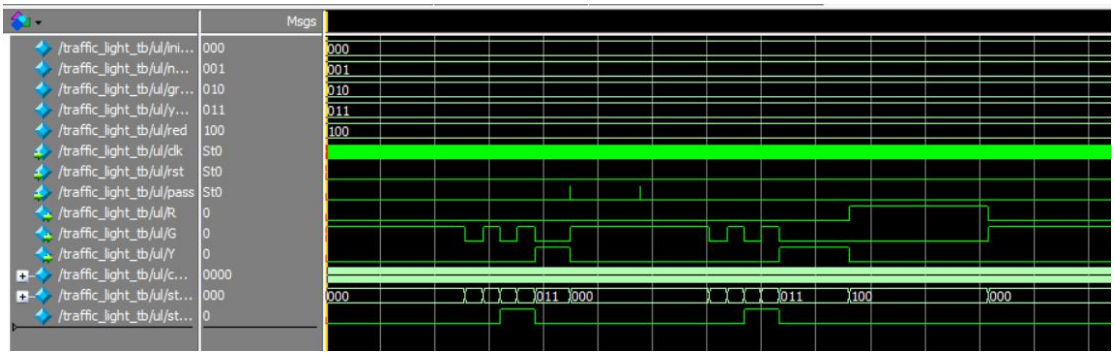
Computer Organization 2022

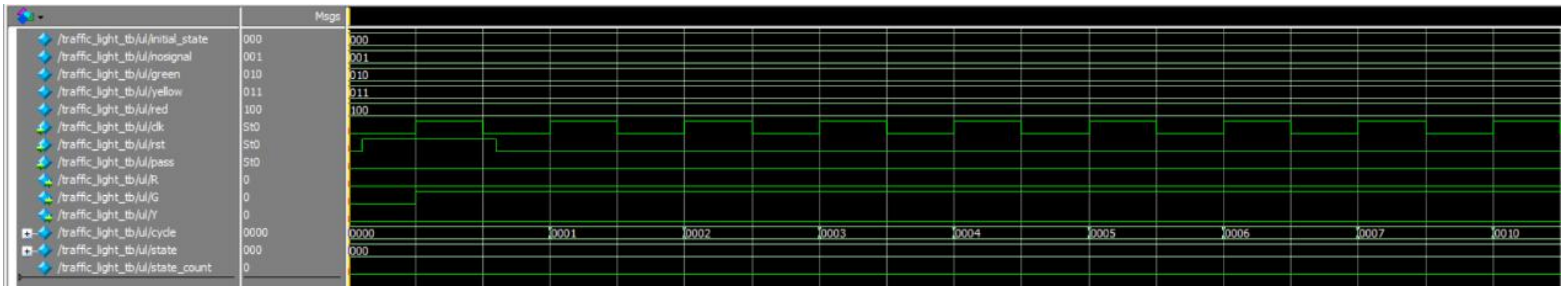
HOMEWORK 2

系級: 資訊工程系　　學號: F74109016　　姓名: 葉惟欣

實驗結果圖:

(波形圖及模擬完成截圖)






```
VSIM 7> run -all
#
#
#          ****************************
#          **                        **       /|__/|
#          **  Congratulations !!    **      / 0,0  |
#          **                        **    /_____   |
#          **  Simulation PASS!!     **   /^ ^ ^ \  |
#          **                        **  |^ ^ ^ ^ |w|
#          ************** ************   \m___m__|_|
#
#
# ** Note: $finish    : E:/altera/13.1/traffic_light_tb.v(82)
#    Time: 40961 ns  Iteration: 0  Instance: /traffic_light_tb
# 1
# Break in Module traffic_light_tb at E:/altera/13.1/traffic_light_tb.v line 82
```

程式運作流程:

(簡單說明波形變化的意義)

當 rst 為 1 的時候（也就是整個波形圖剛開始時)，clk 為正緣時 count 不加一。rst 為 0 的時候，clk 為正緣時 count 加一。clk 為正緣的時候才會判斷狀態是否改變而改變燈號。cycle 也是在正緣訊號來的時候才做加一的動作。因此 cycle 的週期是從 clk 為正緣開始到下一個 clk 的正緣。這也是為什麼起初 rst 要在起初有高電位的波型圖。Cycle 每次狀態改變都從 0 開始數。

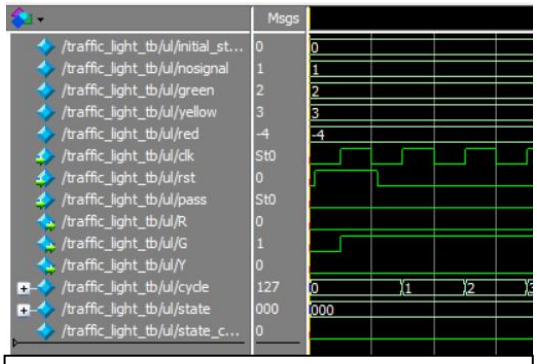
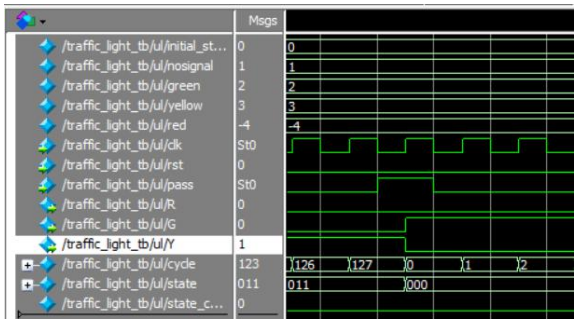


圖:一開始 rst 為一的時候的波形圖

```
always@(posedge rst)
        begin
                cycle <= 10'd0;
                state <= initial_state;
                state_count <= 1'b0;
        end
```

圖:一判斷 rst 的程式碼

有 clk 正緣來的時候，判斷此時 pass 是否為一，如果是的話，就繼續判斷現在狀態。如果不是 initial 的綠燈，就需要變成 initial 的綠燈，且 cycle 也要變成 0，state 也要變成 initial 的綠燈。但當 pass 是在 initial 的綠燈狀態時來的話，則不改變燈號 state，但 cycle 仍會繼續加一。




```
if(pass == 1'b1)
        begin
                if(state != initial_state)
                        begin
                                cycle = 10'd0;
                                state = initial_state;
                                state_count = 1'b0;
                        end
                else
                        begin
                                cycle = cycle + 10'd1;
                        end
        end
```

上圖:pass 為一的判斷條件

圖:pass 為一且狀態不為 initial 的綠燈時的波形圖

當 pass 不為一的時候，就判斷現在的 State 與 clk，當 clk+1 後的燈號等於下一個周期的燈號，而先在已經經過多少週期為 cycle+1 的數量，也因此當如果加完的 cycle 值等於 512, 64, 256 的值(還要依據現在的 state)就可以改變 state，同時也會將 cycle 數量設為 0，但 cycle 值得改變會在下一個 clk 正緣來時才會顯現。

```
else
        begin
                if(rst == 1'b0)
                        begin
                                cycle = cycle + 10'd1;
                        end
                if(cycle == 10'd512 && state == initial_state)
                        begin
                                cycle = 10'd0;
                                state = nosignal;
                        end
                else if(cycle == 10'd64 && state == nosignal)
                        begin
                                cycle = 10'd0;
                                state = green;
                        end
                else if(cycle == 10'd64 && state == green)
                        begin
                                cycle = 10'd0;
                                if(state_count == 1'b0)
                                        begin
                                                state = nosignal;
                                        end
                                else
                                        begin
                                                state = yellow;
                                        end
                                state_count = state_count + 1'b1;
                        end
                else if(cycle == 10'd256 && state == yellow)
                        begin
                                cycle = 10'd0;
                                state = red;
                        end
                else if(cycle == 10'd512 && state == red)
                        begin
                                cycle = 10'd0;
                                state = initial_state;
                        end
        end
end
```

```
module traffic_light (
    input  clk,
    input  rst,
    input  pass,
    output reg R,
    output reg G,
    output reg Y
);
        reg [9:0] cycle;
        reg [2:0] state;
        reg state_count;
        parameter initial_state = 3'd0;
        parameter nosignal = 3'd1;
        parameter green = 3'd2;
        parameter yellow = 3'd3;
        parameter red = 3'd4;
        initial
                begin
                        cycle <= 10'd0;
                        state <= initial_state;
                        state_count <= 1'b0;
                        R<=1'b0;
                        G<=1'b0;
                        Y<=1'b0;
                end
        always@(posedge clk)
                begin
                        if(pass == 1'b1)
                                begin
                                        if(state != initial_state)
                                                begin
                                                        cycle = 10'd0;
                                                        state = initial_state;
                                                        state_count = 1'b0;
                                                end
                                        else
                                                begin
                                                        cycle = cycle + 10'd1;
                                                end
                                end
                        else
                                begin
                                        if(rst == 1'b0)
                                                begin
                                                        cycle = cycle + 10'd1;
                                                end
                                        if(cycle == 10'd512 && state == initial_state)
                                                begin
                                                        cycle = 10'd0;
                                                        state = nosignal;
                                                end
                                        else if(cycle == 10'd64 && state == nosignal)
                                                begin
                                                        cycle = 10'd0;
                                                        state = green;
                                                end
                                        else if(cycle == 10'd64 && state == green)
                                                begin
                                                        cycle = 10'd0;
                                                        if(state_count == 1'b0)
                                                                begin
                                                                        state = nosignal;
                                                                end
                                                        else
                                                                begin
                                                                        state = yellow;
                                                                end
                                                        state_count = state_count + 1'b1;
                                                end
                                        else if(cycle == 10'd256 && state == yellow)
                                                begin
                                                        cycle = 10'd0;
                                                        state = red;
                                                end
                                        else if(cycle == 10'd512 && state == red)
                                                begin
                                                        cycle = 10'd0;
                                                        state = initial_state;
                                                end
                                end
                end
        always@(posedge clk)
          begin
           case(state)
               green,initial_state:
                  begin
                   Y=1'b0;
                   G=1'b1;
                   R=1'b0;
                  end
               red:
                  begin
                   Y=1'b0;
                   G=1'b0;
                   R=1'b1;
                  end
               yellow:
                  begin
                   Y=1'b1;
                   G=1'b0;
                   R=1'b0;
                  end
               default:
                  begin
                    Y=1'b0;
                    G=1'b0;
                    R=1'b0;
                  end
            endcase
          end
        always@(posedge rst)
                begin
                        cycle <= 10'd0;
                        state <= initial_state;
                        state_count <= 1'b0;
                end
endmodule
```

也因為 clk 如果有歸零，也就代表有狀態的改變，因為 state 在上一個循序電路中改變，第二個循序電路中要在下一次 clk 正緣來的時候顯示出 state 的改變，而這也正好對應到上一個循序電路 clk 做加一後也是會在下一個 clk 正緣時才顯現出來。因為在當次 clk 正緣來時，兩個循序電路會同時被觸發。

心得

（請寫下完成本次作業的心得、學到哪些東西、困難點的部分。）

在本次作業中我覺得最困難的部分就是去判斷現在 cycle 是否已經到一個 state 的上限，到底要不要換到下一個 state。因為我一開始會被作業的指示設為 clk 為一搞混，而寫出有些地方重設為 1，有些地方重設為 0，但最後統一都用換狀態後統一將 clk 設為 0 來解決。

第二個困難的地方是，剛開始搞不清楚 cycle 與 state 變換是要在下一個正緣來時改變，還是現在就改變。起初寫的時候是 cycle 是循序電路正緣出發。而 state 是由 cycle 改變就會出發是組合電路。但如此一來導致許多可能還沒有滿 512 個週期就會換狀態…等的問題。後來我又將現在兩個循序電路合併在一起寫，但也有一寫問題，所以最後就改成這樣子了!

第三個困難的地方是，一開始將 rst 為正緣時也做 cycle 加一的動作。還有 pass 為 1 時且 state 為 initial state 的時候，不會做 cycle 加一的動作。但因為找到這些問題，所以最後 debug 變得比較容易。

從此次作業中，我學到 debug 的時候，看似只差一個 cycle，本以為把某個變數調整一下即可，但調整許多都沒有成功，最後才會到波型圖去算到底經過幾個週期，才 debug 成功。讓我在之後寫作業的時候都會多利用波形圖去 debug。也將想法寫在紙上，方便確認自己的思路，而不只是光用腦袋想，讓看問題時不能看的全面。