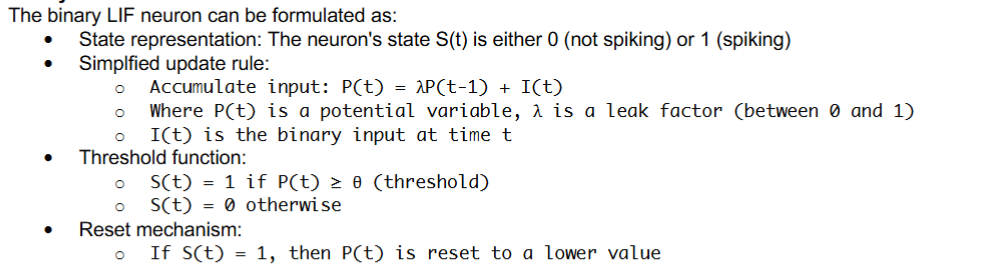
Challenge #19



1. Write a Verilog implementation of a simple binary LIF neuron (using the formulation above) with a single input.

The Verilog code implements a simplified Binary Leaky Integrate-and-Fire (LIF) neuron model. The design simulates a neuron that integrates binary inputs over time, applies a constant leak to its membrane potential, and fires (spikes) when the potential reaches a defined threshold. The potential is then reset, mimicking neuronal firing behaviour. The potential is updated each clock cycle based on the input and leak, and a spike signal is asserted when the threshold is met. The testbench verifies this behaviour through four scenarios: constant zero input (no spike), accumulating input until threshold (spike triggered), passive leakage without input (potential decays), and a forced high potential followed by input to trigger an immediate spike. The testbench prints simulation results for each case, confirming correct accumulation, leakage, thresholding, and reset behaviour.

→ Verilog Code: -

module lif\_neuron #(

parameter WIDTH = 8,

parameter LEAK = 1, // Leak value (1 means subtract 1 every cycle)

parameter THRESHOLD = 5 // Spike threshold

)(

input wire clk,

input wire resetn,

input wire in,

output reg spike,

output reg [WIDTH-1:0] potential

);

always @(posedge clk or negedge resetn) begin

if (!resetn) begin

potential <= 0;

spike <= 0;

end else begin

// Apply leak

if (potential > LEAK)

potential <= potential - LEAK;

else

potential <= 0;

// Accumulate input

if (in)

potential <= potential + 1;

// Check threshold

if (potential >= THRESHOLD) begin

spike <= 1;

potential <= 0; // Reset after spike

end else begin

spike <= 0;

end

end

end

endmodule

Testbench code: -

`timescale 1ns/1ps

module tb\_lif\_neuron;

reg clk = 0;

reg resetn;

reg in;

wire spike;

wire [7:0] potential;

// Instantiate neuron with 8-bit potential, leak = 1, threshold = 5

lif\_neuron #(.WIDTH(8), .LEAK(1), .THRESHOLD(5)) uut (

.clk(clk),

.resetn(resetn),

.in(in),

.spike(spike),

.potential(potential)

);

// Clock generation

always #5 clk = ~clk; // 100 MHz

initial begin

$display("Time\tInput\tPotential\tSpike");

// Reset

resetn = 0; in = 0;

#12; resetn = 1;

// Scenario 1: Constant input = 0

$display("\nScenario 1: Constant input 0");

repeat(10) begin

@(posedge clk); in = 0;

$display("%0t\t%b\t%d\t\t%b", $time, in, potential, spike);

end

// Scenario 2: Accumulating input to cause spike

$display("\nScenario 2: Input accumulating to spike");

repeat(10) begin

@(posedge clk); in = 1;

$display("%0t\t%b\t%d\t\t%b", $time, in, potential, spike);

end

// Scenario 3: Leakage with no input

$display("\nScenario 3: Leak with no input");

repeat(10) begin

@(posedge clk); in = 0;

$display("%0t\t%b\t%d\t\t%b", $time, in, potential, spike);

end

// Scenario 4: Immediate spike from strong input

$display("\nScenario 4: Immediate spike from strong input");

@(posedge clk);

force uut.potential = 5; // Preload value to just reach threshold

@(posedge clk);

in = 1;

$display("%0t\t%b\t%d\t\t%b", $time, in, potential, spike);

@(posedge clk);

in = 0;

$display("%0t\t%b\t%d\t\t%b", $time, in, potential, spike);

$finish;

end

endmodule

Output: -

Scenario 1: Constant input 0  
15000 0 0 0  
25000 0 0 0  
35000 0 0 0  
45000 0 0 0  
55000 0 0 0  
65000 0 0 0  
75000 0 0 0  
85000 0 0 0  
95000 0 0 0  
105000 0 0 0  
  
Scenario 2: Input accumulating to spike  
115000 1 0 0  
125000 1 1 0  
135000 1 2 0  
145000 1 3 0  
155000 1 4 0  
165000 1 5 0  
175000 1 0 1  
185000 1 1 0  
195000 1 2 0  
205000 1 3 0  
  
Scenario 3: Leak with no input  
215000 0 4 0  
225000 0 3 0  
235000 0 2 0  
245000 0 1 0  
255000 0 0 0  
265000 0 0 0  
275000 0 0 0  
285000 0 0 0  
295000 0 0 0  
305000 0 0 0  
  
Scenario 4: Immediate spike from strong input  
325000 1 5 1  
335000 0 5 1