2022 Digital IC Design Homework 2

NAME	柳譯筑
Student ID	NE6101034

Functional Simulation Result

Stage 1	Pass/Fail	Stage 2	Pass/Fail	Stage 3	Pass/Fail

Stage 1

```
# ----------------------
# -- Simulation Start --
# ----------------------
# --stage1 simulation--
#
# Setting1: PASS
#
# Setting2: PASS
#
# Setting3: PASS
#
# Setting4: PASS
#
# Setting5: PASS
#
# Setting6: PASS
#
# Setting7: PASS
#
# Setting8: PASS
#
# Setting9: PASS
#
# Setting10: PASS
#
```

(your simulation result)

Stage 2

```
# --stage2 simulation--
#
# Setting11: PASS
#
# Setting12: PASS
#
# Setting13: PASS
#
# Setting14: PASS
#
# Setting15: PASS
#
# Setting16: PASS
#
# Setting17: PASS
#
# Setting18: PASS
#
# Setting19: PASS
#
# Setting20: PASS
#
```

(your simulation result)

Stage 3

(your simulation result)

```
# --stage3 simulation--
#
# Setting21: PASS
#
# Setting22: PASS
#
# Setting23: PASS
#
# Setting24: PASS
#
# Setting25: PASS
#
# Setting26: PASS
#
# Setting27: PASS
#
# Setting28: PASS
#
# Setting29: PASS
#
# Setting30: PASS
```

Description of your design

```
module TLS(clk, reset, Set, Stop, Jump, Gin, Yin, Rin, Gout, Yout, Rout);
input           clk;
input           reset;
input           Set;
input           Stop;
input           Jump;
input      [3:0] Gin;
input      [3:0] Yin;
input      [3:0] Rin;

reg        [3:0] Gnum, Ynum, Rnum, Cnt;
reg        [3:0] NextCnt;

output          Gout;
output          Yout;
output          Rout;
reg             Gout,Yout,Rout;
reg        [1:0] State, NextState;

parameter Idle=2'b00, Green=2'b01,
          Yellow=2'b10, Red=2'b11;

// state register

always @(posedge clk or posedge reset)
begin
  if(reset)begin
    State <= Idle;
    Cnt <= 4'd0;
  end

  else begin
    State <= NextState;
    Cnt <= NextCnt;
  end
end


always @(posedge Set)
begin
    Gnum <= Gin;
    Rnum <= Rin;
    Ynum <= Yin;
end

```

我把整個專案分為三個部份，分別是 state register, next state logic 跟 output logic。

首先，先從 state register 開始，每次 clk 有一個 posedge 或是 reset 的時候 state register 負責更新下一個 state 為 NextState，還有更新 Cnt 為 NextCnt。我多加一個 Idle State，如果 reset == 1 就會進入 Idle state。

另外一個 always block 負責 Set 的控制，首先我多加了 Gnum, Rnum, Ynum 三個暫存器放 Gin, Rin, Yin 的值。如果在正緣時 Set =1，就更新紅綠燈的秒數。

```
always @(posedge Set)
begin
    Gnum <= Gin;
    Rnum <= Rin;
    Ynum <= Yin;
end

// Next state logic
always @(*)begin
  if(Set)begin
    NextState <= Green;
    NextCnt <= 4'd0;
  end
  else if (Jump)begin
    NextState <= Red;
    NextCnt <= 4'd0;
    end
  else if (Stop) begin
    NextState <= State;
    NextCnt <= Cnt;
    end

else begin
  case(State)
    Green:begin
            if (Cnt == Gnum-4'd1) begin
              NextState <= Yellow;
              NextCnt <= 4'd0;
            end else begin
              NextState <= Green;
              NextCnt <= Cnt+4'd1;
            end
          end

    Yellow:begin
             if (Cnt == Ynum-4'd1) begin
               NextState <= Red;
               NextCnt <= 4'd0;
             end else begin
               NextState <= Yellow;
               NextCnt <= Cnt+4'd1;
             end
           end
    Red:begin
          if (Cnt == Rnum-4'd1) begin
              NextState <= Green;
              NextCnt <= 4'd0;
          end else begin
            NextState <= Red;
            NextCnt <= Cnt+4'd1;
          end
        end
    default:
      begin
        NextState <= Idle;
        NextCnt <= Cnt;
      end
    endcase
    end
end
```

接著是 Next State Logic，這個 always block 負責處理下一個 state 在哪個情況下要接哪個 State 跟 Cnt。

有三個影響的控制訊號分別是 Set, Jump, Stop。

當 Set 時下個狀態會進入綠色，下個 Cnt = 0，當 Jump 下個狀態會進入紅色，下個 Cnt = 0，當 Stop 時會維持現在的 State 跟 Cnt。

當沒有這些訊號則是做 State 的邏輯判斷，在綠燈時當 Cnt 是 Gnum-1 時下個狀態就是黃燈並且計數器會歸零，否則下一個狀態仍然是綠燈且計數器要

+1，黃燈跟紅燈也是同樣的判斷邏輯。
另外我加 default 狀態則是下個狀態為 Idle，下個 Cnt 維持原本的值。

```
99
100     // output logic
101     always @(State)begin
102       case(State)
103         Green:begin
104                 Gout = 1'b1;
105                 Rout = 1'b0;
106                 Yout = 1'b0;
107                 end
108         Yellow:begin
109                 Gout = 1'b0;
110                 Rout = 1'b0;
111                 Yout = 1'b1;
112               end
113         Red:begin
114                 Gout = 1'b0;
115                 Rout = 1'b1;
116                 Yout = 1'b0;
117               end
118         Idle:begin
119                 Gout = 1'b0;
120                 Rout = 1'b0;
121                 Yout = 1'b0;
122               end
123         default:
124         begin
125           Gout = 1'b0;
126           Rout = 1'b0;
127           Yout = 1'b0;
128           end
129         endcase
130     end
131
```

最後是 output logic，控制當 state 分別為綠燈、黃燈、紅燈的時候，Gout, Rout, Yout 分別是 0/1 的 output 。
以上是我用 Moore machine 跟 Flip flop 控制紅綠燈的方式，感謝助教批閱。