3300 Lab 8
RGB LED PWM Controller

Group C
Rohan Walia
Parsa Ghasemi
8/14/25

```
// top_lab8.v
`timescale 1ns/1ps
module top_lab8(
    input wire clk100mhz,
    input wire btnc_n,
    input wire btnr,
    input wire [7:0] sw,
    output wire [3:0] led,
    output wire rgb_r, rgb_g, rgb_b
);
    wire rst_n = ~btnc_n;
    wire clk_1k, clk_pwm;

    clock_divider_fixed u_div(
        .clk_in(clk100mhz),
        .rst_n(rst_n),
        .clk_1k(clk_1k),
        .clk_pwm(clk_pwm)
    );

    wire load_pulse;

    debounce_onepulse #(.STABLE_TICKS(20)) u_db(
        .clk(clk_1k),
        .rst_n(rst_n),
        .din(btnr),
        .pulse(load_pulse)
    );

    wire [1:0] slot;
    wire [3:0] slot_oh;
    wire wr_res, wr_r, wr_g, wr_b;

    load_fsm u_fsm(
        .clk(clk_1k),
        .rst_n(rst_n),
        .load_pulse(load_pulse),
        .slot(slot),
        .slot_onehot(slot_oh),
        .wr_res(wr_res),
        .wr_r(wr_r),
        .wr_g(wr_g),
        .wr_b(wr_b)
    );

    assign led = slot_oh;

    reg [7:0] reg_res, reg_r, reg_g, reg_b;

    always @(posedge clk_1k or negedge rst_n) begin
        if (!rst_n) begin reg_res <= 8'd0; reg_r <= 8'd0; reg_g <= 8'd0; reg_b <= 8'd0; end
        else begin
            if (wr_res) reg_res <= sw;
            if (wr_r) reg_r <= sw;
            if (wr_g) reg_g <= sw;
            if (wr_b) reg_b <= sw;
        end
    end

    reg [7:0] res_q1, res_q2, r_q1, r_q2, g_q1, g_q2, b_q1, b_q2;
    always @(posedge clk_pwm or negedge rst_n) begin
        if (!rst_n) begin
            res_q1 <= 0;
            res_q2 <= 0;
            r_q1 <= 0;
            r_q2 <= 0;
            g_q1 <= 0;
            g_q2 <= 0;
            b_q1 <= 0;
            b_q2 <= 0;
        end
        else begin
            res_q1 <= reg_res;
            res_q2 <= res_q1;
            r_q1 <= reg_r;
            r_q2 <= r_q1;
            g_q1 <= reg_g;
            g_q2 <= g_q1;
            b_q1 <= reg_b;
            b_q2 <= b_q1;
        end
    end

    wire pwm_r, pwm_g, pwm_b;

    pwm_core u_pwm(
        .clk(clk_pwm),
        .rst_n(rst_n),
        .period(res_q2),
        .duty_r(r_q2),
        .duty_g(g_q2),
        .duty_b(b_q2),
        .pwm_r(pwm_r),
        .pwm_g(pwm_g),
        .pwm_b(pwm_b)
    );

    rgb_led_driver #(.ACTIVE_LOW(0)) u_led(
        .pwm_r(pwm_r),
        .pwm_g(pwm_g),
        .pwm_b(pwm_b),
        .led_r(rgb_r),
        .led_g(rgb_g),
        .led_b(rgb_b)
    );
endmodule




// clock_divider_fixed.v
module clock_divider_fixed #(
    parameter integer INPUT_HZ = 100_000_000,
    parameter integer TICK1_HZ = 1_000,
    parameter integer PWM_HZ = 20_000
)(

    input wire clk_in,
    input wire rst_n,
    output reg clk_1k,
    output reg clk_pwm
);

    localparam integer DIV1H = (INPUT_HZ/TICK1_HZ)/2;
    localparam integer DIVPMH = (INPUT_HZ/PWM_HZ)/2;
    reg [$clog2(DIV1H):0] c1;
    reg [$clog2(DIVPMH):0] c2;

    always @(posedge clk_in or negedge rst_n) begin
        if (!rst_n) begin
            c1 <= 0;
            clk_1k <= 0;
            c2 <= 0;
            clk_pwm <= 0;
            end
        else begin
        if (c1 == DIV1H-1) begin
            c1 <= 0;
            clk_1k <= ~clk_1k;
            end
        else
            c1 <= c1+1;
        if (c2 == DIVPMH-1) begin
            c2 <= 0;
            clk_pwm <= ~clk_pwm;
            end
        else
            c2 <= c2+1;
        end
    end
endmodule




// debounce_onepulse.v
`timescale 1ns/1ps
module debounce_onepulse #(
    parameter integer STABLE_TICKS = 20
)(
    input wire clk,
    input wire rst_n,
    input wire din,
    output reg pulse
);
    reg d0, d1;
    reg stable, stable_q;
    reg [$clog2(STABLE_TICKS+1)-1:0] cnt;

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin d0<=0; d1<=0; end else begin d0<=din; d1<=d0; end
    end

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin cnt<=0; stable<=0; end
        else if (d1 != stable) begin
            if (cnt==STABLE_TICKS) begin stable<=d1; cnt<=0; end
            else cnt<=cnt+1;
        end else cnt<=0;
    end

    always @(posedge clk or negedge rst_n) begin
```

```
        if (!rst_n) begin stable_q<=0; pulse<=0; end
        else begin pulse <= (~stable_q) & stable; stable_q <= stable; end
    end
endmodule
```

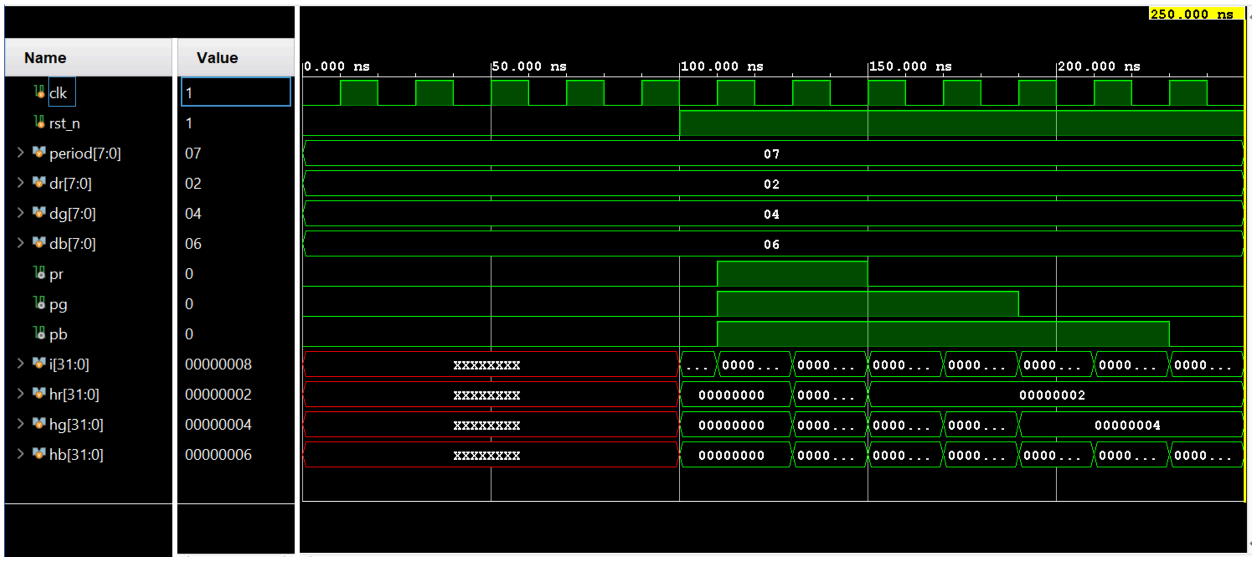
```
// load_fsm.v
`timescale 1ns/1ps
module load_fsm(
    input wire clk,
    input wire rst_n,
    input wire load_pulse,
    output reg [1:0] slot,
    output wire [3:0] slot_onehot,
    output reg wr_res, wr_r, wr_g, wr_b
);
    assign slot_onehot = 4'b0001 << slot;

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) slot <= 2'd0;
        else if (load_pulse) slot <= slot + 2'd1;
    end

    always @* begin
        wr_res = 0; wr_r = 0; wr_g = 0; wr_b = 0;
        case (slot)
            2'd0: wr_res = load_pulse;
            2'd1: wr_r = load_pulse;
            2'd2: wr_g = load_pulse;
            2'd3: wr_b = load_pulse;
        endcase
    end
endmodule
```

```
// pwm_core.v
`timescale 1ns/1ps
module pwm_core(
    input wire clk,
    input wire rst_n,
    input wire [7:0] period,
    input wire [7:0] duty_r, duty_g, duty_b,
    output reg pwm_r, pwm_g, pwm_b
);
    wire [8:0] eff_period = {1'b0, period} + 9'd1;

    function [8:0] clamp9(input [7:0] d);
        clamp9 = ( {1'b0,d} >= eff_period ) ? (eff_period - 9'd1) : {1'b0,d};
    endfunction

    reg [8:0] cnt;

    always @(posedge clk or negedge rst_n) begin
```

```
        if (!rst_n) cnt <= 0;
        else if (cnt == eff_period - 1) cnt <= 0;
        else cnt <= cnt + 1;
    end

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) {pwm_r, pwm_g, pwm_b} <= 0;
        else begin
            pwm_r <= (cnt < clamp9(duty_r));
            pwm_g <= (cnt < clamp9(duty_g));
            pwm_b <= (cnt < clamp9(duty_b));
        end
    end
endmodule
```

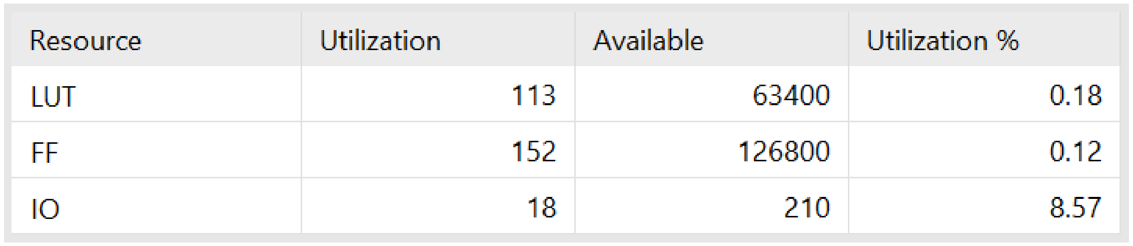
```
// rgb_led_driver.v
`timescale 1ns/1ps
module rgb_led_driver #(parameter ACTIVE_LOW=0)(
    input wire pwm_r, pwm_g, pwm_b,
    output wire led_r, led_g, led_b
);
    generate
        if (ACTIVE_LOW) begin
            assign led_r = ~pwm_r;
            assign led_g = ~pwm_g;
            assign led_b = ~pwm_b;
        end else begin
            assign led_r = pwm_r;
            assign led_g = pwm_g;
            assign led_b = pwm_b;
        end
    endgenerate
endmodule
```

pwm_core_tb

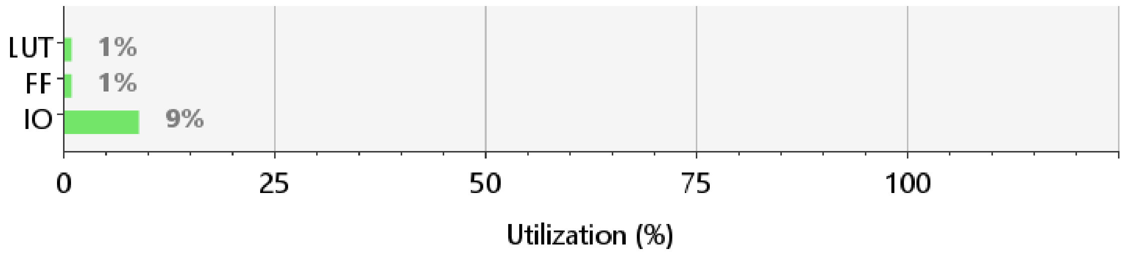


Name	Value
clk	1
rst_n	1
period[7:0]	07
dr[7:0]	02
dg[7:0]	04
db[7:0]	06
pr	0
pg	0
pb	0
i[31:0]	00000008
hr[31:0]	00000002
hg[31:0]	00000004
hb[31:0]	00000006

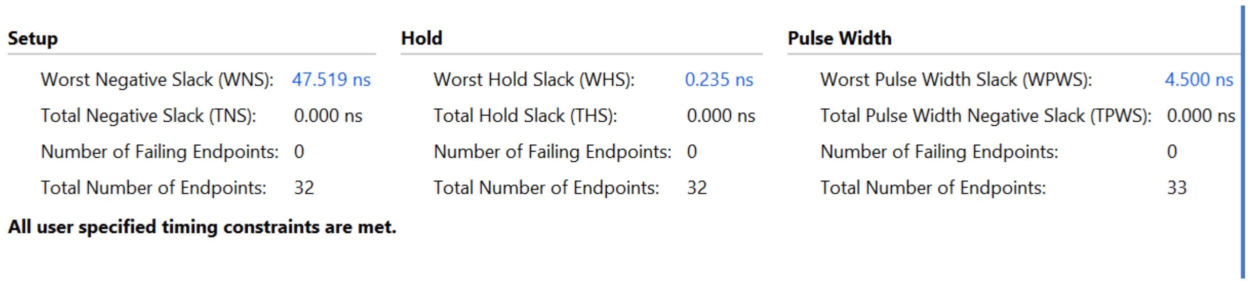
LUTs and FF screenshot:



Resource	Utilization	Available	Utilization %
LUT	113	63400	0.18
FF	152	126800	0.12
IO	18	210	8.57



Timing screenshot:



Setup

Worst Negative Slack (WNS):	47.519 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	32

Hold

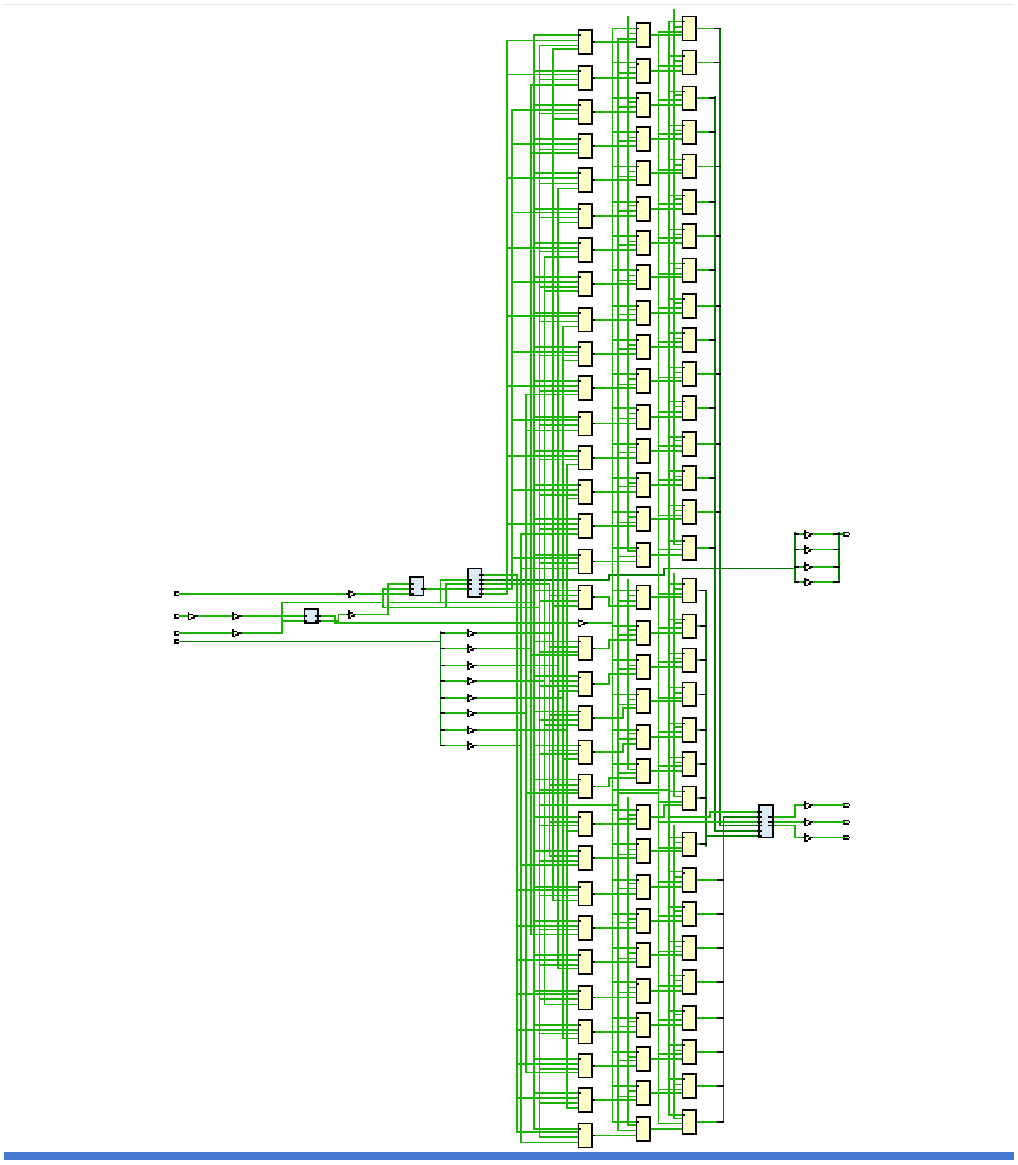
Worst Hold Slack (WHS):	0.235 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	32

Pulse Width

Worst Pulse Width Slack (WPWS):	4.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	33

All user specified timing constraints are met.

Block diagram:



Contribution:
Rohan Walia (50%): Implementation, demo, verilog, report
Parsa Ghasemi (50%): testbench code, testing, report