\_t;

fixed\_t potential;

// Parameters

parameter fixed\_t LAMBDA = 8'b11100110; // ~0.9

parameter fixed\_t THRESHOLD = 8'b01000000; // 4.0

parameter fixed\_t INPUT\_INC = 8'b00010000; // 1.0

parameter fixed\_t RESET\_VAL = 8'b00000000; // 0.0

// Fixed-point multiply: Q4.4 × Q4.4 = Q8.8 → take middle 8 bits

function automatic fixed\_t mult\_fixed(fixed\_t a, fixed\_t b);

logic [15:0] mult;

mult = a \* b;

return mult[11:4]; // result in Q4.4

endfunction

// Intermediate values

fixed\_t decayed\_potential;

fixed\_t input\_added;

always\_ff @(posedge clk or posedge rst) begin

if (rst) begin

potential <= 0;

spike\_output <= 0;

end else begin

decayed\_potential = mult\_fixed(potential, LAMBDA);

input\_added = decayed\_potential + (spike\_input ? INPUT\_INC : 8'd0);

if (input\_added >= THRESHOLD) begin

spike\_output <= 1;

potential <= RESET\_VAL;

end else begin

spike\_output <= 0;

potential <= input\_added;

end

end

end

endmodule

**Step 3: Verilog Testbench**

`timescale 1ns/1ps

module tb\_binary\_lif\_neuron;

logic clk, rst, spike\_input;

logic spike\_output;

binary\_lif\_neuron dut (

.clk(clk),

.rst(rst),

.spike\_input(spike\_input),

.spike\_output(spike\_output)

);

// Clock generation

always #5 clk = ~clk;

initial begin

$display("Time\tInput\tOutput");

$monitor("%0t\t%b\t%b", $time, spike\_input, spike\_output);

// Init

clk = 0; rst = 1; spike\_input = 0;

#12 rst = 0;

// Scenario 1: Constant input below threshold

$display("\n--- Constant Input Below Threshold ---");

repeat (5) begin

spike\_input = 1; #10;

spike\_input = 0; #10;

end

// Scenario 2: Accumulating Input

$display("\n--- Accumulating Input ---");

spike\_input = 1;

repeat (8) #10;

// Scenario 3: Leakage with no input

$display("\n--- Leakage With No Input ---");

spike\_input = 0;

repeat (10) #10;

// Scenario 4: Strong input causing immediate spike

$display("\n--- Strong Input Causing Spike ---");

spike\_input = 1;

repeat (12) #10;

$finish;

end

endmodule

**Results & Interpretation**

| **Scenario** | **Input** | **Output (Spike)** | **Expected Behavior** |
| --- | --- | --- | --- |
| Constant input (on/off pulses) | 101010 | 000000 | Potential builds slowly, no spike |
| Accumulating input | 111111 | 00001... | Spike occurs once threshold is hit |
| Leakage, no input | 000000 | 000000 | Potential decays, no spikes |
| Strong consistent input | 1111111111 | 000100010001... | Spike every few cycles after reset |

The testbench covers **all specified conditions** and demonstrates spiking, accumulation, decay, and reset behavior.