EE235241 SISTEM DAN RANGKAIAN ELEKTRONIKA

LAPORAN PROJECT IMPLEMENTASI *SPIKING NEURAL NETWORK* (SNN) PADA FPGA XILINX CMOD A7-35t

Disusun Oleh:

Karina Adi Putri

6022231071

Sibghah Rakasiwi Fiddihaq

6022231038

Ditya Garda Nugraha

5022201212

Dosen Pengampu:

Astria Nur Irfansyah, S.T., M.Eng., Ph.D.

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institur Teknologi Sepuluh Nopember

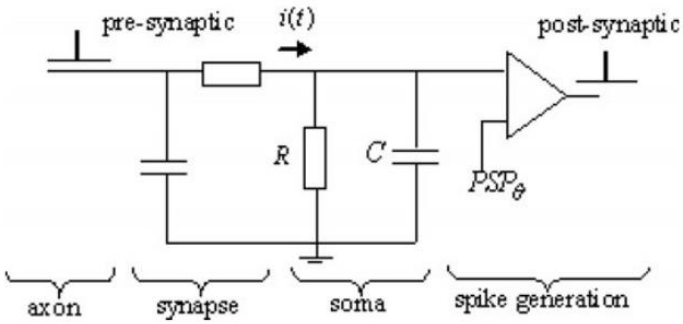
2023/2024

A. Dasar Teori

Spiking Neural Network (SNN) merupakan jenis artificial neural network (ANN) yang informasinya direpresentasikan sebagai binary events (spikes), meniru cara kerja sel saraf di otak manusia bekerja, dan proses learning juga memiliki prinsip kerja yang sama dengan otak manusia. SNN adalah generasi ketiga dari neural network, yang meniru sel saraf pada otak dan komunikasi melalui spikes.

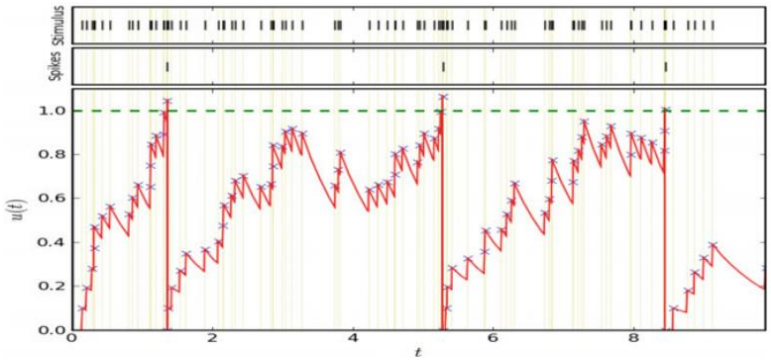
Pada tahun 1952, Hodgkin dan Huxley melakukan eksperimen pada akson dari cumi-cumi. Berdasarkan penelitian ini, diketahui bahwa terdapat tiga channel atau tiga saluran pada neuron, yang merupakan Sodium(Na), Potassium(K), dan leakage (L) dengan resistansi. Berdasarkan penelitian ini, dikembangkan model neuron Hodgkin-Huxley yang merupakan deskripsi dari ketiga channel dan aliran ion dari neuron ketika menghasilkan spike, yang mana tidak mirip dengan sel saraf, dan juga memiliki beberapa kelemahan. Penelitian ini merupakan awal permulaan dari perkembangan model neuron,

Dibandingkan dengan model Hodgkin dan Huxley, model Leaky Integrate-and-Fire (LIF) melihat neuron sebagai leaky integrator, yang mana akan menghasilkan output spike apabila input voltage mencapai nilai threshold kemudian akan reset pada resting state. Weighted input dari neuron akan dijumlahkan terus menerus, dengan waktu bersamaan juga terjadi leakage. Ketika nilai input melebihi nilai threshold, maka LIF neuron akan menghasilkan spike.



(a) Electrical circuit representing the LIF spiking neuron model

Gambar 1. Rangkaian LIF Neuron (Sumber: Kasabov, 2019)



(b) The membrane potential of a LIF neuron accumulates input spikes as stimuli. When the potential reaches a threshold, the neuron emits an output spike.

Gambar 2. LIF Neuron. (Sumber: Kasabov, 2019)

LIF neuron dapat direpresentasikan dengan rangkaian RC sederhana. Bentuk standar dari neuron dapat dituliskan sebagai persamaan 1, yang mana u(t) merupakan potensial membran dan $\tau_m = RC$ merupakan time constant dari membran neuron.

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \quad (1)$$

B. Simulasi Icarus Verilog

1. Program Verilog

```
`timescale 1ns / 1ps

module lif(
    input wire clk,
    input wire rst,
    input wire bit0,
    output reg led0,
    output reg [31:0] v_mem  // biar bisa monitor vmem
    );

    // Define constants as integers scaled by a factor of 1e6 (for
example)
    parameter integer TIME_STEP = 1; // 1e6 to match scaling of TAU
    parameter integer V_TH = 50;
    parameter integer V_RESET = 0;
    parameter integer WEIGHT = 5;
    parameter integer LEAK = 1;

    parameter integer RESISTOR = 1; // 1 (1 Ohm) scaled by 1e6
    parameter integer CAPACITOR = 1; // 1 (1 Farad) scaled by 1e6
    parameter integer TAU = (RESISTOR * CAPACITOR); // tau = R * C
scaled by 1e6

    reg input_current;
    reg spike;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            v_mem <= V_RESET;
            spike <= 0;
            led0 <= 0;
            input_current <= 0; //biar jelas nialainya
        end else begin
            if (bit0 == 1) begin
                input_current <= 1;
            end else begin
                input_current <= 0;
```

```
            end

            // Update v_mem with fixed-point arithmetic
            v_mem <= v_mem + (TIME_STEP / TAU) * (-LEAK +
(input_current * WEIGHT));

            if (v_mem >= V_TH) begin
                spike <= 1;
                v_mem <= V_RESET;
            end else begin
                spike <= 0;
            end

            led0 <= spike;
        end
    end
endmodule
```

2. Program Test bench

```
`timescale 1ns / 1ps

module lif_testbench;
    reg clk;
    reg rst;
    reg bit0;
    wire led0;
    wire [31:0] v_mem;  // Declare v_mem as wire

    // Instantiate the Unit Under Test (UUT)
    lif uut (
        .clk(clk),
        .rst(rst),
        .bit0(bit0),
        .led0(led0),
        .v_mem(v_mem)  // Connect v_mem to the UUT
    );

    initial begin
        // Initialize Inputs
        clk = 0;
```

```
        rst = 0;
        bit0 = 0;

        // Apply reset
        rst = 1;
        #10;
        rst = 0;

        // Test case 1: Apply bit0 = 1
        #10 bit0 = 1;
        #1000 bit0 = 0;

        // Test case 2: Apply bit0 = 0
        #10 bit0 = 0;
        #1000 bit0 = 1;

        // End simulation
        #2000 $finish;
    end

    always #20 clk = ~clk; // Clock generator with 10ns period

    initial begin
        // Monitor the outputs
        $monitor("Time = %0dns, clk = %b, rst = %b, bit0 = %b, v_mem =
%d, led0 = %b", $time, clk, rst, bit0, v_mem, led0);

        // Dump waveform data
        $dumpfile("lif_testbench.vcd");
        $dumpvars(0, lif_testbench);
    end

endmodule
```

3. Hasil Simulasi

```
Time = 0ns, clk = 0, rst = 1, bit0 = 0, v_mem =          0, led0 = 0
Time = 10ns, clk = 0, rst = 0, bit0 = 0, v_mem =         0, led0 = 0
Time = 20ns, clk = 1, rst = 0, bit0 = 1, v_mem = 4294967295, led0 = 0
Time = 40ns, clk = 0, rst = 0, bit0 = 1, v_mem = 4294967295, led0 = 0
Time = 60ns, clk = 1, rst = 0, bit0 = 1, v_mem =          0, led0 = 0
Time = 80ns, clk = 0, rst = 0, bit0 = 1, v_mem =          0, led0 = 0
Time = 100ns, clk = 1, rst = 0, bit0 = 1, v_mem =         4, led0 = 1
Time = 120ns, clk = 0, rst = 0, bit0 = 1, v_mem =         4, led0 = 1
```

```
Time = 140ns, clk = 1, rst = 0, bit0 = 1, v_mem =  8, led0 = 0
Time = 160ns, clk = 0, rst = 0, bit0 = 1, v_mem =  8, led0 = 0
Time = 180ns, clk = 1, rst = 0, bit0 = 1, v_mem = 12, led0 = 0
Time = 200ns, clk = 0, rst = 0, bit0 = 1, v_mem = 12, led0 = 0
Time = 220ns, clk = 1, rst = 0, bit0 = 1, v_mem = 16, led0 = 0
Time = 240ns, clk = 0, rst = 0, bit0 = 1, v_mem = 16, led0 = 0
Time = 260ns, clk = 1, rst = 0, bit0 = 1, v_mem = 20, led0 = 0
Time = 280ns, clk = 0, rst = 0, bit0 = 1, v_mem = 20, led0 = 0
Time = 300ns, clk = 1, rst = 0, bit0 = 1, v_mem = 24, led0 = 0
Time = 320ns, clk = 0, rst = 0, bit0 = 1, v_mem = 24, led0 = 0
Time = 340ns, clk = 1, rst = 0, bit0 = 1, v_mem = 28, led0 = 0
Time = 360ns, clk = 0, rst = 0, bit0 = 1, v_mem = 28, led0 = 0
Time = 380ns, clk = 1, rst = 0, bit0 = 1, v_mem = 32, led0 = 0
Time = 400ns, clk = 0, rst = 0, bit0 = 1, v_mem = 32, led0 = 0
Time = 420ns, clk = 1, rst = 0, bit0 = 1, v_mem = 36, led0 = 0
Time = 440ns, clk = 0, rst = 0, bit0 = 1, v_mem = 36, led0 = 0
Time = 460ns, clk = 1, rst = 0, bit0 = 1, v_mem = 40, led0 = 0
Time = 480ns, clk = 0, rst = 0, bit0 = 1, v_mem = 40, led0 = 0
Time = 500ns, clk = 1, rst = 0, bit0 = 1, v_mem = 44, led0 = 0
Time = 520ns, clk = 0, rst = 0, bit0 = 1, v_mem = 44, led0 = 0
Time = 540ns, clk = 1, rst = 0, bit0 = 1, v_mem = 48, led0 = 0
Time = 560ns, clk = 0, rst = 0, bit0 = 1, v_mem = 48, led0 = 0
Time = 580ns, clk = 1, rst = 0, bit0 = 1, v_mem = 52, led0 = 0
Time = 600ns, clk = 0, rst = 0, bit0 = 1, v_mem = 52, led0 = 0
Time = 620ns, clk = 1, rst = 0, bit0 = 1, v_mem =  0, led0 = 0
Time = 640ns, clk = 0, rst = 0, bit0 = 1, v_mem =  0, led0 = 0
Time = 660ns, clk = 1, rst = 0, bit0 = 1, v_mem =  4, led0 = 1
Time = 680ns, clk = 0, rst = 0, bit0 = 1, v_mem =  4, led0 = 1
Time = 700ns, clk = 1, rst = 0, bit0 = 1, v_mem =  8, led0 = 0
Time = 720ns, clk = 0, rst = 0, bit0 = 1, v_mem =  8, led0 = 0
Time = 740ns, clk = 1, rst = 0, bit0 = 1, v_mem = 12, led0 = 0
Time = 760ns, clk = 0, rst = 0, bit0 = 1, v_mem = 12, led0 = 0
Time = 780ns, clk = 1, rst = 0, bit0 = 1, v_mem = 16, led0 = 0
Time = 800ns, clk = 0, rst = 0, bit0 = 1, v_mem = 16, led0 = 0
Time = 820ns, clk = 1, rst = 0, bit0 = 1, v_mem = 20, led0 = 0
Time = 840ns, clk = 0, rst = 0, bit0 = 1, v_mem = 20, led0 = 0
Time = 860ns, clk = 1, rst = 0, bit0 = 1, v_mem = 24, led0 = 0
Time = 880ns, clk = 0, rst = 0, bit0 = 1, v_mem = 24, led0 = 0
Time = 900ns, clk = 1, rst = 0, bit0 = 1, v_mem = 28, led0 = 0
Time = 920ns, clk = 0, rst = 0, bit0 = 1, v_mem = 28, led0 = 0
Time = 940ns, clk = 1, rst = 0, bit0 = 1, v_mem = 32, led0 = 0
Time = 960ns, clk = 0, rst = 0, bit0 = 1, v_mem = 32, led0 = 0
Time = 980ns, clk = 1, rst = 0, bit0 = 1, v_mem = 36, led0 = 0
Time = 1000ns, clk = 0, rst = 0, bit0 = 1, v_mem = 36, led0 = 0
Time = 1020ns, clk = 1, rst = 0, bit0 = 0, v_mem = 40, led0 = 0
Time = 1040ns, clk = 0, rst = 0, bit0 = 0, v_mem = 40, led0 = 0
Time = 1060ns, clk = 1, rst = 0, bit0 = 0, v_mem = 39, led0 = 0
Time = 1080ns, clk = 0, rst = 0, bit0 = 0, v_mem = 39, led0 = 0
Time = 1100ns, clk = 1, rst = 0, bit0 = 0, v_mem = 38, led0 = 0
```

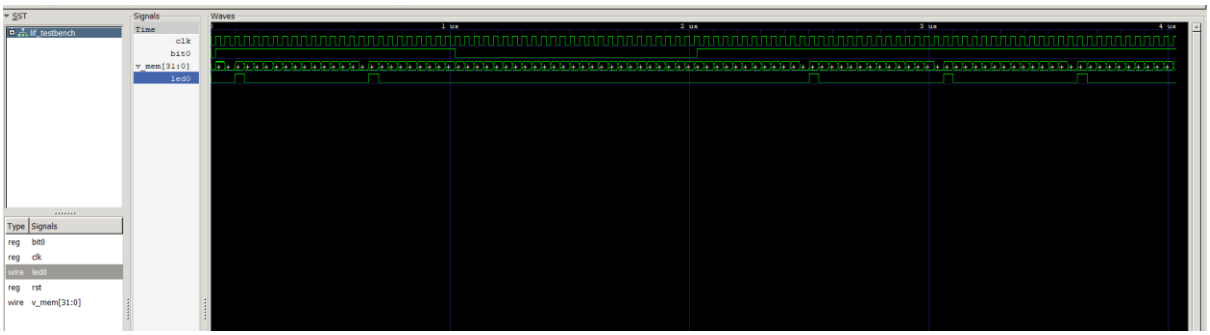
```
Time = 1120ns, clk = 0, rst = 0, bit0 = 0, v_mem =       38, led0 = 0
Time = 1140ns, clk = 1, rst = 0, bit0 = 0, v_mem =       37, led0 = 0
Time = 1160ns, clk = 0, rst = 0, bit0 = 0, v_mem =       37, led0 = 0
Time = 1180ns, clk = 1, rst = 0, bit0 = 0, v_mem =       36, led0 = 0
Time = 1200ns, clk = 0, rst = 0, bit0 = 0, v_mem =       36, led0 = 0
Time = 1220ns, clk = 1, rst = 0, bit0 = 0, v_mem =       35, led0 = 0
Time = 1240ns, clk = 0, rst = 0, bit0 = 0, v_mem =       35, led0 = 0
Time = 1260ns, clk = 1, rst = 0, bit0 = 0, v_mem =       34, led0 = 0
Time = 1280ns, clk = 0, rst = 0, bit0 = 0, v_mem =       34, led0 = 0
Time = 1300ns, clk = 1, rst = 0, bit0 = 0, v_mem =       33, led0 = 0
Time = 1320ns, clk = 0, rst = 0, bit0 = 0, v_mem =       33, led0 = 0
Time = 1340ns, clk = 1, rst = 0, bit0 = 0, v_mem =       32, led0 = 0
Time = 1360ns, clk = 0, rst = 0, bit0 = 0, v_mem =       32, led0 = 0
Time = 1380ns, clk = 1, rst = 0, bit0 = 0, v_mem =       31, led0 = 0
Time = 1400ns, clk = 0, rst = 0, bit0 = 0, v_mem =       31, led0 = 0
Time = 1420ns, clk = 1, rst = 0, bit0 = 0, v_mem =       30, led0 = 0
Time = 1440ns, clk = 0, rst = 0, bit0 = 0, v_mem =       30, led0 = 0
Time = 1460ns, clk = 1, rst = 0, bit0 = 0, v_mem =       29, led0 = 0
Time = 1480ns, clk = 0, rst = 0, bit0 = 0, v_mem =       29, led0 = 0
Time = 1500ns, clk = 1, rst = 0, bit0 = 0, v_mem =       28, led0 = 0
Time = 1520ns, clk = 0, rst = 0, bit0 = 0, v_mem =       28, led0 = 0
Time = 1540ns, clk = 1, rst = 0, bit0 = 0, v_mem =       27, led0 = 0
Time = 1560ns, clk = 0, rst = 0, bit0 = 0, v_mem =       27, led0 = 0
Time = 1580ns, clk = 1, rst = 0, bit0 = 0, v_mem =       26, led0 = 0
Time = 1600ns, clk = 0, rst = 0, bit0 = 0, v_mem =       26, led0 = 0
Time = 1620ns, clk = 1, rst = 0, bit0 = 0, v_mem =       25, led0 = 0
Time = 1640ns, clk = 0, rst = 0, bit0 = 0, v_mem =       25, led0 = 0
Time = 1660ns, clk = 1, rst = 0, bit0 = 0, v_mem =       24, led0 = 0
Time = 1680ns, clk = 0, rst = 0, bit0 = 0, v_mem =       24, led0 = 0
Time = 1700ns, clk = 1, rst = 0, bit0 = 0, v_mem =       23, led0 = 0
Time = 1720ns, clk = 0, rst = 0, bit0 = 0, v_mem =       23, led0 = 0
Time = 1740ns, clk = 1, rst = 0, bit0 = 0, v_mem =       22, led0 = 0
Time = 1760ns, clk = 0, rst = 0, bit0 = 0, v_mem =       22, led0 = 0
Time = 1780ns, clk = 1, rst = 0, bit0 = 0, v_mem =       21, led0 = 0
Time = 1800ns, clk = 0, rst = 0, bit0 = 0, v_mem =       21, led0 = 0
Time = 1820ns, clk = 1, rst = 0, bit0 = 0, v_mem =       20, led0 = 0
Time = 1840ns, clk = 0, rst = 0, bit0 = 0, v_mem =       20, led0 = 0
Time = 1860ns, clk = 1, rst = 0, bit0 = 0, v_mem =       19, led0 = 0
Time = 1880ns, clk = 0, rst = 0, bit0 = 0, v_mem =       19, led0 = 0
Time = 1900ns, clk = 1, rst = 0, bit0 = 0, v_mem =       18, led0 = 0
Time = 1920ns, clk = 0, rst = 0, bit0 = 0, v_mem =       18, led0 = 0
Time = 1940ns, clk = 1, rst = 0, bit0 = 0, v_mem =       17, led0 = 0
Time = 1960ns, clk = 0, rst = 0, bit0 = 0, v_mem =       17, led0 = 0
Time = 1980ns, clk = 1, rst = 0, bit0 = 0, v_mem =       16, led0 = 0
Time = 2000ns, clk = 0, rst = 0, bit0 = 0, v_mem =       16, led0 = 0
Time = 2020ns, clk = 1, rst = 0, bit0 = 0, v_mem =       15, led0 = 0
Time = 2030ns, clk = 1, rst = 0, bit0 = 1, v_mem =       15, led0 = 0
Time = 2040ns, clk = 0, rst = 0, bit0 = 1, v_mem =       15, led0 = 0
Time = 2060ns, clk = 1, rst = 0, bit0 = 1, v_mem =       14, led0 = 0
```

```
Time = 2080ns, clk = 0, rst = 0, bit0 = 1, v_mem =       14, led0 = 0
Time = 2100ns, clk = 1, rst = 0, bit0 = 1, v_mem =       18, led0 = 0
Time = 2120ns, clk = 0, rst = 0, bit0 = 1, v_mem =       18, led0 = 0
Time = 2140ns, clk = 1, rst = 0, bit0 = 1, v_mem =       22, led0 = 0
Time = 2160ns, clk = 0, rst = 0, bit0 = 1, v_mem =       22, led0 = 0
Time = 2180ns, clk = 1, rst = 0, bit0 = 1, v_mem =       26, led0 = 0
Time = 2200ns, clk = 0, rst = 0, bit0 = 1, v_mem =       26, led0 = 0
Time = 2220ns, clk = 1, rst = 0, bit0 = 1, v_mem =       30, led0 = 0
Time = 2240ns, clk = 0, rst = 0, bit0 = 1, v_mem =       30, led0 = 0
Time = 2260ns, clk = 1, rst = 0, bit0 = 1, v_mem =       34, led0 = 0
Time = 2280ns, clk = 0, rst = 0, bit0 = 1, v_mem =       34, led0 = 0
Time = 2300ns, clk = 1, rst = 0, bit0 = 1, v_mem =       38, led0 = 0
Time = 2320ns, clk = 0, rst = 0, bit0 = 1, v_mem =       38, led0 = 0
Time = 2340ns, clk = 1, rst = 0, bit0 = 1, v_mem =       42, led0 = 0
Time = 2360ns, clk = 0, rst = 0, bit0 = 1, v_mem =       42, led0 = 0
Time = 2380ns, clk = 1, rst = 0, bit0 = 1, v_mem =       46, led0 = 0
Time = 2400ns, clk = 0, rst = 0, bit0 = 1, v_mem =       46, led0 = 0
Time = 2420ns, clk = 1, rst = 0, bit0 = 1, v_mem =       50, led0 = 0
Time = 2440ns, clk = 0, rst = 0, bit0 = 1, v_mem =       50, led0 = 0
Time = 2460ns, clk = 1, rst = 0, bit0 = 1, v_mem =        0, led0 = 0
Time = 2480ns, clk = 0, rst = 0, bit0 = 1, v_mem =        0, led0 = 0
Time = 2500ns, clk = 1, rst = 0, bit0 = 1, v_mem =        4, led0 = 1
Time = 2520ns, clk = 0, rst = 0, bit0 = 1, v_mem =        4, led0 = 1
Time = 2540ns, clk = 1, rst = 0, bit0 = 1, v_mem =        8, led0 = 0
Time = 2560ns, clk = 0, rst = 0, bit0 = 1, v_mem =        8, led0 = 0
Time = 2580ns, clk = 1, rst = 0, bit0 = 1, v_mem =       12, led0 = 0
Time = 2600ns, clk = 0, rst = 0, bit0 = 1, v_mem =       12, led0 = 0
Time = 2620ns, clk = 1, rst = 0, bit0 = 1, v_mem =       16, led0 = 0
Time = 2640ns, clk = 0, rst = 0, bit0 = 1, v_mem =       16, led0 = 0
Time = 2660ns, clk = 1, rst = 0, bit0 = 1, v_mem =       20, led0 = 0
Time = 2680ns, clk = 0, rst = 0, bit0 = 1, v_mem =       20, led0 = 0
Time = 2700ns, clk = 1, rst = 0, bit0 = 1, v_mem =       24, led0 = 0
Time = 2720ns, clk = 0, rst = 0, bit0 = 1, v_mem =       24, led0 = 0
Time = 2740ns, clk = 1, rst = 0, bit0 = 1, v_mem =       28, led0 = 0
Time = 2760ns, clk = 0, rst = 0, bit0 = 1, v_mem =       28, led0 = 0
Time = 2780ns, clk = 1, rst = 0, bit0 = 1, v_mem =       32, led0 = 0
Time = 2800ns, clk = 0, rst = 0, bit0 = 1, v_mem =       32, led0 = 0
Time = 2820ns, clk = 1, rst = 0, bit0 = 1, v_mem =       36, led0 = 0
Time = 2840ns, clk = 0, rst = 0, bit0 = 1, v_mem =       36, led0 = 0
Time = 2860ns, clk = 1, rst = 0, bit0 = 1, v_mem =       40, led0 = 0
Time = 2880ns, clk = 0, rst = 0, bit0 = 1, v_mem =       40, led0 = 0
Time = 2900ns, clk = 1, rst = 0, bit0 = 1, v_mem =       44, led0 = 0
Time = 2920ns, clk = 0, rst = 0, bit0 = 1, v_mem =       44, led0 = 0
Time = 2940ns, clk = 1, rst = 0, bit0 = 1, v_mem =       48, led0 = 0
Time = 2960ns, clk = 0, rst = 0, bit0 = 1, v_mem =       48, led0 = 0
Time = 2980ns, clk = 1, rst = 0, bit0 = 1, v_mem =       52, led0 = 0
Time = 3000ns, clk = 0, rst = 0, bit0 = 1, v_mem =       52, led0 = 0
Time = 3020ns, clk = 1, rst = 0, bit0 = 1, v_mem =        0, led0 = 0
Time = 3040ns, clk = 0, rst = 0, bit0 = 1, v_mem =        0, led0 = 0
```

```
Time = 3060ns, clk = 1, rst = 0, bit0 = 1, v_mem =  4, led0 = 1
Time = 3080ns, clk = 0, rst = 0, bit0 = 1, v_mem =  4, led0 = 1
Time = 3100ns, clk = 1, rst = 0, bit0 = 1, v_mem =  8, led0 = 0
Time = 3120ns, clk = 0, rst = 0, bit0 = 1, v_mem =  8, led0 = 0
Time = 3140ns, clk = 1, rst = 0, bit0 = 1, v_mem = 12, led0 = 0
Time = 3160ns, clk = 0, rst = 0, bit0 = 1, v_mem = 12, led0 = 0
Time = 3180ns, clk = 1, rst = 0, bit0 = 1, v_mem = 16, led0 = 0
Time = 3200ns, clk = 0, rst = 0, bit0 = 1, v_mem = 16, led0 = 0
Time = 3220ns, clk = 1, rst = 0, bit0 = 1, v_mem = 20, led0 = 0
Time = 3240ns, clk = 0, rst = 0, bit0 = 1, v_mem = 20, led0 = 0
Time = 3260ns, clk = 1, rst = 0, bit0 = 1, v_mem = 24, led0 = 0
Time = 3280ns, clk = 0, rst = 0, bit0 = 1, v_mem = 24, led0 = 0
Time = 3300ns, clk = 1, rst = 0, bit0 = 1, v_mem = 28, led0 = 0
Time = 3320ns, clk = 0, rst = 0, bit0 = 1, v_mem = 28, led0 = 0
Time = 3340ns, clk = 1, rst = 0, bit0 = 1, v_mem = 32, led0 = 0
Time = 3360ns, clk = 0, rst = 0, bit0 = 1, v_mem = 32, led0 = 0
Time = 3380ns, clk = 1, rst = 0, bit0 = 1, v_mem = 36, led0 = 0
Time = 3400ns, clk = 0, rst = 0, bit0 = 1, v_mem = 36, led0 = 0
Time = 3420ns, clk = 1, rst = 0, bit0 = 1, v_mem = 40, led0 = 0
Time = 3440ns, clk = 0, rst = 0, bit0 = 1, v_mem = 40, led0 = 0
Time = 3460ns, clk = 1, rst = 0, bit0 = 1, v_mem = 44, led0 = 0
Time = 3480ns, clk = 0, rst = 0, bit0 = 1, v_mem = 44, led0 = 0
Time = 3500ns, clk = 1, rst = 0, bit0 = 1, v_mem = 48, led0 = 0
Time = 3520ns, clk = 0, rst = 0, bit0 = 1, v_mem = 48, led0 = 0
Time = 3540ns, clk = 1, rst = 0, bit0 = 1, v_mem = 52, led0 = 0
Time = 3560ns, clk = 0, rst = 0, bit0 = 1, v_mem = 52, led0 = 0
Time = 3580ns, clk = 1, rst = 0, bit0 = 1, v_mem =  0, led0 = 0
Time = 3600ns, clk = 0, rst = 0, bit0 = 1, v_mem =  0, led0 = 0
Time = 3620ns, clk = 1, rst = 0, bit0 = 1, v_mem =  4, led0 = 1
Time = 3640ns, clk = 0, rst = 0, bit0 = 1, v_mem =  4, led0 = 1
Time = 3660ns, clk = 1, rst = 0, bit0 = 1, v_mem =  8, led0 = 0
Time = 3680ns, clk = 0, rst = 0, bit0 = 1, v_mem =  8, led0 = 0
Time = 3700ns, clk = 1, rst = 0, bit0 = 1, v_mem = 12, led0 = 0
Time = 3720ns, clk = 0, rst = 0, bit0 = 1, v_mem = 12, led0 = 0
Time = 3740ns, clk = 1, rst = 0, bit0 = 1, v_mem = 16, led0 = 0
Time = 3760ns, clk = 0, rst = 0, bit0 = 1, v_mem = 16, led0 = 0
Time = 3780ns, clk = 1, rst = 0, bit0 = 1, v_mem = 20, led0 = 0
Time = 3800ns, clk = 0, rst = 0, bit0 = 1, v_mem = 20, led0 = 0
Time = 3820ns, clk = 1, rst = 0, bit0 = 1, v_mem = 24, led0 = 0
Time = 3840ns, clk = 0, rst = 0, bit0 = 1, v_mem = 24, led0 = 0
Time = 3860ns, clk = 1, rst = 0, bit0 = 1, v_mem = 28, led0 = 0
Time = 3880ns, clk = 0, rst = 0, bit0 = 1, v_mem = 28, led0 = 0
Time = 3900ns, clk = 1, rst = 0, bit0 = 1, v_mem = 32, led0 = 0
Time = 3920ns, clk = 0, rst = 0, bit0 = 1, v_mem = 32, led0 = 0
Time = 3940ns, clk = 1, rst = 0, bit0 = 1, v_mem = 36, led0 = 0
Time = 3960ns, clk = 0, rst = 0, bit0 = 1, v_mem = 36, led0 = 0
Time = 3980ns, clk = 1, rst = 0, bit0 = 1, v_mem = 40, led0 = 0
Time = 4000ns, clk = 0, rst = 0, bit0 = 1, v_mem = 40, led0 = 0
Time = 4020ns, clk = 1, rst = 0, bit0 = 1, v_mem = 44, led0 = 0
```

- Pada simulasi, diberikan input bit0= 1 selama 1000 unit waktu yang mengakibatkan v_mem terus bertambah namun saat v_mem melebihi 50 maka v_mem akan reset ke nilai 0 dan diikuti dengan perubahan nilai led0 menjadi 1 dengan delay 2 clock
- Kemudian untuk unit waktu 10011 hingga 2010 bit0=0 yang mengakibatkan nilai v_mem terus menurun
- lalu untuk unit waktu 2011 hingga 4010 bit0 dikembalikan lagi bernilai 1 yang menghasilkan hasil yang sama pada unit waktu 11-1010

4. Hasil Gelombang Simulasi dengan menggunakan GTKWave



- hasil graphic meunjukkan spike hanya terjadi pada saat bit0 bernillai 1 dan tidak terjadi spike pada saat bit0 bernilai 0

C. Implementasi FPGA dengan CMOD A7

1. Input digital ESP32

```
const int bit0 = 25;

// Interval untuk setiap pin dalam milidetik
const unsigned long interval0 = 20;  // 50Hz

// Waktu terakhir berubah untuk setiap pin
unsigned long previousMillis0 = 0;

void setup() {
  Serial.begin(9600);
  pinMode(bit0, OUTPUT);

  // Inisialisasi seed untuk random number generator
  randomSeed(analogRead(0));
}

void loop() {
  unsigned long currentMillis = millis();

  // bit0
```

```
  if (currentMillis - previousMillis0 >= interval0) {
    previousMillis0 = currentMillis;
    int randomBit = random(2); // Menghasilkan nilai acak 0 atau 1
    Serial.println(randomBit);
    digitalWrite(bit0, randomBit);
  }
}
```

Gelombang Input random bit (0 or 1) pada ESP32

Gambar 3. Input spike random 01 FPGA dengan ESP32.

Program Arduino ini menghasilkan gelombang kotak acak dengan nilai 0 dan 1 pada pin digital 25 dengan frekuensi 50 Hz. Berikut adalah hasil analisis dari program tersebut:

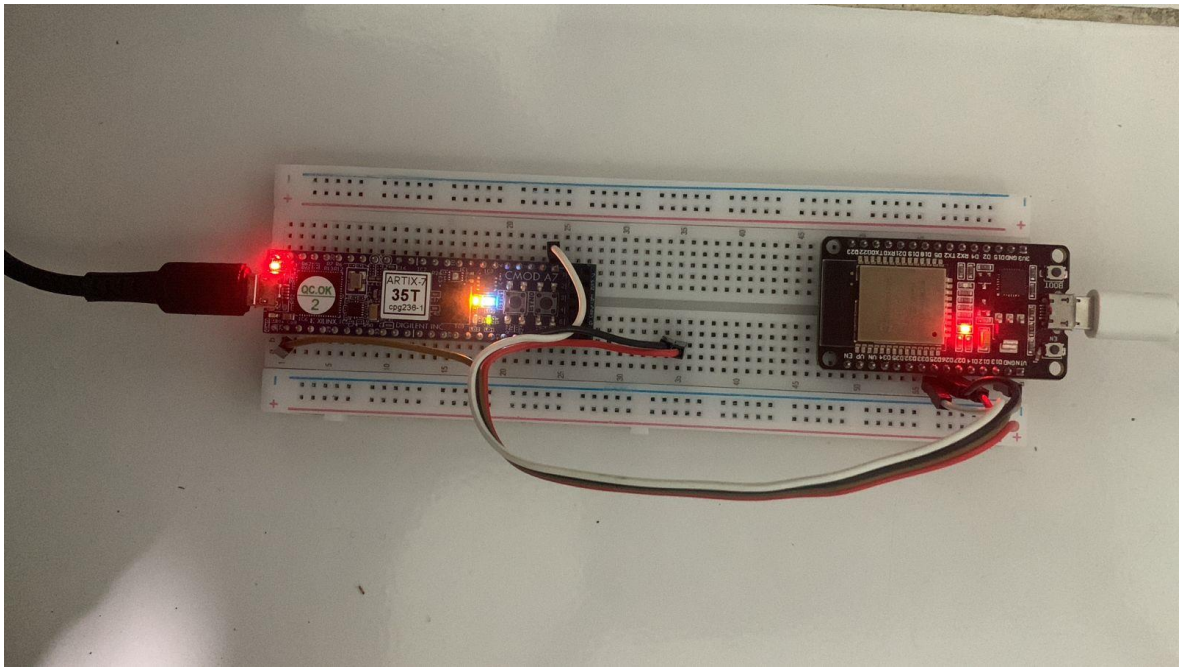
- Frekuensi Gelombang Kotak:
  Interval waktu yang digunakan adalah 20 ms, yang berarti frekuensi gelombang adalah 1 / 0.02 detik = 50 Hz. Ini sesuai dengan frekuensi yang diinginkan.
- Penggunaan Fungsi random:
  Fungsi random(2) digunakan untuk menghasilkan nilai acak antara 0 dan 1. Setiap 20 ms, nilai ini di output kan ke pin digital 25 dan dicetak ke serial monitor. Ini memastikan bahwa keluaran gelombang kotak benar-benar acak.

2. Program FPGA

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
```

```
// Company:
// Engineer:
//
// Create Date: 08.07.2024 20:13:00
// Design Name:
// Module Name: lif
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module lif(
    input wire clk,
    input wire rst,

    input wire bit0,
    output reg led0
    );

    // Define constants as integers scaled by a factor of 1e6 (for example)
    parameter integer TIME_STEP = 1;
    parameter integer V_TH = 50;
    parameter integer V_RESET = 0;
    parameter integer WEIGHT = 5;
    parameter integer LEAK = 1;

    parameter integer RESISTOR = 1; // 1 (1 Ohm) scaled by 1e6
    parameter integer CAPACITOR = 1; // 1 (1 Farad) scaled by 1e6
    parameter integer TAU = (RESISTOR * CAPACITOR); // tau = R * C scaled by 1e6

    integer v_mem;
    reg input_current;
    reg spike;

    always @(posedge clk or posedge rst) begin
```

```
        if (rst) begin
            v_mem <= V_RESET;
            spike <= 0;
        end else begin
            if (bit0 == 1) begin
                input_current <= 1;
            end else if (bit0 == 0) begin
                input_current <= 0;
            end

//                  v_mem <= v_mem + (TIME_STEP / TAU) * (-v_mem + (input_current *
WEIGHT));
            v_mem <= v_mem + (TIME_STEP / TAU) * (-LEAK + (input_current * WEIGHT));

            if (v_mem >= V_TH) begin
                spike <= 1;
                v_mem <= V_RESET;
            end else begin
                spike <= 0;
            end

            led0 <= spike;
        end
    end
endmodule
```



Gambar 4. Rangkaian FPGA dan ESP32.

Program FPGA menggunakan Logika LIF (Leaky Integrate and Fire) untuk memproses sinyal dari Arduino. Berikut adalah analisis dari implementasi ini:

- Model Integrate and Fire:
  Potensial memori sekarang ($u(t + \Delta t)$) diperbarui setiap siklus clock. Persamaan yang digunakan adalah:

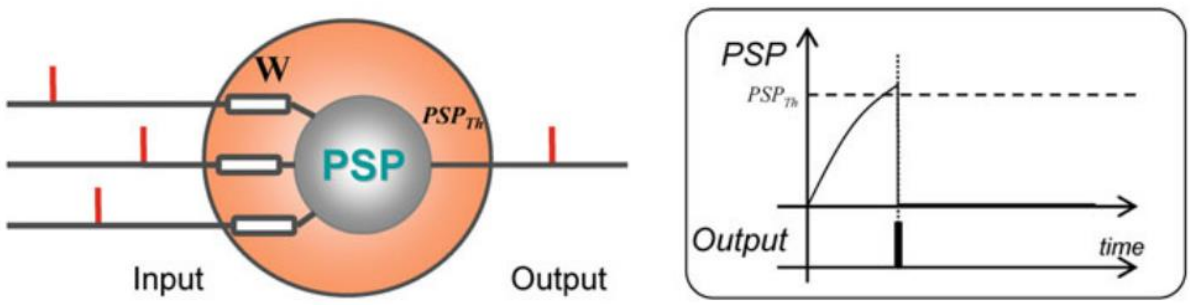
$$U(t + \Delta t) = U(t) + \frac{\Delta t}{\tau}\big(- U(t) + RI_{\text{in}}(t)\big)$$

  TIME_STEP (Δt) adalah langkah waktu simulasi.
  TAU (τ) adalah konstanta waktu yang dihitung sebagai perkalian dari resistansi dan kapasitansi.
  Potensial memori sebelumnya ($u(t)$) adalah nilai hasil penyimpanan memori tegangan sebelumnya.
  LEAK (-$u(t)$) adalah konstanta kebocoran yang mengurangi potensial memori seiring waktu.
  WEIGHT (R) adalah konstanta yang menentukan kontribusi input terhadap potensial memori.
  Input Current (I_{in}(t)) adalah variabel yang digunakan untuk merepresentasikan arus masukan yang dihasilkan dari sinyal input (0 atau 1).
- Inisialisasi dan Reset:
  Saat reset (rst) diaktifkan, v_mem diatur ke V_RESET dan spike diatur ke 0 (0Volt). Ini memastikan bahwa memori potensial neuron dimulai dari nilai dasar 0v atau start tegangan.
- Pemrosesan Input:
  input_current diatur berdasarkan nilai bit0 dari ESP32. Jika bit0 adalah 1, maka arus input diatur ke 1, dan sebaliknya.

Kesimpulan : Jika v_mem melebihi ambang batas (V_TH), spike dihasilkan (led0 diatur ke 1 atau led0 menyala), dan v_mem direset ke V_RESET. Jika tidak, spike diatur ke 0 atau led0 mati.

D. Tugas Tambahan

Ketika diberikan dua atau lebih input pada satu neuron, maka pemrosesan spike yang terjadi pada neuron tersebut akan berbeda dari neuron dengan satu input.



Gambar 5. Ilustrasi neuron dengan dua atau lebih input. (Sumber: Kasabov, 2019))

Apabila memiliki satu input atau lebih, pada masing-masing input spike akan memiliki weight yang berbeda. Diketahui, rumus dan model dari LIF neuron secara matematis dapat dituliskan sebagai:

$$U(t + \Delta t) = u(t) + \frac{\Delta t}{t}(-U(t) + RI_{in}(t)$$

Untuk nilai I yang lebih dari satu, maka diketahui juga weight memiliki jumlah lebih dari satu (untuk setiap I, terdapat R sebagai weight-nya). Maka, rumus yang digunakan adalah:

$$U(t + \Delta t) = u(t) + \frac{\Delta t}{t}(-U(t) + \sum_{n=1}^{i} R_n(I_{in}(t))_n$$

Yang mana, untuk keseluruhan input I dikalikan dengan nilai weightnya dan dijumlahkan.

Berdasarkan rumus ini, dapat diimplementasikan langsung dengan kode verilog, dengan mengganti rumus atau model LIF neuron pada program dan memberikan input sinyal yang lain pada testbench.

Implementasi FPGA dengan modifikasi 2 buah input 1 neuron

```
parameter integer WEIGHT0 = 5;
parameter integer WEIGHT1 = 3;

integer i; reg [1:0] bit_array;
integer                                                    weights[1:0];

initial begin
  weights[0] = WEIGHT0;
  weights[1] = WEIGHT1;
 end

always @(posedge clk or posedge rst) begin
      if (rst) begin
         v_mem <= V_RESET;
         spike <= 0;
      end else begin
         bit_array[0] = bit0;
         bit_array[1] = bit1;

         input_current <= 0;

         for (i = 0; i < 2; i = i + 1) begin
            if (bit_array[i] == 1) begin
               input_current[i] <= 1;
```

```
            end else begin
               input_current[i] <= 0;
            end
         end

         v_mem <= v_mem + (TIME_STEP / TAU) * (-LEAK + (input_current[0] *
weights[0]) + (input_current[1] * weights[1]));

         if (v_mem >= V_TH) begin
            spike <= 1;
            v_mem <= V_RESET;
         end else begin
            spike <= 0;
         end

         led0 <= spike;
      end
   end
```

REFERENSI

Kasabov, N. K. (2019). *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence* (Vol. 7). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-57715-8

snnTorch. (n.d.). Tutorial 2. *snntorch.readthedocs.io*. Retrieved July 12, 2024, from https://snntorch.readthedocs.io/en/latest/tutorials/tutorial_2.html