**REPORT TRAFFIC LIGHT**

A close-up of a document

AI-generated content may be incorrect.

* **CHI TIẾT PERFORMANCE**

- Tần số Clock: 10 MHz (chu kỳ 100ns)

- Độ trễ hệ thống: 2 chu kỳ clock cho mỗi lần chuyển trạng thái

- Cập nhật Display: 0 chu kỳ clock (logic tổ hợp)

- Tiêu thụ điện năng: Tối thiểu (máy trạng thái hiệu quả)

- Sử dụng tài nguyên: Thấp (bộ đếm và logic đơn giản)

* **Mục tiêu thiết kế:**

-An toàn giao thông: Đảm bảo phương tiện di chuyển an toàn

- Tự động hóa: Hoạt động độc lập không cần can thiệp thủ công

- Hiển thị thời gian: Cho biết thời gian còn lại của mỗi pha

- Độ tin cậy cao: Hoạt động ổn định trong mọi điều kiện

* **Nguyên lý hoạt động :**

Hệ thống chuyển đổi 3 trạng thái LED, Green, Yellow, Red với trạng thái mặc định là Green. Các led sẽ hiển thị màu cùng với khoảng thời gian đếm ngược của từng led lên LED 7 đoạn với led green 15s đếm ngược sau đó chuyển sang yellow với 3s đếm ngược sau đó chuyển sang red với 18s đếm ngược rồi quay lại với green để tiếp tục vòng lặp.

* **Tín hiệu vào**

- `clk`: Clock 10MHz (100ns period)

- `rst\_n`: Reset tích cực mức thấp (asynchronous)

* **Tín hiệu ra**

- `red\_light`: Điều khiển đèn đỏ (1=ON, 0=OFF)

- `yellow\_light`: Điều khiển đèn vàng (1=ON, 0=OFF)

- `green\_light`: Điều khiển đèn xanh (1=ON, 0=OFF)

- `display\_led`: Hiển thị LED 7 đoạn [15:0] (2 chữ số)

- `timer\_load`: Tín hiệu nạp timer (debug)

- `timer`: Giá trị timer hiện tại [31:0] (debug)

* **CẤU TRÚC PROJECT**
* **Các module chính**

- `traffic\_light.v`: Module top-level, tích hợp tất cả các module con

- `traffic\_light\_fsm.v`: Máy trạng thái quản lý chuyển đổi giữa các trạng thái

- `timer.v`: Bộ đếm thời gian cho mỗi trạng thái

* **Testbench**

- `tb\_traffic\_light.v`: Testbench cho module top-level

- `traffic\_light\_fsm\_tb.v`: Testbench cho module FSM

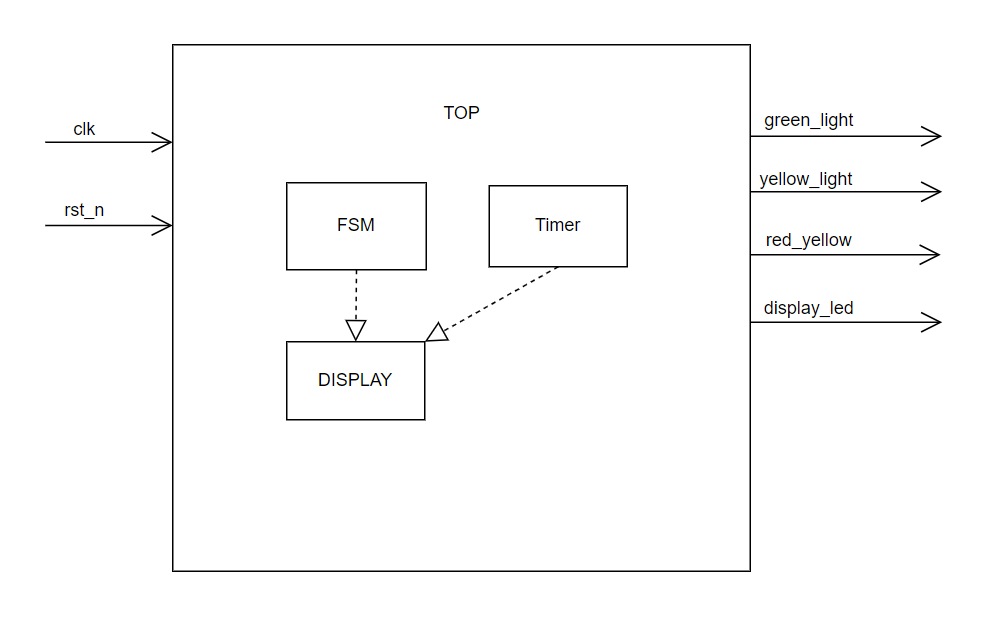
- `tb\_timer.v`: Testbench cho module timer

* **Các file phụ trợ**

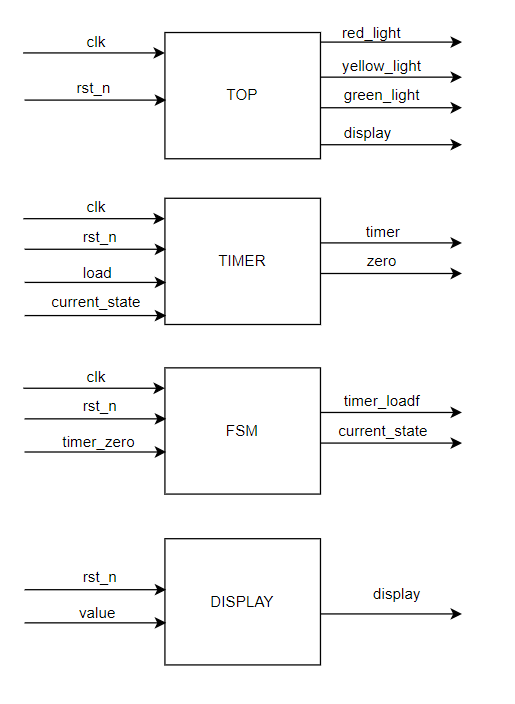
- `run\_traffic\_light\_sim.sh`: Script chạy mô phỏng đầy đủ

- `run\_traffic\_quick.sh`: Script chạy mô phỏng nhanh

* **Sơ tổng quát của hệ thống :**



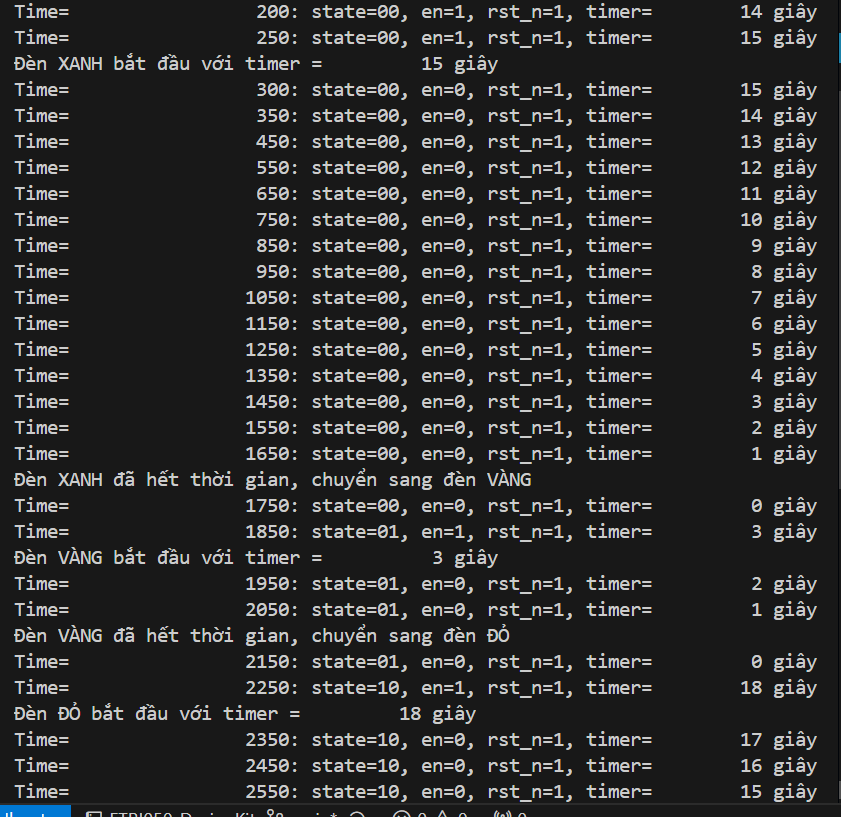
* **Sơ đồ các khối :**

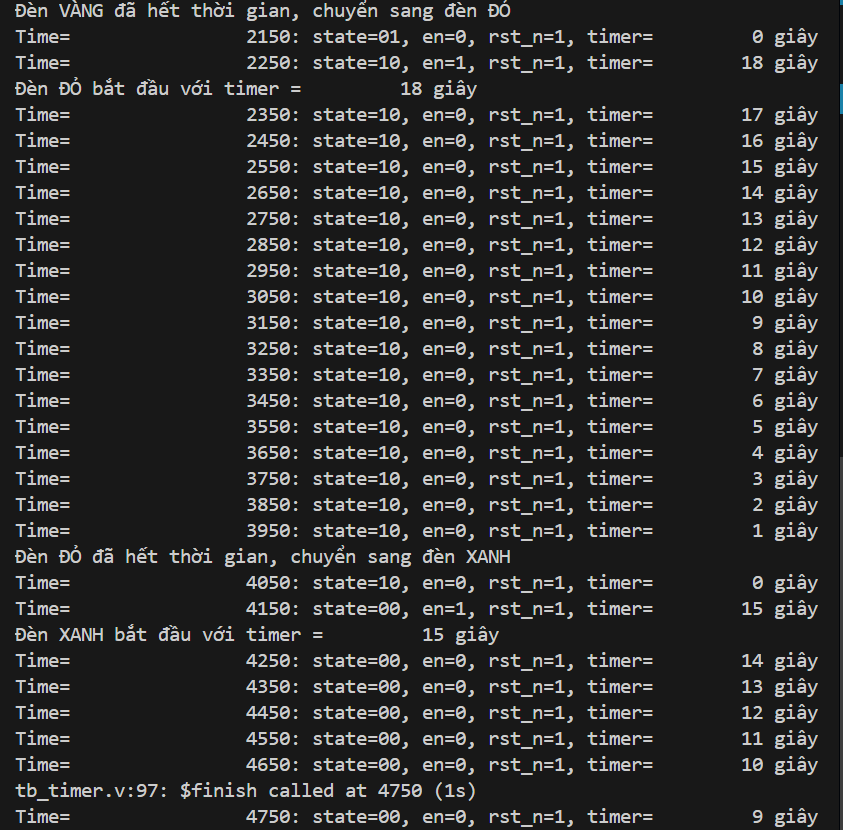


* **Khối Timer :**

chức năng chính là đếm thời gian gian đếm ngược của từng trạng thái và chuyển đổi theo thứ tự green 15s, yellow 3s, red 18s.

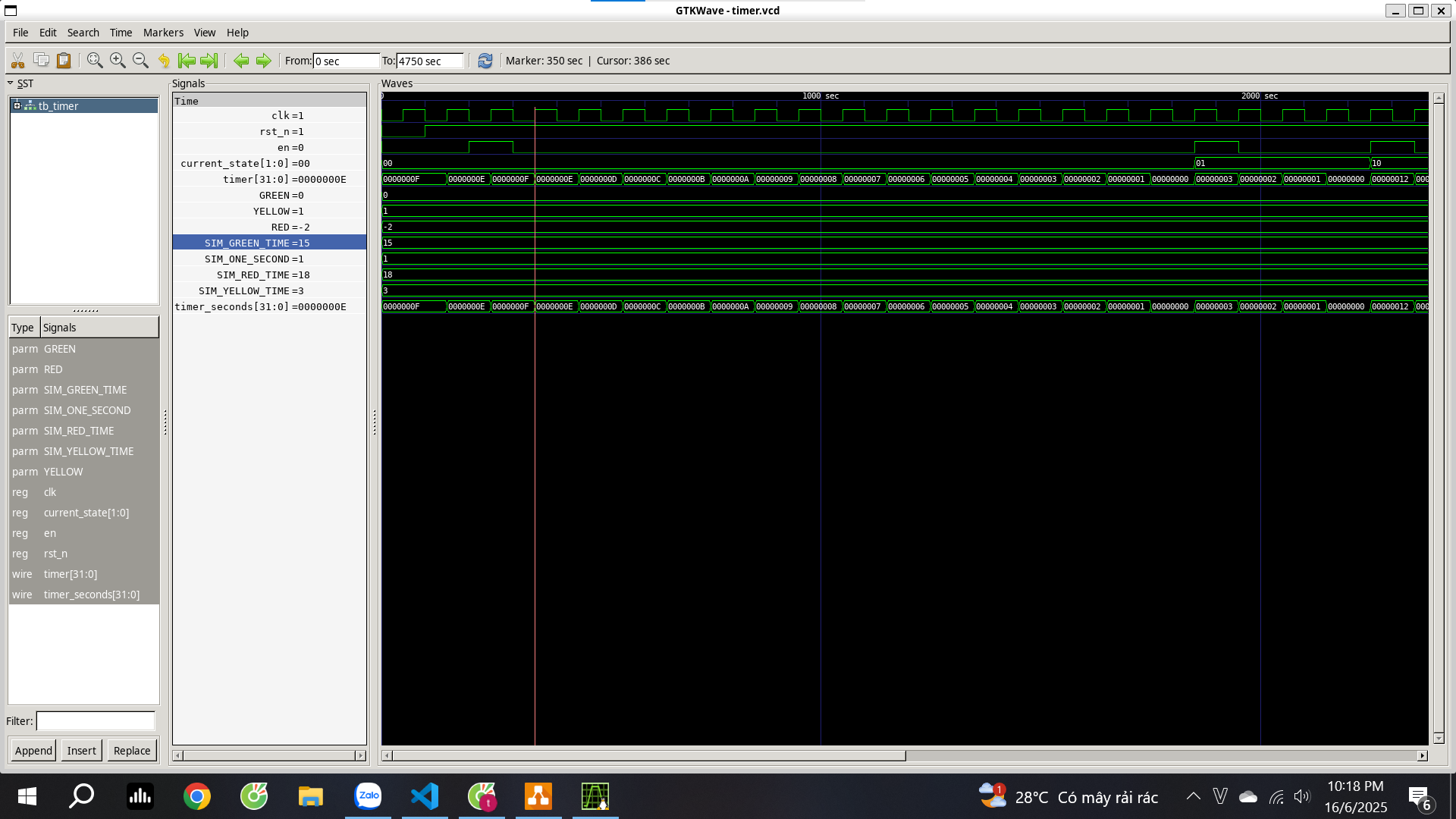
kết quả mô phỏng :





từ kết quả cho thấy module timer dã thực hiện đúng chức năng đếm ngược thời gian theo từng trạng thái và chuyển đổi sang trạng thái tiếp theo đúng với logic của bài toán.

mô phỏng trên gtkwave :



Timer.v :

module timer(

input clk,

input rst\_n,

input load,

input [1:0] current\_state,

output reg [31:0] timer,

output reg zero

);

// Định nghĩa các trạng thái

localparam GREEN = 2'b00;

localparam YELLOW = 2'b01;

localparam RED = 2'b10;

// Tham số thời gian

parameter ONE\_SECOND = 1; // clock cycle = 1 giây (đơn giản hóa mô phỏng)

parameter GREEN\_TIME = 15;

parameter YELLOW\_TIME = 3;

parameter RED\_TIME = 18;

// Giá trị nạp vào timer

reg [31:0] load\_value;

// Xác định giá trị nạp

always @(\*) begin

case (current\_state)

GREEN: load\_value = GREEN\_TIME \* ONE\_SECOND;

YELLOW: load\_value = YELLOW\_TIME \* ONE\_SECOND;

RED: load\_value = RED\_TIME \* ONE\_SECOND;

default: load\_value = GREEN\_TIME \* ONE\_SECOND;

endcase

end // Logic đếm ngược

always @(posedge clk or negedge rst\_n) begin

if (!rst\_n) begin

timer <= GREEN\_TIME \* ONE\_SECOND; // Khởi tạo với GREEN\_TIME khi reset

end else if (load) begin

timer <= load\_value; // Nạp giá trị khi load = 1

end else if (timer > 0) begin

timer <= timer - 1;

end

end

// Cờ zero

always @(posedge clk or negedge rst\_n) begin

if (!rst\_n)

zero <= 1'b0;

else

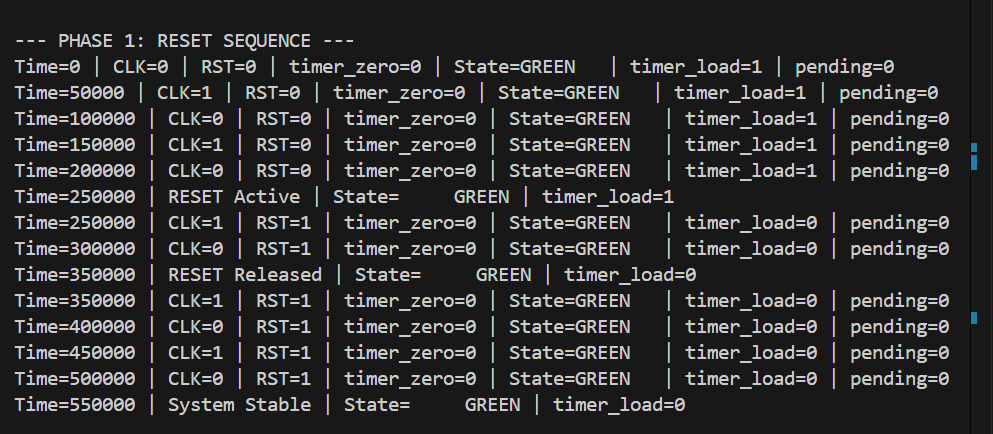
zero <= (timer == 0);

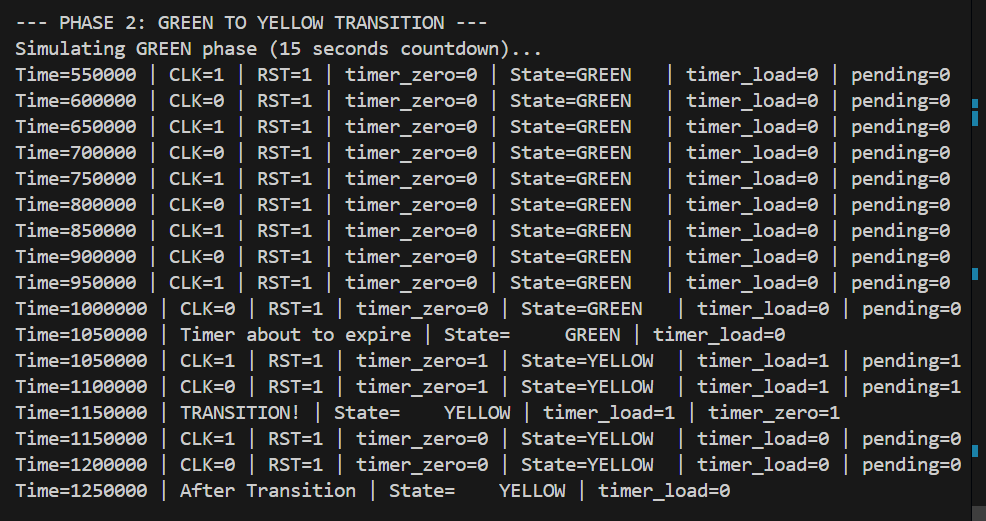
end

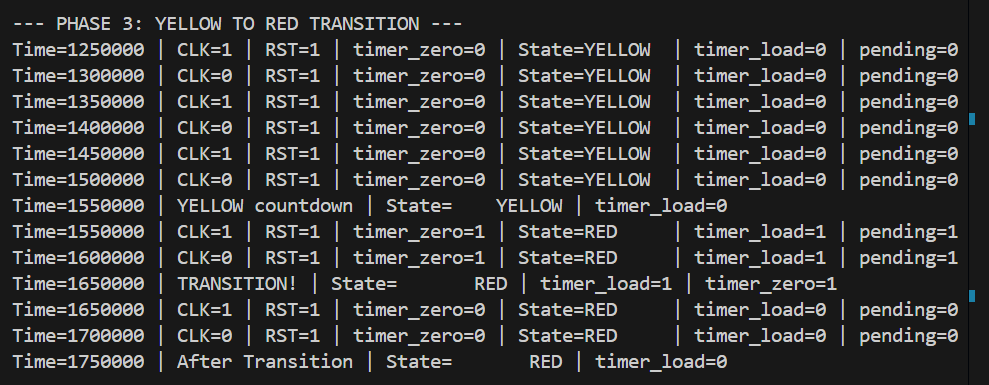
endmodule

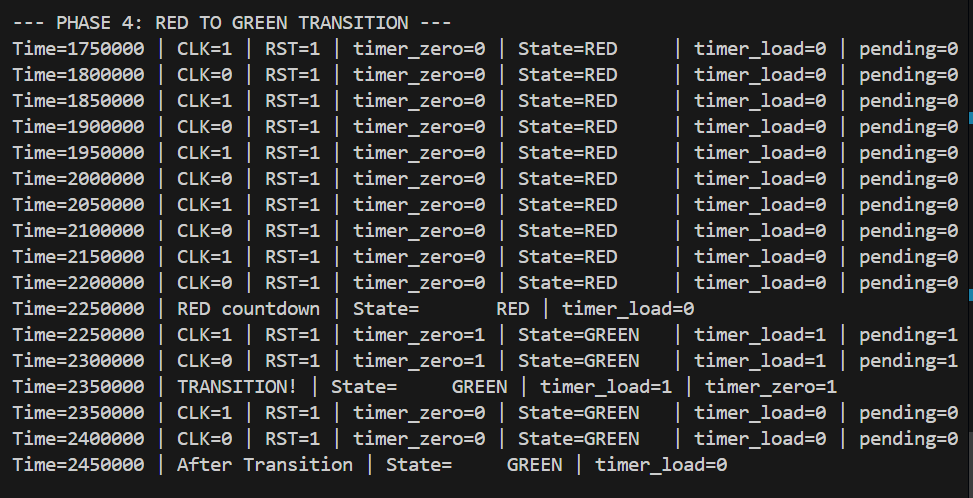
* **Khối FSM**: đây là khối với chức năng để chuyển đổi các trạng thái từ green sang yellow và yellow sang red , red sang green dựa vào việc xác định timer của state hiện tại đã đếm đến gần 0 chưa bằng biến load\_zero và timer\_load để nạp giá trị mới cho state tiếp theo..

kết quả mô phỏng :



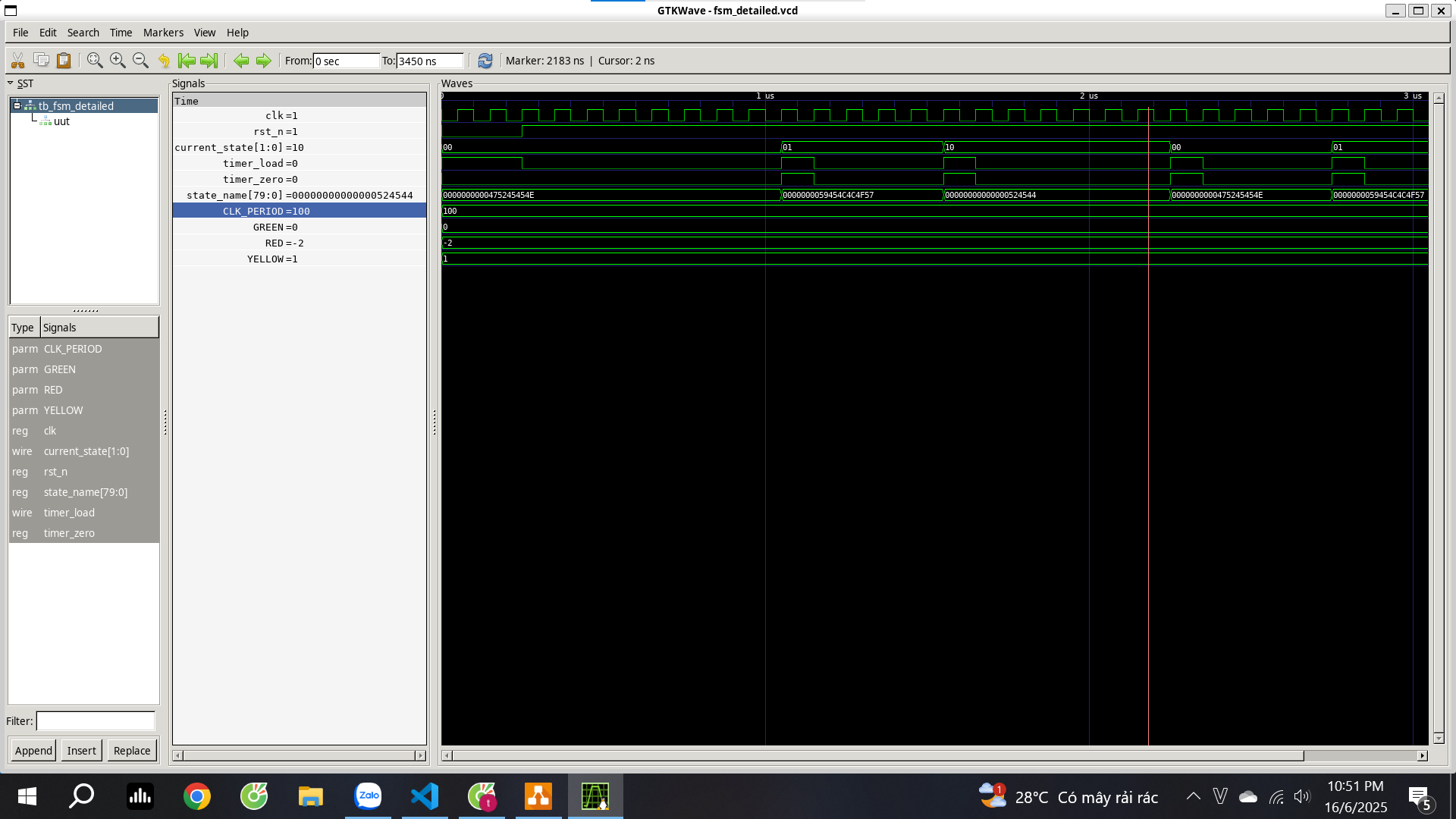






việc chuyển đổi trạng thái dựa vào timer\_load và timer\_zeo đã được thực hiện một cách chính xác khi timer\_zero = 1 thì báo hiệu timer sắp về 0 chuyển đổi sang state tiếp theo và timer\_load = 1 để nạp giá trị đếm mới vào.

mô phỏng trên gtkwave :



Traffic\_light\_fsm.v :

module traffic\_light\_fsm(

input clk,

input rst\_n,

input timer\_zero, // Tín hiệu báo timer đã đếm về 0

output reg [1:0] current\_state, // Trạng thái hiện tại

output reg timer\_load // Tín hiệu để load giá trị mới vào timer

);

// Định nghĩa các trạng thái

localparam GREEN = 2'b00;

localparam YELLOW = 2'b01;

localparam RED = 2'b10;

reg [1:0] next\_state;

reg pending\_state\_change; // Cờ delay chuyển trạng thái

// Xác định trạng thái tiếp theo

always @(\*) begin

case (current\_state)

GREEN: next\_state = YELLOW;

YELLOW: next\_state = RED;

RED: next\_state = GREEN;

default: next\_state = GREEN;

endcase

end // Luật chuyển trạng thái và phát xung nạp timer

always @(posedge clk or negedge rst\_n) begin

if (!rst\_n) begin

current\_state <= GREEN;

timer\_load <= 1'b1;

pending\_state\_change <= 1'b0;

end else begin

if (timer\_zero && !pending\_state\_change) begin

// Khi timer == 1, chuẩn bị chuyển trạng thái

current\_state <= next\_state;

timer\_load <= 1'b1;

pending\_state\_change <= 1'b1;

end else if (pending\_state\_change) begin

// Clock tiếp theo: tắt load signal

timer\_load <= 1'b0;

pending\_state\_change <= 1'b0;

end else begin

timer\_load <= 1'b0;

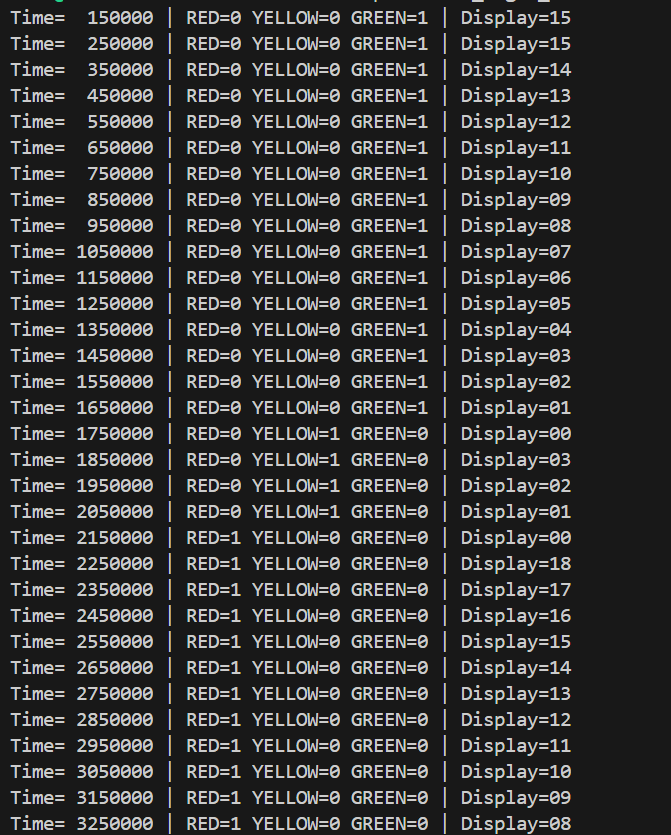
end

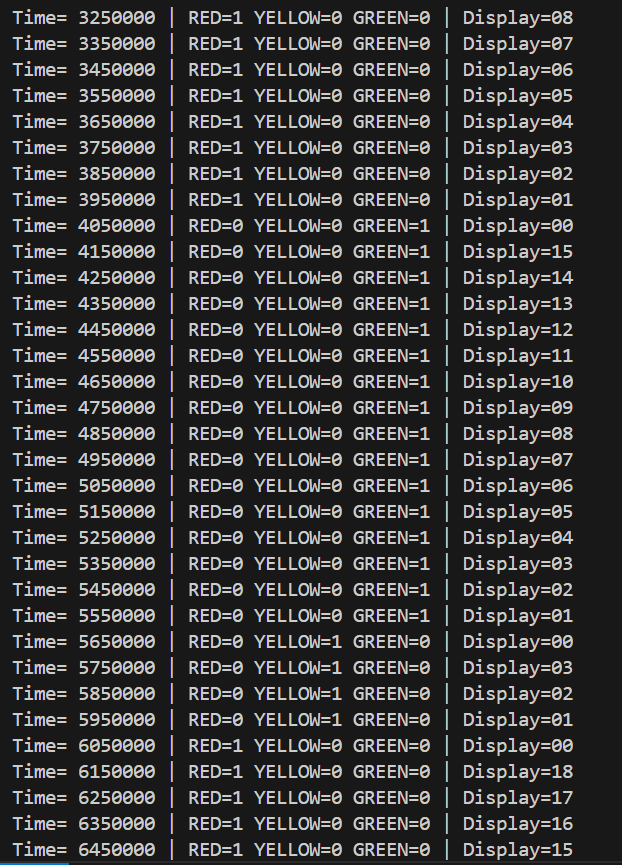
end

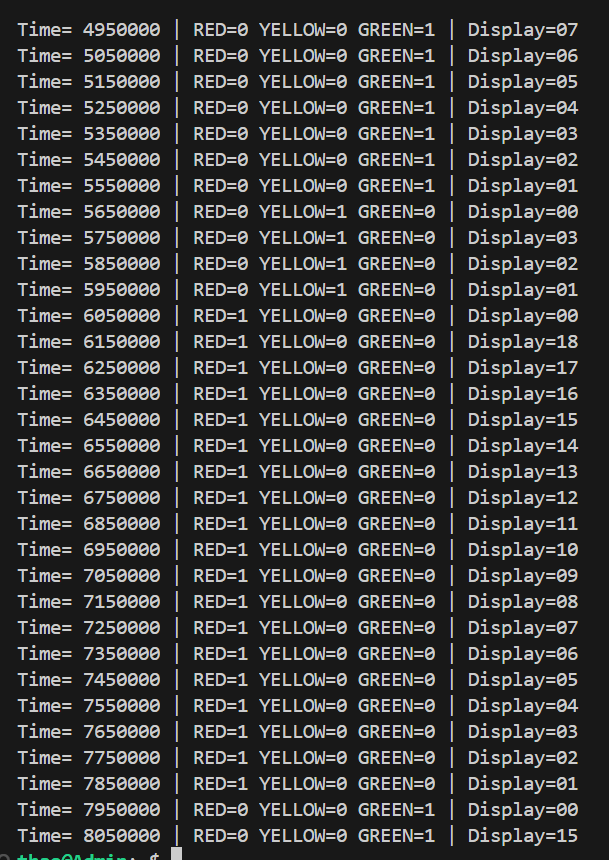
end

endmodule

* **Khối top\_module :**

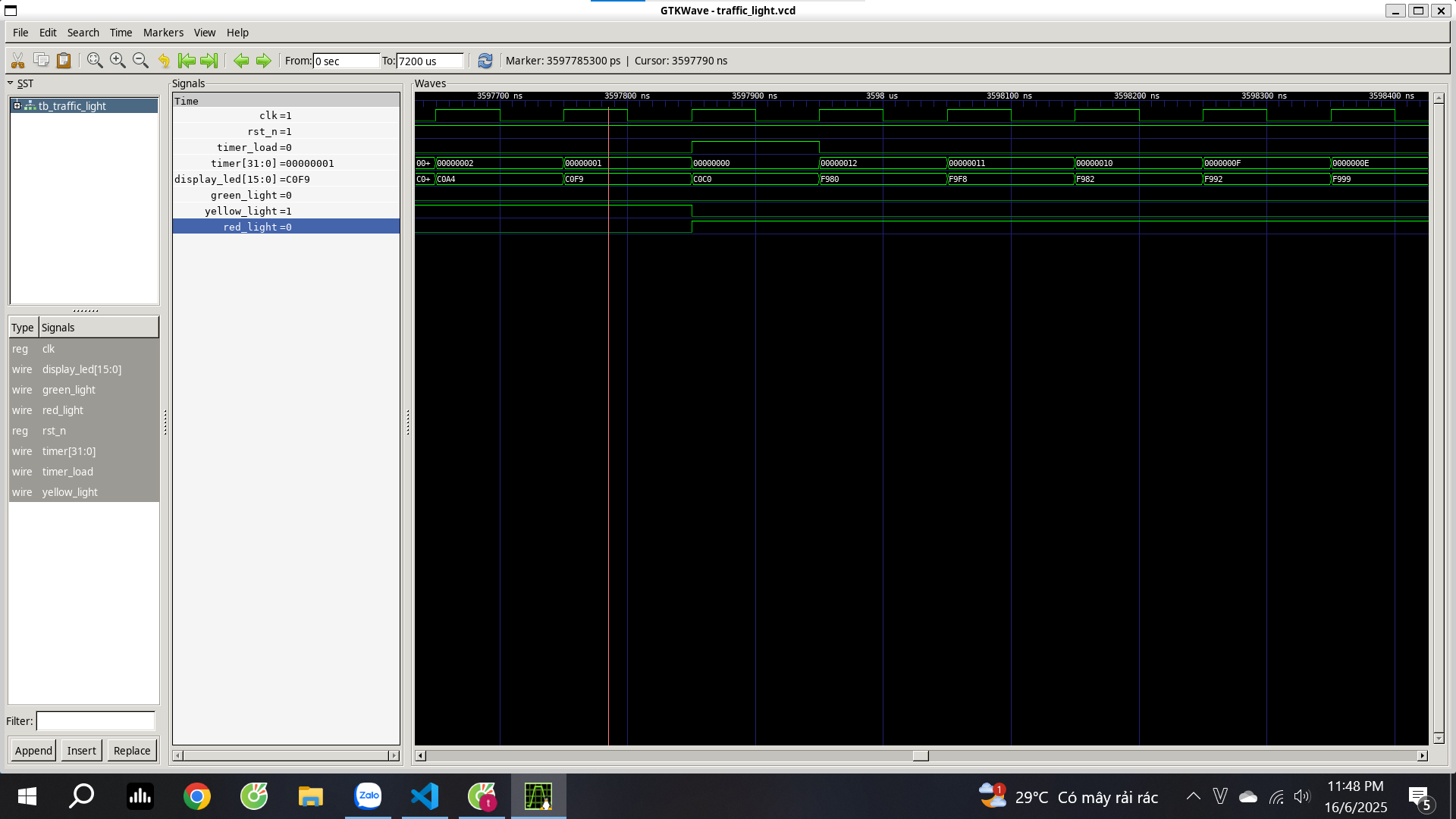






kết quả mô phỏng của file top\_module đã hoàn toán đúng logic của từng trạng thái, đếm ngược đúng thời gian quy định, chuyển đổi trạng thái đúng khi hết đếm ngược, hiển thị đúng thời gian đếm ngược của từng state trên led 7 đoạn.

mô phỏng bằng gtkwave :



Traffic\_light.v :

module traffic\_light(

input clk, // 10MHz clock

input rst\_n, // Active low asynchronous reset

output reg red\_light, // Enable red light

output reg yellow\_light, // Enable yellow light

output reg green\_light, // Enable green light

output [15:0] display\_led, // 7 segment display, 2 digits

output timer\_load, // Timer load signal

output [31:0] timer // Timer value

);

// Định nghĩa các trạng thái

localparam GREEN = 2'b00;

localparam YELLOW = 2'b01;

localparam RED = 2'b10;

// Khai báo các tín hiệu nội bộ

wire [1:0] current\_state; // Trạng thái hiện tại từ FSM

wire timer\_zero; // Tín hiệu báo timer đã đếm về 0 // Tín hiệu báo timer đã đếm về 0 (sử dụng timer == 1 để detect trước 1 clock)

assign timer\_zero = (timer == 1);

// Khởi tạo module FSM

traffic\_light\_fsm fsm\_inst(

.clk(clk),

.rst\_n(rst\_n),

.timer\_zero(timer\_zero),

.current\_state(current\_state),

.timer\_load(timer\_load)

);

// Khởi tạo module timer

timer timer\_inst(

.clk(clk),

.rst\_n(rst\_n),

.load(timer\_load),

.current\_state(current\_state),

.timer(timer)

);

// Module hiển thị LED 7 đoạn - sử dụng combinational logic

seg7\_display display\_inst(

.clk(clk),

.rst\_n(rst\_n),

.value(timer[7:0] > 0 ? timer[7:0] : 8'd00), // Hiển thị timer hiện tại, min = 00

.display(display\_led)

);

// Logic điều khiển đèn

always @(\*) begin

// Mặc định tắt tất cả đèn

green\_light = 1'b0;

yellow\_light = 1'b0;

red\_light = 1'b0;

case (current\_state)

GREEN: begin

green\_light = 1'b1;

end

YELLOW: begin

yellow\_light = 1'b1;

end

RED: begin

red\_light = 1'b1;

end

default: begin

// Giữ tất cả đèn tắt trong trường hợp không xác định

end

endcase

end

// Add monitoring for timer, value, and display signals

initial begin

$monitor("Time=%t | Timer=%d | Value=%d | Display=%h", $time, timer, display\_inst.value, display\_led);

end

endmodule

// Module hiển thị LED 7 đoạn

module seg7\_display(

input clk,

input rst\_n,

input [7:0] value, // Giá trị cần hiển thị (0-99)

output [15:0] display // 16 bit cho 2 chữ số LED 7 đoạn

);

// Chữ số hàng đơn vị và hàng chục

wire [3:0] ones, tens;

// Tách chữ số

assign ones = value % 10;

assign tens = (value / 10) % 10;

// Chuyển đổi từ số thành mã LED 7 đoạn

function [7:0] to\_seg7;

input [3:0] num;

begin

case (num)

4'h0: to\_seg7 = 8'b11000000; // 0

4'h1: to\_seg7 = 8'b11111001; // 1

4'h2: to\_seg7 = 8'b10100100; // 2

4'h3: to\_seg7 = 8'b10110000; // 3

4'h4: to\_seg7 = 8'b10011001; // 4

4'h5: to\_seg7 = 8'b10010010; // 5

4'h6: to\_seg7 = 8'b10000010; // 6

4'h7: to\_seg7 = 8'b11111000; // 7

4'h8: to\_seg7 = 8'b10000000; // 8

4'h9: to\_seg7 = 8'b10010000; // 9

default: to\_seg7 = 8'b11111111; // Tắt

endcase

end

endfunction

// Cập nhật hiển thị - sử dụng combinational logic để không có delay

assign display = (!rst\_n) ? 16'hFFFF : {to\_seg7(tens), to\_seg7(ones)};

endmodule

**PHÂN TÍCH TÍN HIỆU**

* **Timing critical points:**

- Clock N-1: timer=2, hoạt động bình thường

- Clock N: timer=1, timer\_zero=1, FSM phát hiện

- Clock N+1: Thay đổi trạng thái, timer\_load=1

- Clock N+2: Nạp giá trị timer mới, timer\_load=0

* **Tính năng an toàn:**

- Reset fail-safe: Chuyển ngay lập tức về trạng thái GREEN + nạp timer

- State Protection: Cờ pending\_state\_change ngăn chặn chuyển đổi trạng thái liên tục

- Timer Protection: Không thể đếm ngược về giá trị âm

- Display Protection: Logic tổ hợp, cập nhật tức thời

- Timing Protection: timer\_zero active trong đúng 1 chu kỳ clock