Telkom University

Fakultas Teknik Elektro
Telkom University

TUGAS BESAR MATA KULIAH
DESAIN SOC & FPGA

TRAFFIC CONTROLLER MENGGUNAKAN FPGA DENGAN TAMBAHAN SEVEN SEGMENT DAN TOMBOL PEJALAN KAKI

- Tengku Abdurrazak Johan | 1102223123
- Nabil Al-Faiq Rinovka | 1102223150
- Nazer Muhammad Noor | 1102223120

Made By : Kelompok 4

DESKRIPSI TUGAS BESAR

- **Lampu lalu lintas** merupakan sistem kendali digital berbasis logika sekuensial
- FPGA cocok untuk implementasi sistem kendali **real-time**
- Finite State Machine (FSM) digunakan untuk **mengatur urutan** lampu lalu lintas
- Penambahan **seven segment** dan **tombol** pedestrian meningkatkan fungsionalitas sistem

TUJUAN SISTEM

- Merancang **traffic light** controller berbasis FPGA
- Mengimplementasikan FSM untuk **pengaturan** lampu lalu lintas
- Menampilkan hitung mundur fase lampu menggunakan **seven segment**
- Menyediakan **fitur pedestrian crossing** menggunakan push button

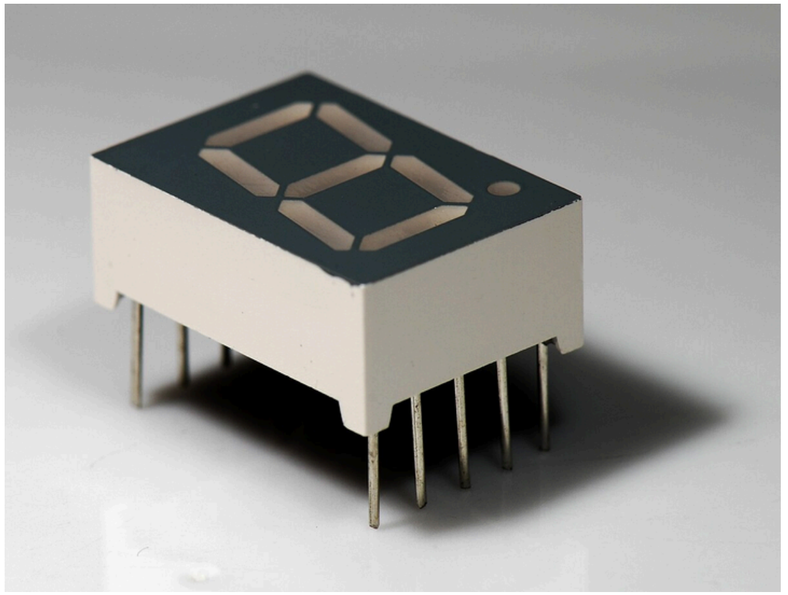
Telkom University | Fakultas Teknik Elektro Telkom University

FUNGSI SISTEM

- Mengatur urutan lampu merah, kuning, dan hijau secara otomatis
- Mengendalikan durasi tiap fase lampu menggunakan clock FPGA
- Memproses input tombol pejalan kaki
- Menampilkan status atau countdown fase lampu

FITUR UTAMA

- Traffic controller berbasis FPGA
- Lampu lalu lintas (LED merah, kuning, hijau)
- Seven segment display untuk countdown
- Tombol pejalan kaki (pedestrian button)
- Reset sistem



Telkom University | Fakultas Teknik Elektro Telkom University

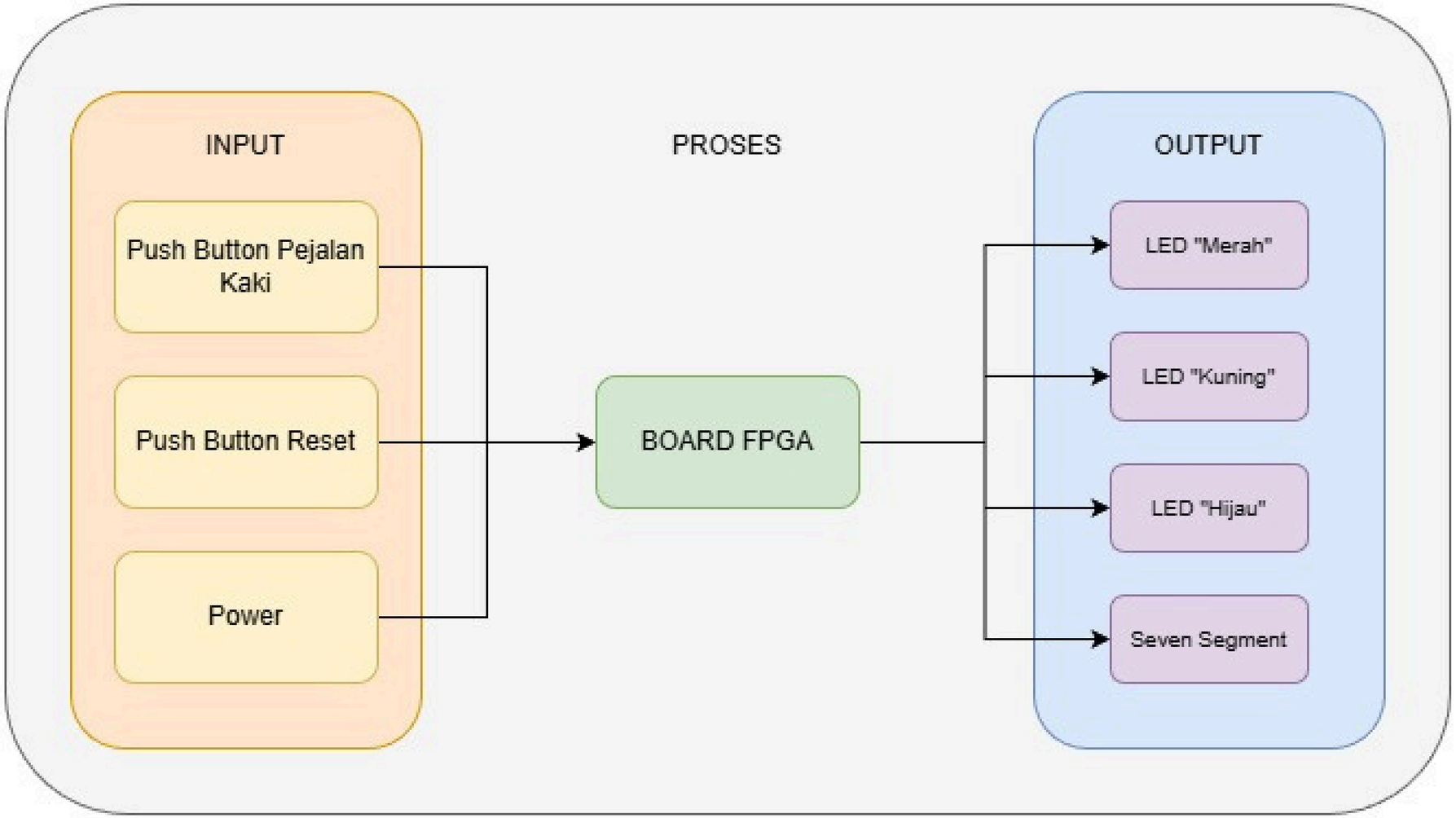
SPESIFIKASI SISTEM

- FPGA sebagai pengendali utama
- Logika kendali menggunakan Finite State Machine (FSM)
- Clock internal FPGA sebagai basis waktu
- Output visual: LED dan seven segment
- Input: push button pedestrian dan reset

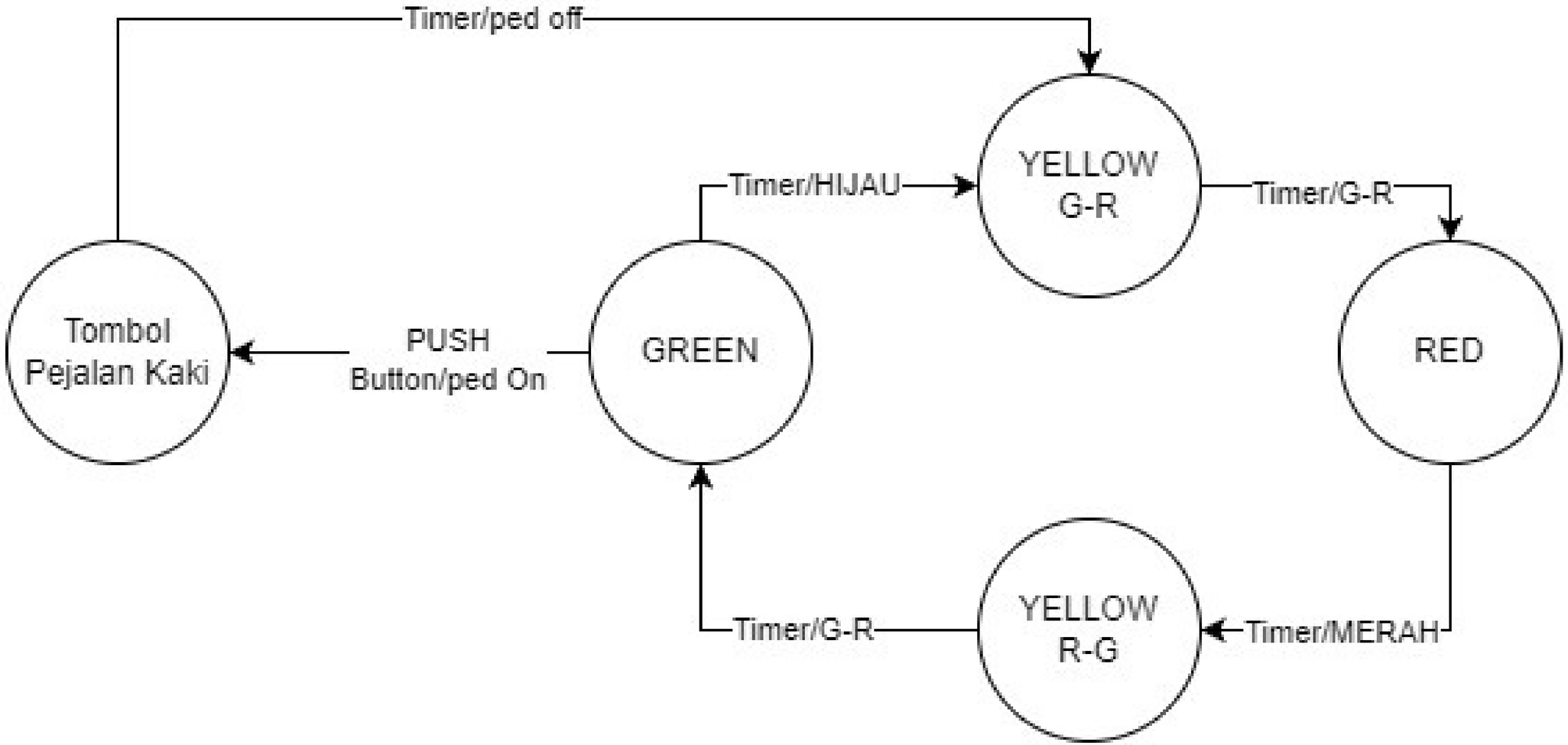
CARA KERJA SISTEM

- Sistem aktif saat FPGA diberi catu daya
- FPGA melakukan inisialisasi dan masuk state awal (lampu merah)
- Counter internal menghitung durasi tiap fase
- FSM mengatur perpindahan state berdasarkan timer dan input
- Seven segment menampilkan countdown fase aktif
- Tombol pedestrian dapat memicu fase penyeberangan

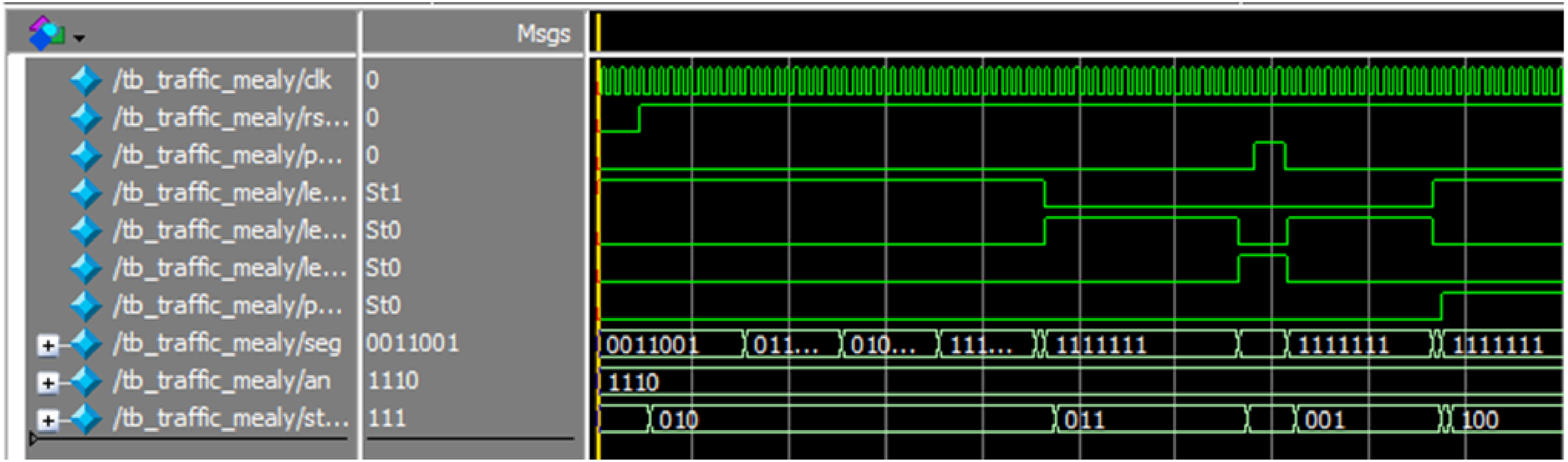
BLOCK DIAGRAM SYSTEM



FSM DIAGRAM


Timer/ped off
YELLOW
G-R
Timer/HIJAU
Timer/G-R
Tombol
Pejalan Kaki
PUSH
Button/ped On
GREEN
RED
YELLOW
R-G
Timer/G-R
Timer/MERAH

FSM DIAGRAM


Msgs
/tb_traffic_mealy/clk
/tb_traffic_mealy/rs...
/tb_traffic_mealy/p...
/tb_traffic_mealy/le...
/tb_traffic_mealy/le...
/tb_traffic_mealy/le...
/tb_traffic_mealy/p...
/tb_traffic_mealy/seg
/tb_traffic_mealy/an
/tb_traffic_mealy/st...
0
0
0
St1
St0
St0
St0
0011001
1110
111
0011001
011...
010...
111...
1111111
1111111
1111111
1110
010
011
001
100

Telkom University

Fakultas Teknik Elektro
Telkom University

LAMPIRAN KODE TOP LEVEL

```
module traffic_mealy #(
    parameter integer CLK_HZ     = 50_000_000,
    parameter integer T_GREEN    = 9,    // detik green
    parameter integer T_RED      = 9,    // detik red
    parameter integer T_YELLOW   = 2,    // detik yellow (sesuai kode Anda sekarang)
    parameter integer T_PED_WALK= 6     // durasi WALK saat kendaraan red
)(
    input  wire clk,
    input  wire rst_n,

    input  wire ped_btn,          // tombol pedestrian (asinkron)

    output reg  led_r,
    output reg  led_y,
    output reg  led_g,
    output reg  ped_walk,         // indikator WALK

    output wire [6:0] seg,        // abcdefg (aktif-low untuk hardware)
    output wire [3:0] an
);

    // ========================
    // 1) Tick 1 detik
    // ========================
    reg [31:0] div_cnt;
    reg tick_1s;

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            div_cnt <= 0;
            tick_1s <= 1'b0;
        end else begin
            if (div_cnt == (CLK_HZ-1)) begin
                div_cnt <= 0;
                tick_1s <= 1'b1;
            end else begin
                div_cnt <= div_cnt + 1;
                tick_1s <= 1'b0;
            end
        end
    end

    // ========================================
    // 2) Sinkronisasi tombol + edge detect
    // ========================================
    reg ped_ff1, ped_ff2, ped_ff2_d;
    wire ped_rise;

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            ped_ff1   <= 1'b0;
            ped_ff2   <= 1'b0;
            ped_ff2_d <= 1'b0;
        end else begin
            ped_ff1   <= ped_btn;
            ped_ff2   <= ped_ff1;
            ped_ff2_d <= ped_ff2;
        end
    end

    assign ped_rise = ped_ff2 & ~ped_ff2_d;

    // ========================
    // 3) Latch request pedestrian
    // ========================
    reg ped_req;

    // ========================
    // 4) FSM State
    // ========================
    localparam S_GREEN = 3'd0;
    localparam S_Y_G2R = 3'd1;
    localparam S_RED   = 3'd2;
    localparam S_Y_R2G = 3'd3;
    localparam S_PED   = 3'd4;

    reg [2:0] state, next_state;

    // countdown detik
    reg [7:0] seconds_left;
    wire timer_done = (seconds_left == 0);

    function [7:0] duration_for_state(input [2:0] st, input ped_req_i);
        begin
            case (st)
                S_GREEN: duration_for_state = T_GREEN[7:0];
                S_Y_G2R: duration_for_state = T_YELLOW[7:0];
                S_RED:   duration_for_state = T_RED[7:0];
                S_Y_R2G: duration_for_state = T_YELLOW[7:0];
                S_PED:   duration_for_state = (ped_req_i) ? T_PED_WALK[7:0
                default: duration_for_state = T_RED[7:0];
            endcase
        end
    endfunction

    // ================================
    // 5) Next-state logic (Mealy)
    // ================================
    always @(*) begin
        next_state = state;
        case (state)
            S_GREEN:  if (timer_done || ped_req) next_state = S_Y_G2R;
            S_Y_G2R:  if (timer_done)            next_state = S_RED;
            S_RED:    if (ped_req)               next_state = S_PED;
                      else if (timer_done)       next_state = S_Y_R2G;
            S_PED:    if (timer_done)            next_state = S_Y_R2G;
            S_Y_R2G:  if (timer_done)            next_state = S_GREEN;
            default:                             next_state = S_RED;
        endcase
    end

    // ================================
    // 6) State register + timer handling
    // ================================
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            state        <= S_RED;
            seconds_left <= duration_for_state(S_RED, 1'b0);
            ped_req      <= 1'b0;
        end else begin
            if (ped_rise) ped_req <= 1'b1;

            if (state != next_state) begin
                // clear ped_req setelah selesai fase pedestrian
                if (state == S_PED && next_state == S_Y_R2G)
                    ped_req <= 1'b0;

                state        <= next_state;
                seconds_left <= duration_for_state(next_state, ped_req);
            end else begin
                if (tick_1s) begin
                    if (seconds_left != 0)
                        seconds_left <= seconds_left - 1;
                end
            end
        end
    end

    // ================================
    // 7) Output LED
    // ================================
    always @(*) begin
        led_r    = 1'b0;
        led_y    = 1'b0;
        led_g    = 1'b0;
        ped_walk = 1'b0;

        case (state)
            S_GREEN: begin
                led_g = 1'b1;
            end
            S_Y_G2R, S_Y_R2G: begin
                led_y = 1'b1;
            end
            S_RED: begin
                led_r = 1'b1;
            end
            S_PED: begin
                led_r    = 1'b1;
                ped_walk = 1'b1;
            end
            default: begin
                led_r = 1'b1;
            end
        endcase
    end

    // ============================================================
    // 8) 7-seg countdown: tampil ONLY saat GREEN dan RED
    //    - saat YELLOW/PED: BLANK (mati)
    //    - seg output aktif-low (0 = nyala)
    // ============================================================
    wire show_timer = (state == S_GREEN) || (state == S_RED);
    wire [3:0] digit = seconds_left % 10;

    wire [6:0] seg_active_low;
    sevenseg_1digit_active_low u7 (
        .bin(digit),
        .blank(~show_timer),        // blank saat bukan green/red
        .seg(seg_active_low)
    );

    assign seg = seg_active_low;

    // untuk kompatibilitas port Anda (di DE1-SoC ini sebenarnya tidak dipakai)
    assign an = 4'b1110;

endmodule
```

LAMPIRAN KODE SUB-LEVEL

```
1  module sevenseg_1digit_active_low(
2      input  wire [3:0] bin,
3      input  wire blank,        // 1 = mati semua segmen
4      output reg  [6:0] seg     // abcdefg, aktif-low
5  );
6      always @(*) begin
7          if (blank) begin
8              seg = 7'b1111111;  // semua mati
9          end else begin
10             case (bin)
11                 4'd0: seg = 7'b1000000;
12                 4'd1: seg = 7'b1111001;
13                 4'd2: seg = 7'b0100100;
14                 4'd3: seg = 7'b0110000;
15                 4'd4: seg = 7'b0011001;
16                 4'd5: seg = 7'b0010010;
17                 4'd6: seg = 7'b0000010;
18                 4'd7: seg = 7'b1111000;
19                 4'd8: seg = 7'b0000000;
20                 4'd9: seg = 7'b0010000;
21                 default: seg = 7'b1111111;
22             endcase
23         end
24     end
25 endmodule
26
```

DEMONSTRASI ALAT

THANK YOU!

Telkom University

Fakultas Teknik Elektro
Telkom University


Panasonic
ANALOG DEVICES
ISSI
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com