HW2 電路設計說明

第14組

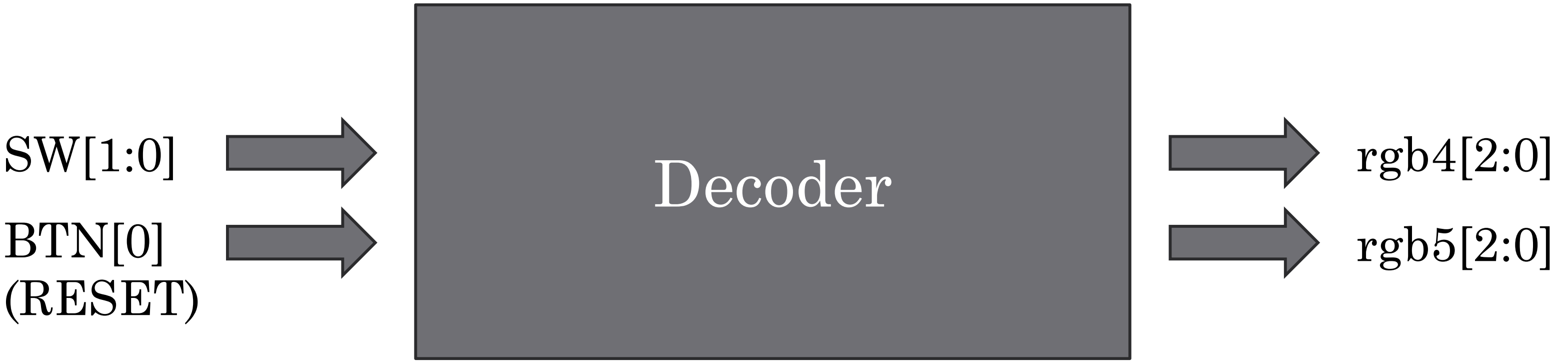
黃偉峰 E34106010

陳識博 E94106096

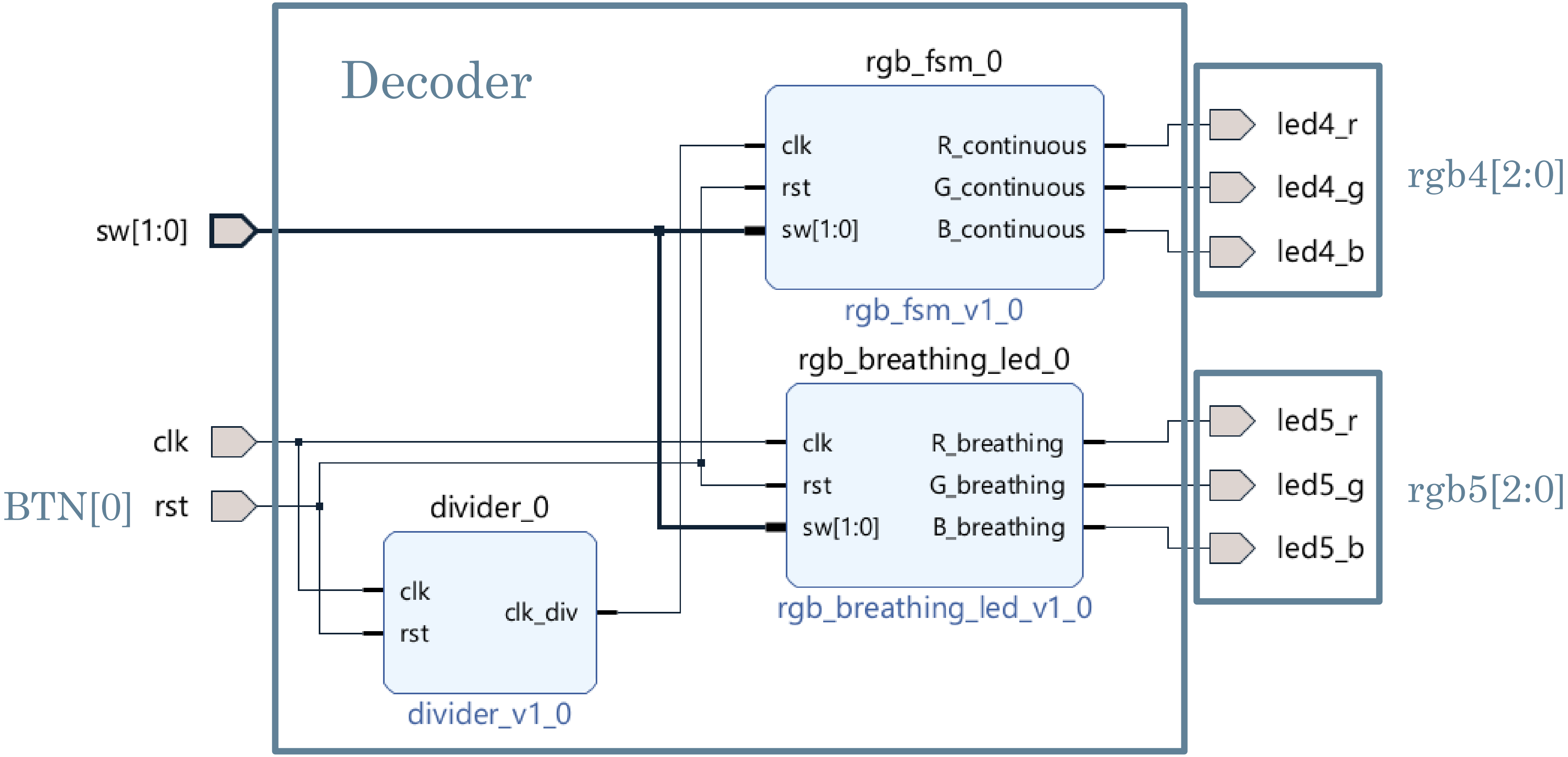
黃芊 F74104040

Problem 1 – Breathing Light

Block Diagram

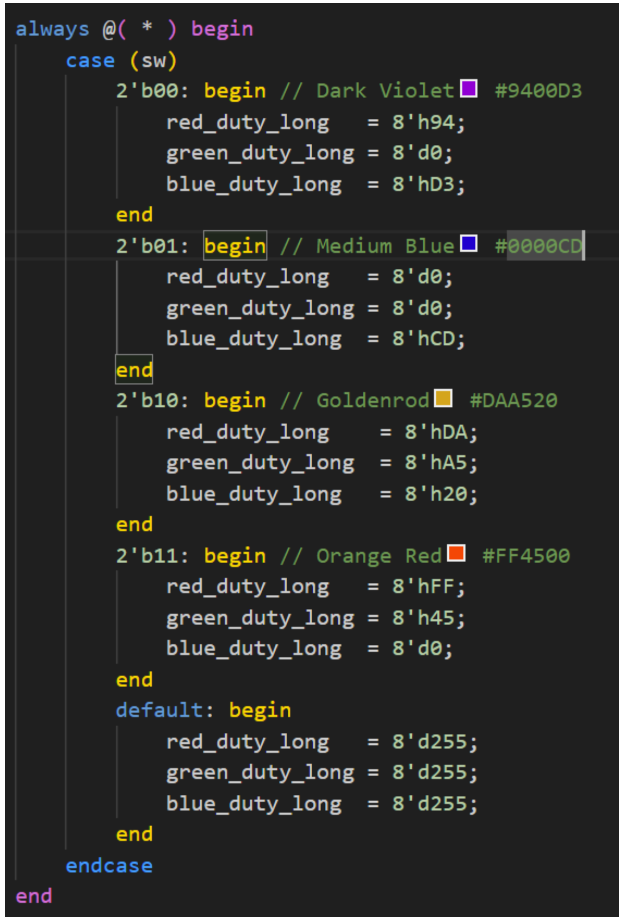


Block Diagram (Our Design)

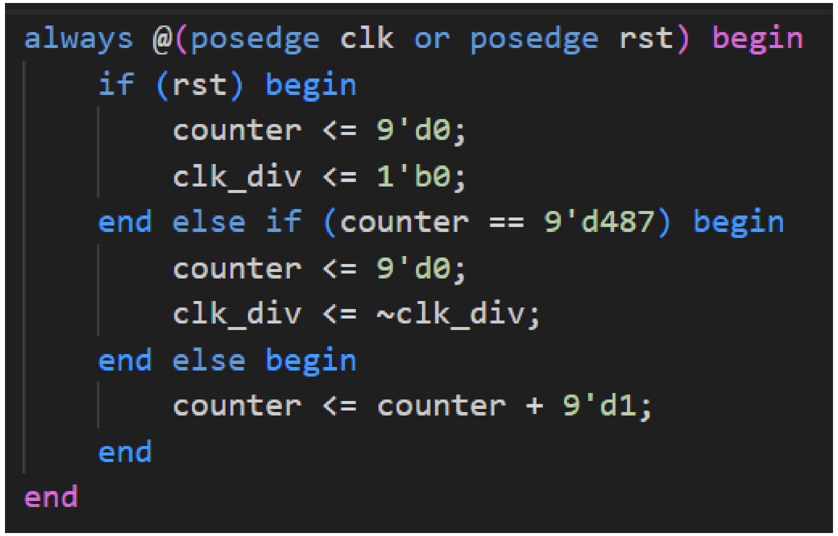


rgb_fsm 模組 - 恆亮

顏色設定 & 除頻器



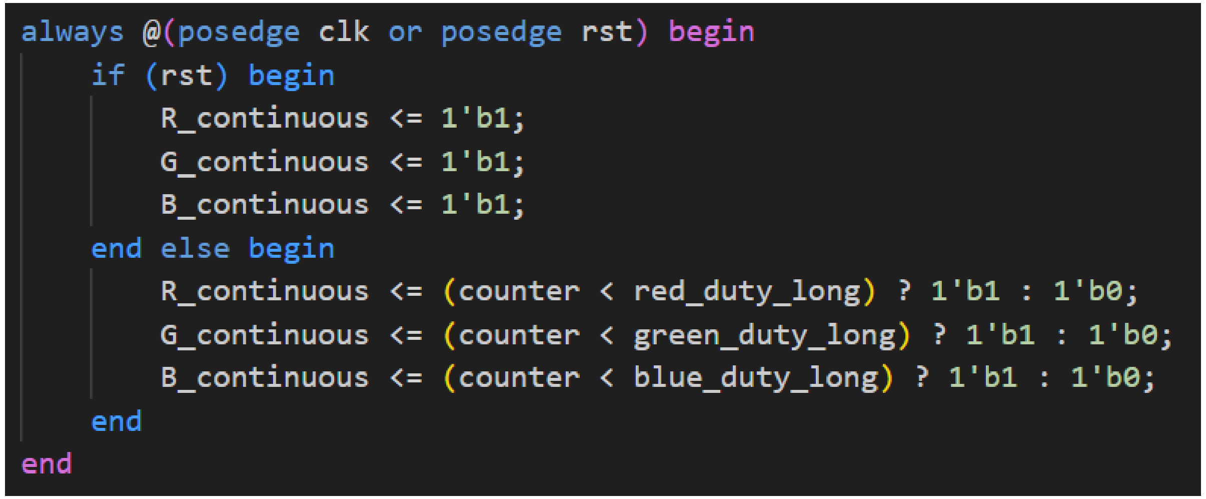
```verilog
always @( * ) begin
    case (sw)
        2'b00: begin // Dark Violet🟪 #9400D3
            red_duty_long   = 8'h94;
            green_duty_long = 8'd0;
            blue_duty_long  = 8'hD3;
        end
        2'b01: begin // Medium Blue🟦 #0000CD
            red_duty_long   = 8'd0;
            green_duty_long = 8'd0;
            blue_duty_long  = 8'hCD;
        end
        2'b10: begin // Goldenrod🟨 #DAA520
            red_duty_long     = 8'hDA;
            green_duty_long  = 8'hA5;
            blue_duty_long    = 8'h20;
        end
        2'b11: begin // Orange Red🟧 #FF4500
            red_duty_long   = 8'hFF;
            green_duty_long = 8'h45;
            blue_duty_long  = 8'd0;
        end
        default: begin
            red_duty_long   = 8'd255;
            green_duty_long = 8'd255;
            blue_duty_long  = 8'd255;
        end
    endcase
end
```



```verilog
always @(posedge clk or posedge rst) begin
    if (rst) begin
        counter <= 9'd0;
        clk_div <= 1'b0;
    end else if (counter == 9'd487) begin
        counter <= 9'd0;
        clk_div <= ~clk_div;
    end else begin
        counter <= counter + 9'd1;
    end
end
```

直接將各個顏色的RGB設成PWM duty cycle的占比，再利用除頻器將rgb_fsm模組的clk頻率降為約128kHz，讓PWM波的頻率不會太快。

PWM 輸出控制（RGB）



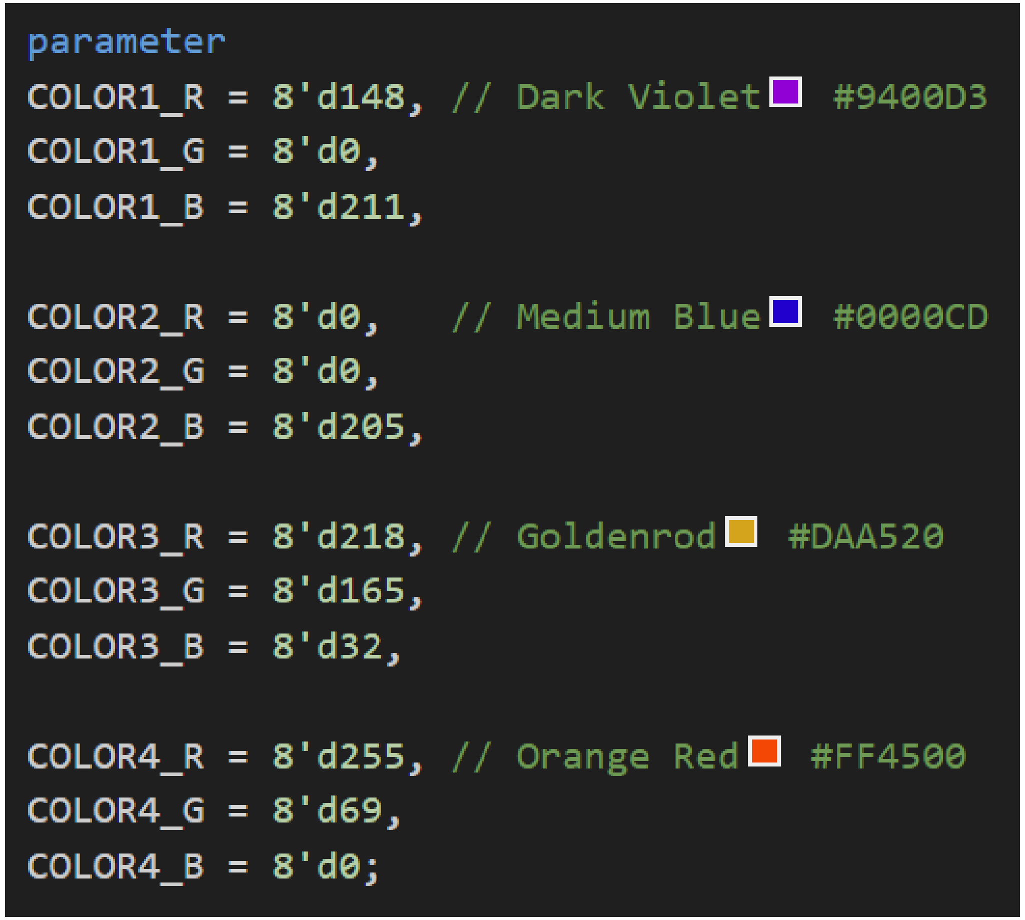
```
always @(posedge clk or posedge rst) begin
    if (rst) begin
        R_continuous <= 1'b1;
        G_continuous <= 1'b1;
        B_continuous <= 1'b1;
    end else begin
        R_continuous <= (counter < red_duty_long) ? 1'b1 : 1'b0;
        G_continuous <= (counter < green_duty_long) ? 1'b1 : 1'b0;
        B_continuous <= (counter < blue_duty_long) ? 1'b1 : 1'b0;
    end
end
```

每個 channel 都會比較目前的 counter 和目標亮度值 (*_duty_long)：

- 若 counter < duty_value，就輸出高電位（開燈）
- 否則就輸出低電位（關燈）
- 這樣便利用每個channel佔空比的不同，創造出不同的顏色。

rgb_breathing_led 模組 - 呼吸

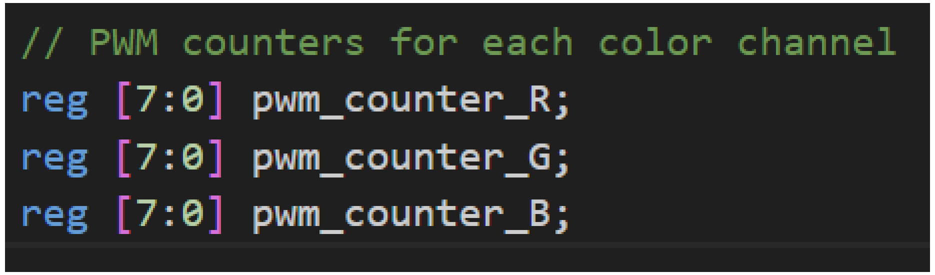
顏色參數定義



```
parameter
COLOR1_R = 8'd148, // Dark Violet🟪  #9400D3
COLOR1_G = 8'd0,
COLOR1_B = 8'd211,

COLOR2_R = 8'd0,   // Medium Blue🟦  #0000CD
COLOR2_G = 8'd0,
COLOR2_B = 8'd205,

COLOR3_R = 8'd218, // Goldenrod🟨  #DAA520
COLOR3_G = 8'd165,
COLOR3_B = 8'd32,

COLOR4_R = 8'd255, // Orange Red🟧  #FF4500
COLOR4_G = 8'd69,
COLOR4_B = 8'd0;
```

選用spec中其中一組顏色，並定義為4種顏色參數，每一個channel（R/G/B）使用 8-bit表示亮度（0~255）。

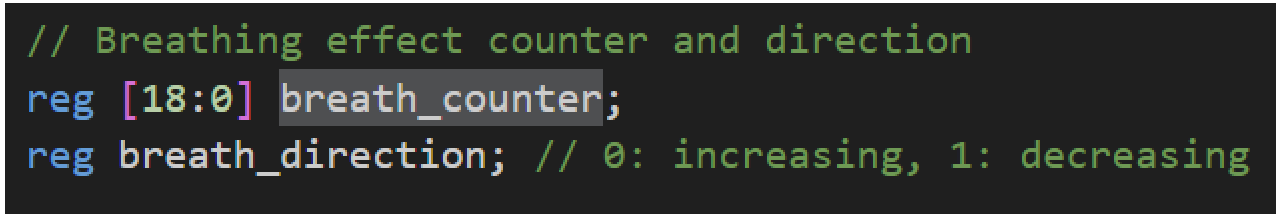
PWM 計數器（每個顏色一個）



```
// PWM counters for each color channel
reg [7:0] pwm_counter_R;
reg [7:0] pwm_counter_G;
reg [7:0] pwm_counter_B;
```

- 各自用來產生 PWM 波，控制亮度。
- 每一個計數器週期從 0 到 255，與 scaled_breath_X 比較決定輸出高低。

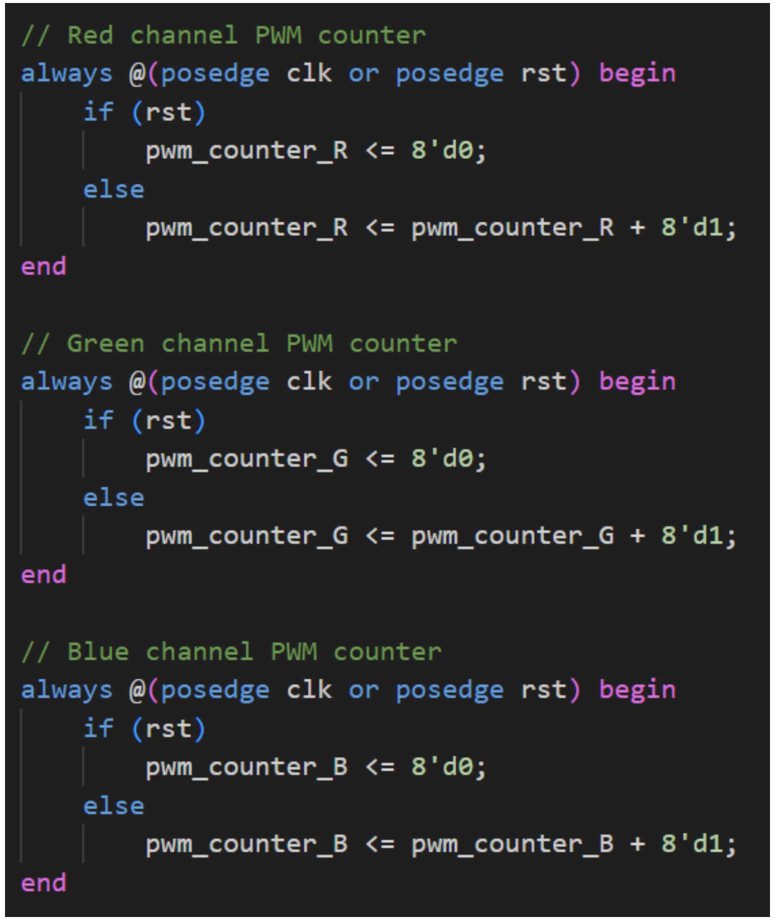
呼吸效果的計數器與方向控制



```
// Breathing effect counter and direction
reg [18:0] breath_counter;
reg breath_direction; // 0: increasing, 1: decreasing
```

- breath_counter控制「呼吸亮度」的變化範圍（19-bit，從 0 ~ 0x7FFFF）
- breath_direction 控制呼吸亮度是遞增還是遞減

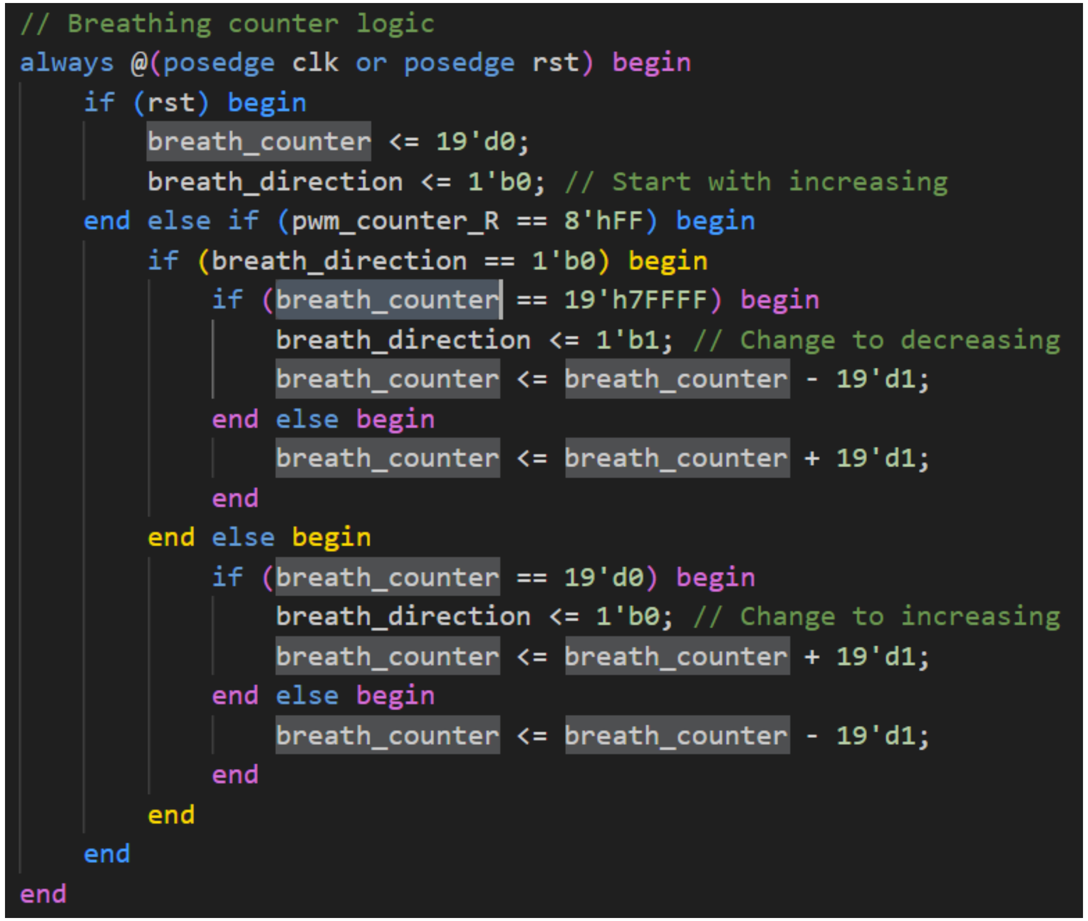
PWM 計數器運作（3組-RGB）



```verilog
// Red channel PWM counter
always @(posedge clk or posedge rst) begin
    if (rst)
        pwm_counter_R <= 8'd0;
    else
        pwm_counter_R <= pwm_counter_R + 8'd1;
end

// Green channel PWM counter
always @(posedge clk or posedge rst) begin
    if (rst)
        pwm_counter_G <= 8'd0;
    else
        pwm_counter_G <= pwm_counter_G + 8'd1;
end

// Blue channel PWM counter
always @(posedge clk or posedge rst) begin
    if (rst)
        pwm_counter_B <= 8'd0;
    else
        pwm_counter_B <= pwm_counter_B + 8'd1;
end
```

- 每次時脈上升沿，3個計數器都會加一（從 0 到 255 循環）
- 每次週期完成時 (==255)， ，就更新呼吸亮度用的 breath_counter

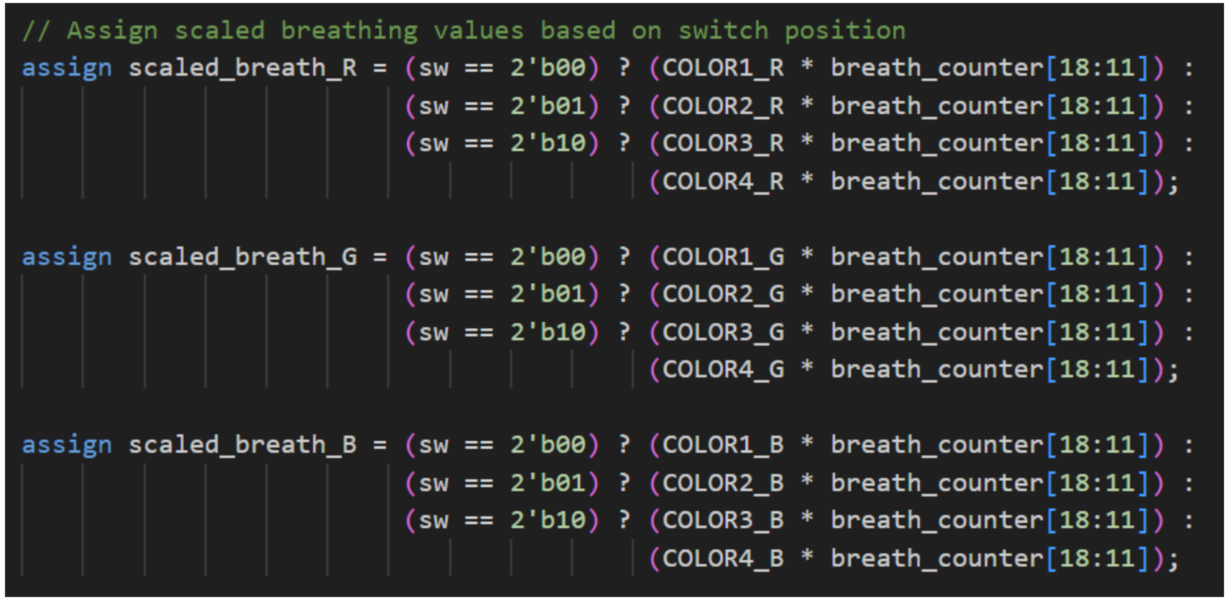
呼吸計數器邏輯



```
// Breathing counter logic
always @(posedge clk or posedge rst) begin
    if (rst) begin
        breath_counter <= 19'd0;
        breath_direction <= 1'b0; // Start with increasing
    end else if (pwm_counter_R == 8'hFF) begin
        if (breath_direction == 1'b0) begin
            if (breath_counter == 19'h7FFFF) begin
                breath_direction <= 1'b1; // Change to decreasing
                breath_counter <= breath_counter - 19'd1;
            end else begin
                breath_counter <= breath_counter + 19'd1;
            end
        end else begin
            if (breath_counter == 19'd0) begin
                breath_direction <= 1'b0; // Change to increasing
                breath_counter <= breath_counter + 19'd1;
            end else begin
                breath_counter <= breath_counter - 19'd1;
            end
        end
    end
end
```

- 當pwm_counter_X == 255時，更新breath_counter
- breath_counter遞增或遞減，創造呼吸效果
- 當到最大值（0x7FFFF）或最小值（0）時，改變方向

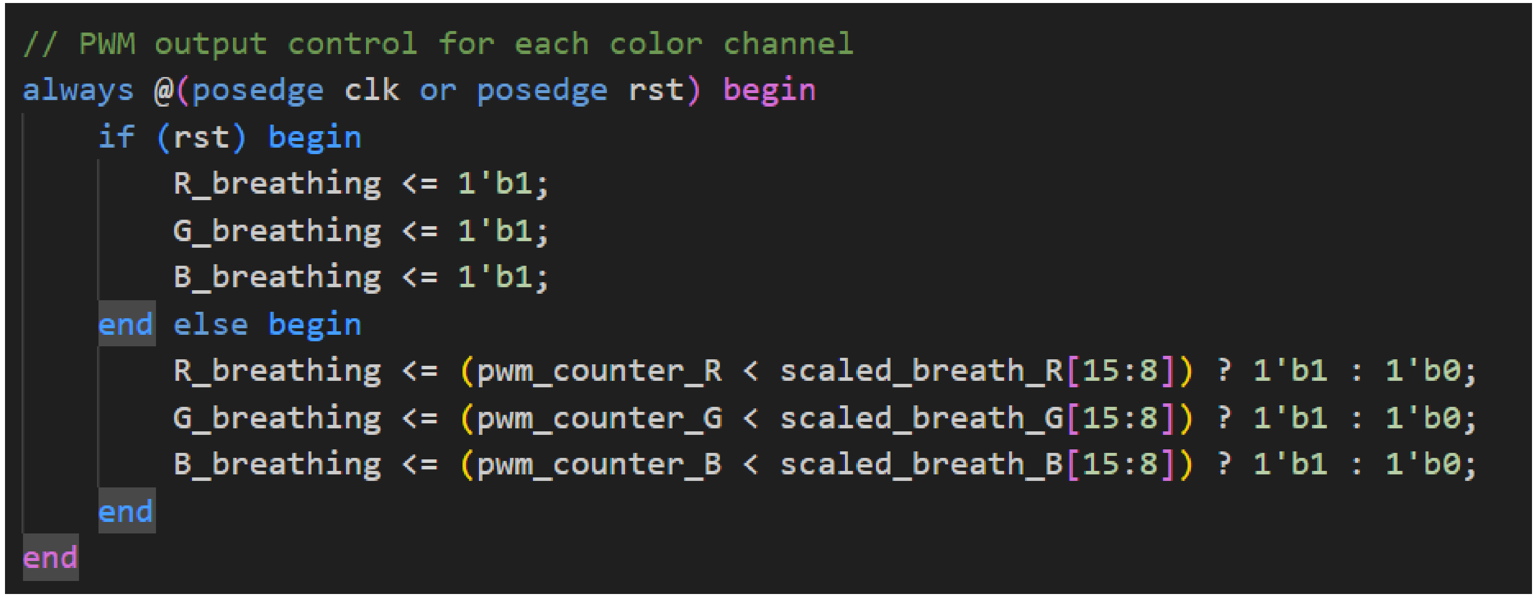
根據 sw 決定顏色模式（亮度縮放）



```
// Assign scaled breathing values based on switch position
assign scaled_breath_R = (sw == 2'b00) ? (COLOR1_R * breath_counter[18:11]) :
                         (sw == 2'b01) ? (COLOR2_R * breath_counter[18:11]) :
                         (sw == 2'b10) ? (COLOR3_R * breath_counter[18:11]) :
                                         (COLOR4_R * breath_counter[18:11]);

assign scaled_breath_G = (sw == 2'b00) ? (COLOR1_G * breath_counter[18:11]) :
                         (sw == 2'b01) ? (COLOR2_G * breath_counter[18:11]) :
                         (sw == 2'b10) ? (COLOR3_G * breath_counter[18:11]) :
                                         (COLOR4_G * breath_counter[18:11]);

assign scaled_breath_B = (sw == 2'b00) ? (COLOR1_B * breath_counter[18:11]) :
                         (sw == 2'b01) ? (COLOR2_B * breath_counter[18:11]) :
                         (sw == 2'b10) ? (COLOR3_B * breath_counter[18:11]) :
                                         (COLOR4_B * breath_counter[18:11]);
```

- breath_counter[18:11]: 只取高8-bits當成縮放因子(範圍0~255)
- 乘上顏色的原始值 → 得到實際的 PWM 亮度（16-bit）
- 根據 sw 值來選擇不同顏色

根據 PWM 比較決定輸出高低（RGB）

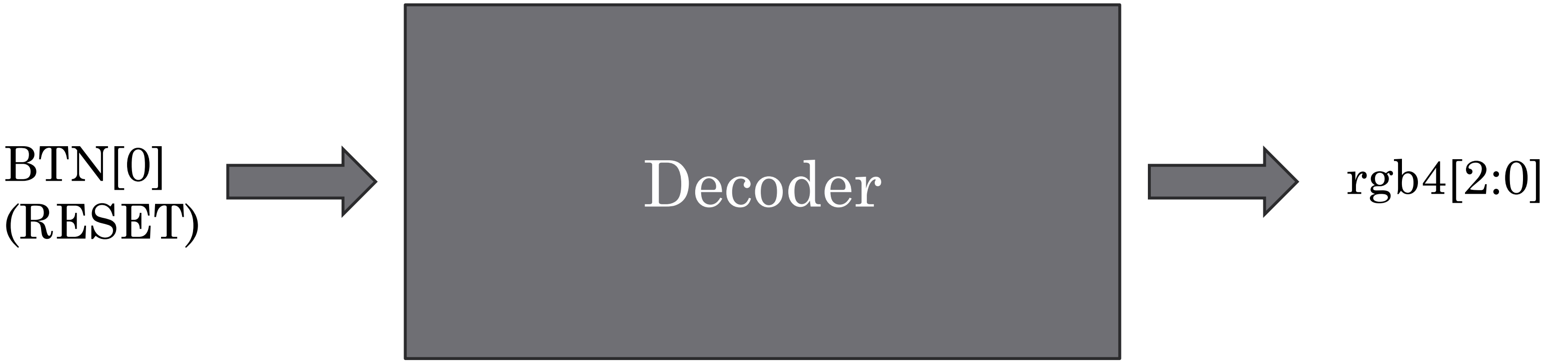


```
// PWM output control for each color channel
always @(posedge clk or posedge rst) begin
    if (rst) begin
        R_breathing <= 1'b1;
        G_breathing <= 1'b1;
        B_breathing <= 1'b1;
    end else begin
        R_breathing <= (pwm_counter_R < scaled_breath_R[15:8]) ? 1'b1 : 1'b0;
        G_breathing <= (pwm_counter_G < scaled_breath_G[15:8]) ? 1'b1 : 1'b0;
        B_breathing <= (pwm_counter_B < scaled_breath_B[15:8]) ? 1'b1 : 1'b0;
    end
end
```

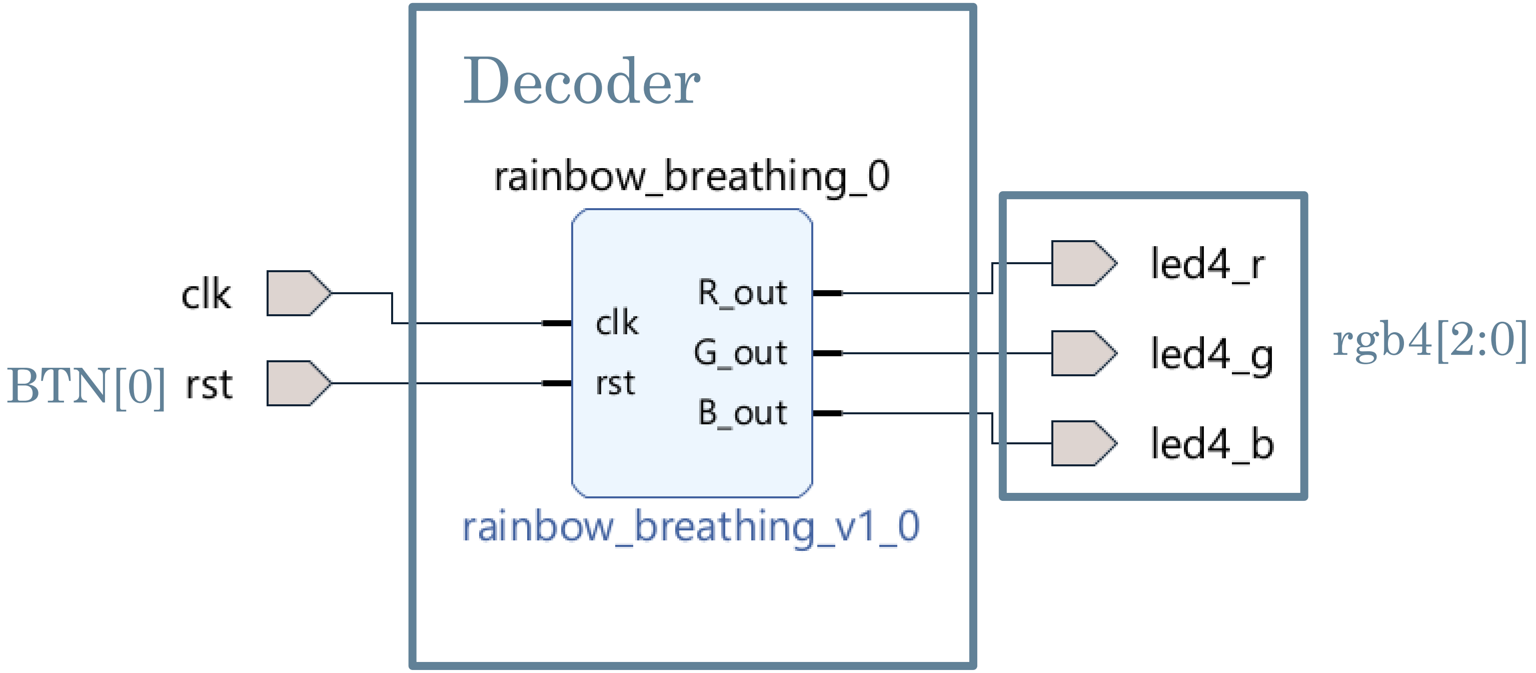
- 比較 pwm_counter_X 和 scaled_breath_X[15:8]
- 如果 counter < scaled_breath→ 輸出高電位（點亮 LED），否則輸出低電位（熄滅 LED）
- 完成呼吸亮滅效果，且每種顏色的強度依比例改變

Problem 2 – Rainbow Breathing Light

Block Diagram

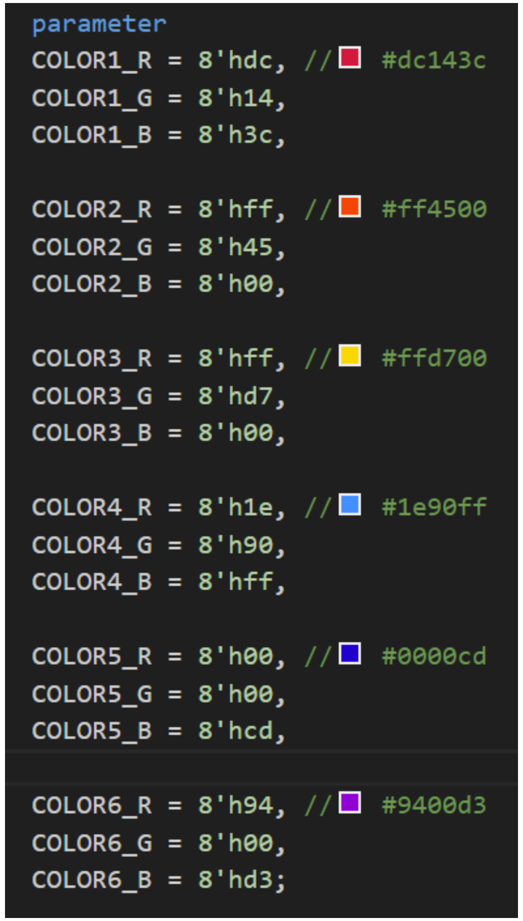


Block Diagram (Our Design)



rainbow_breathing模組

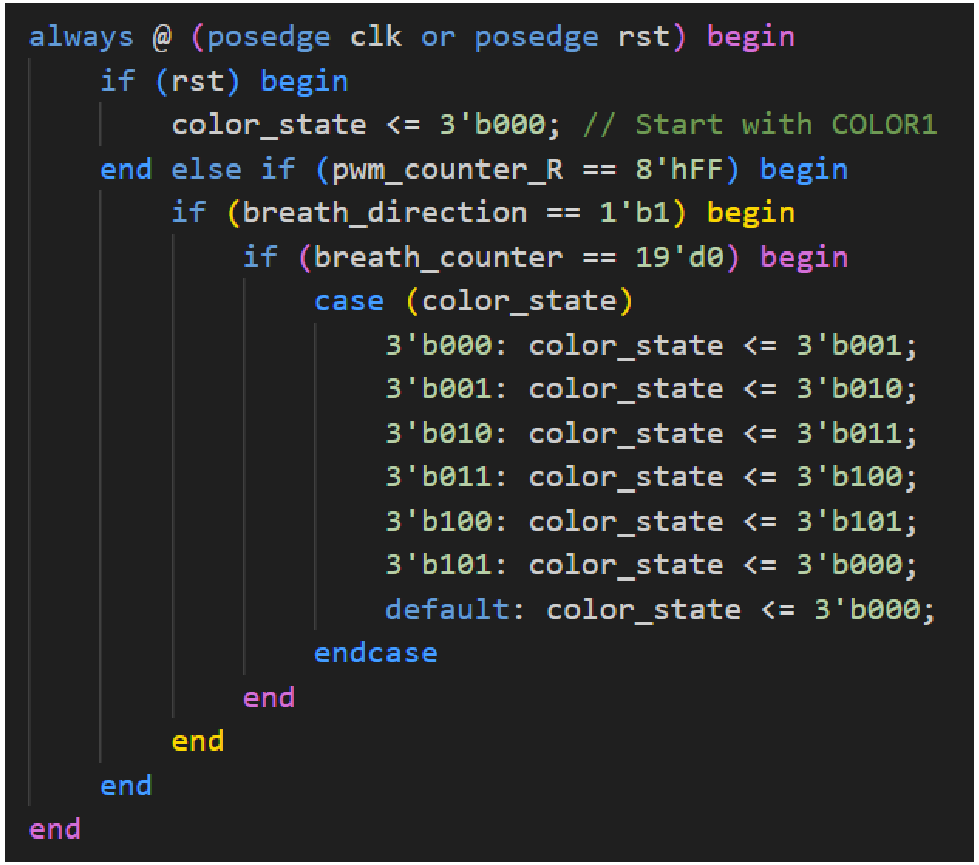
顏色設定



```
parameter
COLOR1_R = 8'hdc, //■ #dc143c
COLOR1_G = 8'h14,
COLOR1_B = 8'h3c,

COLOR2_R = 8'hff, //■ #ff4500
COLOR2_G = 8'h45,
COLOR2_B = 8'h00,

COLOR3_R = 8'hff, //■ #ffd700
COLOR3_G = 8'hd7,
COLOR3_B = 8'h00,

COLOR4_R = 8'h1e, //■ #1e90ff
COLOR4_G = 8'h90,
COLOR4_B = 8'hff,

COLOR5_R = 8'h00, //■ #0000cd
COLOR5_G = 8'h00,
COLOR5_B = 8'hcd,

COLOR6_R = 8'h94, //■ #9400d3
COLOR6_G = 8'h00,
COLOR6_B = 8'hd3;
```

依spec的要求設定6種不同顏色。

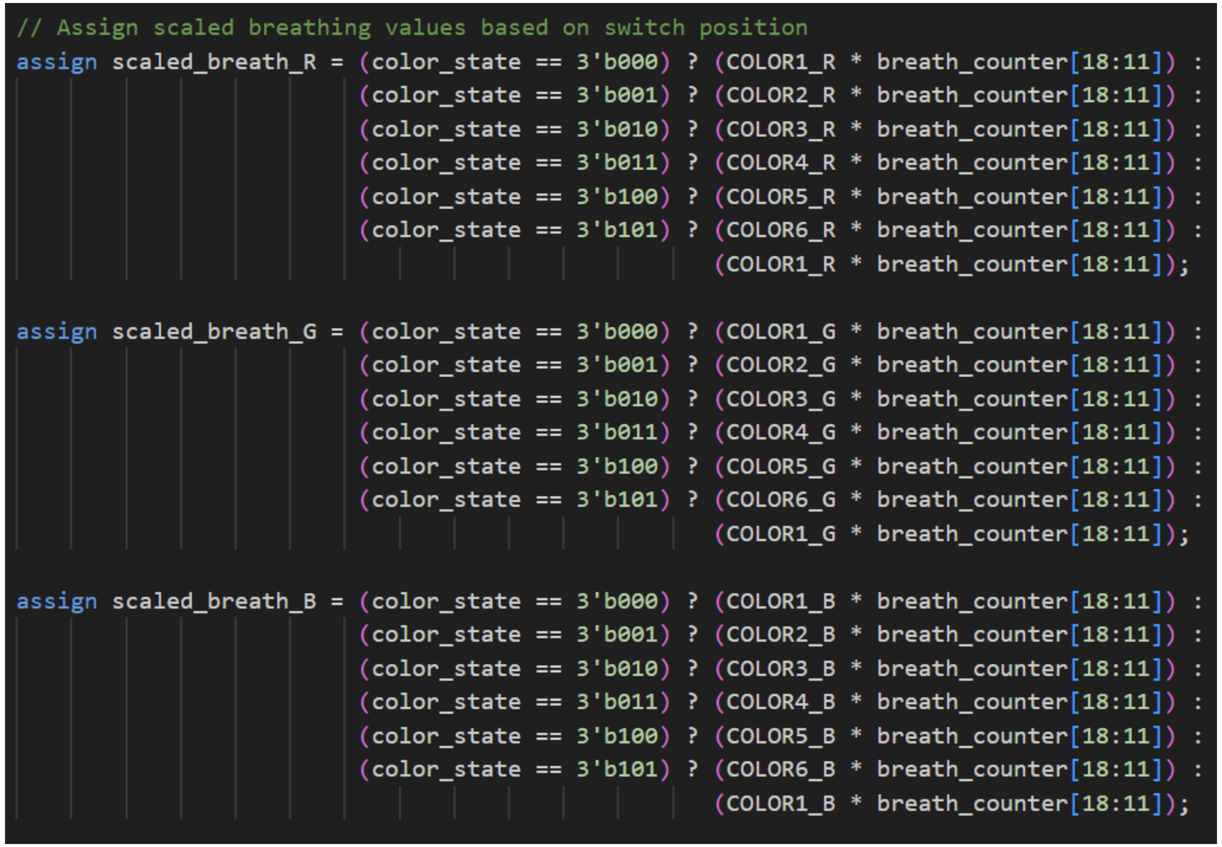
Color_state切換



```
always @ (posedge clk or posedge rst) begin
    if (rst) begin
        color_state <= 3'b000; // Start with COLOR1
    end else if (pwm_counter_R == 8'hFF) begin
        if (breath_direction == 1'b1) begin
            if (breath_counter == 19'd0) begin
                case (color_state)
                    3'b000: color_state <= 3'b001;
                    3'b001: color_state <= 3'b010;
                    3'b010: color_state <= 3'b011;
                    3'b011: color_state <= 3'b100;
                    3'b100: color_state <= 3'b101;
                    3'b101: color_state <= 3'b000;
                    default: color_state <= 3'b000;
                endcase
            end
        end
    end
end
```

照順序切換6種不同顏色，從紅色開始，最後到紫色，再回到紅色。

決定顏色模式 & 呼吸功能

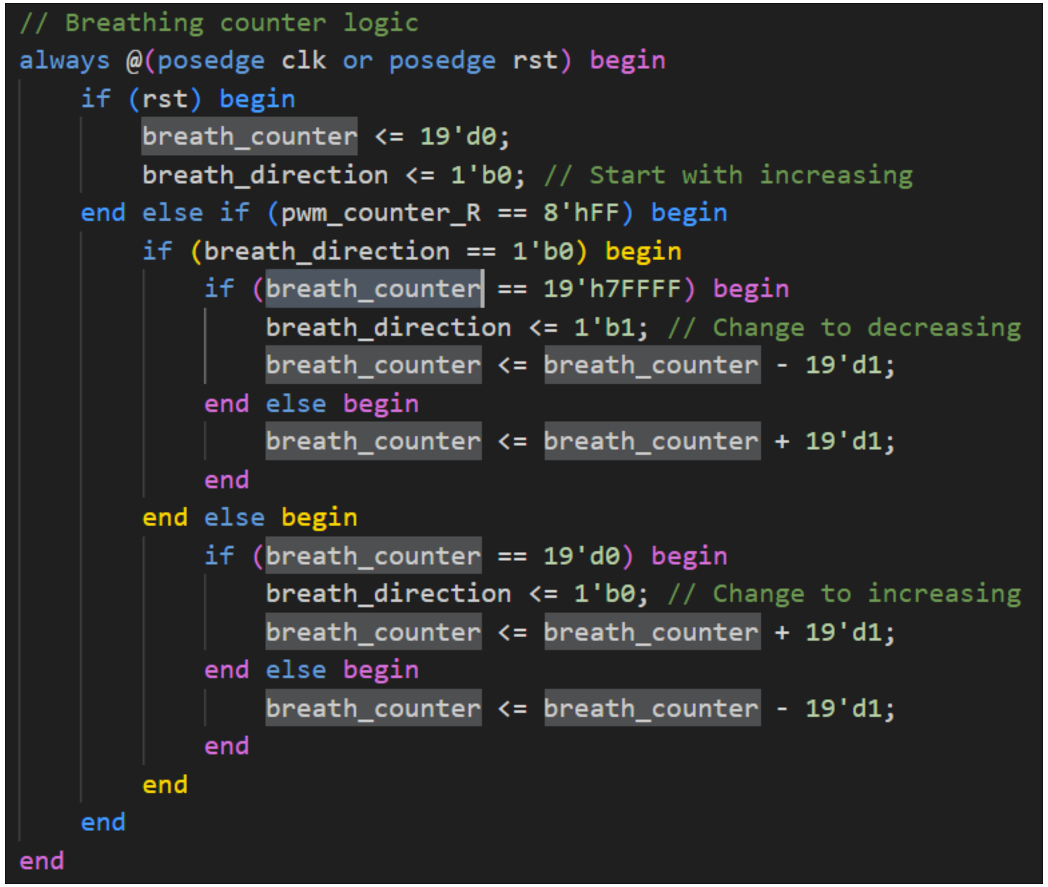


```
// Assign scaled breathing values based on switch position
assign scaled_breath_R = (color_state == 3'b000) ? (COLOR1_R * breath_counter[18:11]) :
                         (color_state == 3'b001) ? (COLOR2_R * breath_counter[18:11]) :
                         (color_state == 3'b010) ? (COLOR3_R * breath_counter[18:11]) :
                         (color_state == 3'b011) ? (COLOR4_R * breath_counter[18:11]) :
                         (color_state == 3'b100) ? (COLOR5_R * breath_counter[18:11]) :
                         (color_state == 3'b101) ? (COLOR6_R * breath_counter[18:11]) :
                                                   (COLOR1_R * breath_counter[18:11]);

assign scaled_breath_G = (color_state == 3'b000) ? (COLOR1_G * breath_counter[18:11]) :
                         (color_state == 3'b001) ? (COLOR2_G * breath_counter[18:11]) :
                         (color_state == 3'b010) ? (COLOR3_G * breath_counter[18:11]) :
                         (color_state == 3'b011) ? (COLOR4_G * breath_counter[18:11]) :
                         (color_state == 3'b100) ? (COLOR5_G * breath_counter[18:11]) :
                         (color_state == 3'b101) ? (COLOR6_G * breath_counter[18:11]) :
                                                   (COLOR1_G * breath_counter[18:11]);

assign scaled_breath_B = (color_state == 3'b000) ? (COLOR1_B * breath_counter[18:11]) :
                         (color_state == 3'b001) ? (COLOR2_B * breath_counter[18:11]) :
                         (color_state == 3'b010) ? (COLOR3_B * breath_counter[18:11]) :
                         (color_state == 3'b011) ? (COLOR4_B * breath_counter[18:11]) :
                         (color_state == 3'b100) ? (COLOR5_B * breath_counter[18:11]) :
                         (color_state == 3'b101) ? (COLOR6_B * breath_counter[18:11]) :
                                                   (COLOR1_B * breath_counter[18:11]);
```

根據不同 color state 亮 color1~6 的顏色

呼吸功能同 rgb_breathing_led 模組

呼吸燈Phase說明



```
// Breathing counter logic
always @(posedge clk or posedge rst) begin
    if (rst) begin
        breath_counter <= 19'd0;
        breath_direction <= 1'b0; // Start with increasing
    end else if (pwm_counter_R == 8'hFF) begin
        if (breath_direction == 1'b0) begin
            if (breath_counter == 19'h7FFFF) begin
                breath_direction <= 1'b1; // Change to decreasing
                breath_counter <= breath_counter - 19'd1;
            end else begin
                breath_counter <= breath_counter + 19'd1;
            end
        end else begin
            if (breath_counter == 19'd0) begin
                breath_direction <= 1'b0; // Change to increasing
                breath_counter <= breath_counter + 19'd1;
            end else begin
                breath_counter <= breath_counter - 19'd1;
            end
        end
    end
end
```

- clk 頻率 : 125MHz → 時脈週期 8 ns
- breath_counter : 19-bit（從 0 到 524287），每來回一次需 1048574 次變化 (暗→亮→暗)
- 呼吸更新條件 : 每當 pwm_counter == 255 才更新一次 (每256 個cycle)

- ➢ 變化次數（完整來回） = 2 × 524287 = 1,048,574 次
- ➢ 每次更新間隔 = 256 個 clock
- ➢ 總 clock 數 = 1,048,574 × 256 = 268,435,456 clocks
- ➢ 總時間 = 268,435,456 × 8 ns = 2,147,483,648 ns ≈ 2.15 秒

總結

- ✓ 完整呼吸週期（從暗→亮→暗）: 約 2.15 秒
- ✓ 利用 breath_counter[18:11] (8-bit) 將一次呼吸(暗→亮→暗)切成512個亮度階層。
  - Phase 1 (暗→亮) : breath_counter = 0 → 524287
  - Phase 2 (暗→亮) : breath_counter = 524287 → 0

  ⇒256 * 2 = 512
- ✓ 本次取這樣的週期剛好使暗→亮，亮→暗的周期在1.05秒，肉眼可見，不會太快也不會太慢。
- ✓ Phase 數多，亮暗切換速度慢，因為每次只讓亮度變化一點點，看起來變化會比較平滑；Phase 數少，亮暗切換速度快，如果真的太快，就會看不太出呼吸的效果。也就是說，切成16個Phase，呼吸就比較急促；切成1024個Phase，呼吸就會緩慢、柔和。

YouTube 影片 連結

- Problem 1
- Problem 2