module Traffic\_Light\_Controller (RST, CLK, GO, HS1, HS2, FS1, FS2);

input RST, CLK, GO;

reg [4:0]counter;

output reg [1:0] HS1, HS2, FS1, FS2;

parameter s0=0,s1=1,s2=2,s3=3,s4=4,s5=5,s6=6,s7=7,s8=8,s9=9,s10=10,s11=11,s12=12,s13=13,s14=14,s15=15,s16=16,s17=17; //these numbers are assigned to each state 5-bits

parameter red=2'b10, green= 2'b00, yellow=2'b01, redyellow=2'b11; //lights are assigned to the outputs

reg [4:0]state; // 18 state each on of them has a 5 bit

parameter delay1=1,delay2=2,delay3=3,delay5=5,delay10=10,delay15=15,delay30=30;//the delay for each state

initial

begin

state=s0;

counter=5'b00000;

end

always@(posedge CLK or negedge RST or GO)

begin

if (RST==1) //if reset is 1 then return to state0 and counter is 0

begin

state = s0;

counter = 5'b00000;

end

else

begin

case(state)

s0: case(GO)// check values of GO

1'b0: begin //if GO is zero then freeze the counter as well as the state

state =s0;

counter = counter;

end

1'b1: begin //if GO is one then check the counter to decide the movement either to stay until the counter limit is achieved then move to the next state

if (counter <delay1)

begin

state=s0;

counter=counter+1;

end

else if (counter ==delay1)

begin

state =s1;

counter =0;

end

end

endcase

s1: case(GO)

1'b0: begin

state =s1;

counter = counter;

end

1'b1: begin

if (counter <delay1)

begin

state =s1;

counter=counter+1;

end

else if (counter ==delay1)

begin

state =s2;

counter =0;

end

end

endcase

s2: case(GO)

1'b0: begin

state =s2;

counter = counter;

end

1'b1: begin

if (counter <delay30)

begin

state=s2;

counter=counter+1;

end

else if (counter ==delay30)

begin

state =s3;

counter =0;

end

end

endcase

s3: case(GO)

1'b0: begin

state =s3;

counter = counter;

end

1'b1: begin

if (counter <delay2)

begin

state=s3;

counter=counter+1;

end

else if (counter ==delay2)

begin

state =s4;

counter =0;

end

end

endcase

s4:case(GO)

1'b0: begin

state =s4;

counter = counter;

end

1'b1: begin

if (counter <delay10)

begin

state=s4;

counter=counter+1;

end

else if (counter ==delay10)

begin

state =s5;

counter =0;

end

end

endcase

s5: case(GO)

1'b0: begin

state =s5;

counter = counter;

end

1'b1: begin

if (counter <delay2)

begin

state=s5;

counter=counter+1;

end

else if (counter ==delay2)

begin

state =s6;

counter =0;

end

end

endcase

s6: case(GO)

1'b0: begin

state =s6;

counter = counter;

end

1'b1: begin

if (counter <delay1)

begin

state=s6;

counter=counter+1;

end

else if (counter ==delay1)

begin

state =s7;

counter =0;

end

end

endcase

s7: case(GO)

1'b0: begin

state =s7;

counter = counter;

end

1'b1: begin

if (counter <delay2)

begin

state=s7;

counter=counter+1;

end

else if (counter ==delay2)

begin

state =s8;

counter =0;

end

end

endcase

s8: case(GO)

1'b0: begin

state =s8;

counter = counter;

end

1'b1: begin

if (counter <delay15)

begin

state=s8;

counter=counter+1;

end

else if (counter ==delay15)

begin

state =s9;

counter =0;

end

end

endcase

s9: case(GO)

1'b0: begin

state =s9;

counter = counter;

end

1'b1: begin

if (counter <delay2)

begin

state=s9;

counter=counter+1;

end

else if (counter ==delay2)

begin

state =s10;

counter =0;

end

end

endcase

s10:case(GO)

1'b0: begin

state =s10;

counter = counter;

end

1'b1: begin

if (counter <delay5)

begin

state=s10;

counter=counter+1;

end

else if (counter ==delay5)

begin

state =s11;

counter =0;

end

end

endcase

s11: case(GO)

1'b0: begin

state =s11;

counter = counter;

end

1'b1: begin

if (counter <delay2)

begin

state=s11;

counter=counter+1;

end

else if (counter ==delay2)

begin

state =s12;

counter=0;

end

end

endcase

s12:case(GO)

1'b0: begin

state =s12;

counter = counter;

end

1'b1: begin

if (counter <delay10)

begin

state=s12;

counter=counter+1;

end

else if (counter ==delay10)

begin

state =s13;

counter =0;

end

end

endcase

s13:case(GO)

1'b0: begin

state =s13;

counter = counter;

end

1'b1: begin

if (counter <delay2)

begin

state=s13;

counter=counter+1;

end

else if (counter ==delay2)

begin

state =s14;

counter =0;

end

end

endcase

s14:case(GO)

1'b0: begin

state =s14;

counter = counter;

end

1'b1: begin

if (counter <delay1)

begin

state=s14;

counter=counter+1;

end

else if (counter ==delay1)

begin

state =s15;

counter =0;

end

end

endcase

s15:case(GO)

1'b0: begin

state =s15;

counter = counter;

end

1'b1: begin

if (counter <delay2)

begin

state=s15;

counter=counter+1;

end

else if (counter ==delay2)

begin

state =s16;

counter =0;

end

end

endcase

s16:case(GO)

1'b0: begin

state =s16;

counter = counter;

end

1'b1: begin

if (counter <delay15)

begin

state=s16;

counter=counter+1;

end

else if (counter ==delay15)

begin

state =s17;

counter =0;

end

end

endcase

s17: case(GO)

1'b0: begin

state =s17;

counter = counter;

end

1'b1: begin

if (counter <delay3)

begin

state=s17;

counter=counter+1;

end

else if (counter ==delay3)

begin

state =s0;

counter =0;

end

end

endcase

endcase

end

end

always @(state) //assigning each state with its outputs (four lights on each lane)

begin

case(state)

s0:begin

HS1= red;

HS2= red;

FS1= red;

FS2= red;

end

s1:begin

HS1= redyellow;

HS2= redyellow;

FS1= red;

FS2= red;

end

s2:begin

HS1= green;

HS2= green;

FS1= red;

FS2= red;

end

s3:begin

HS1= green;

HS2= yellow;

FS1= red;

FS2= red;

end

s4:begin

HS1= green;

HS2= red;

FS1= red;

FS2= red;

end

s5:begin

HS1= yellow;

HS2= red;

FS1= red;

FS2= red;

end

s6:begin

HS1= red;

HS2= red;

FS1= red;

FS2= red;

end

s7:begin

HS1= red;

HS2= red;

FS1= redyellow;

FS2= redyellow;

end

s8:begin

HS1= red;

HS2= red;

FS1= green;

FS2= green;

end

s9:begin

HS1= red;

HS2= red;

FS1= green;

FS2= yellow;

end

s10:begin

HS1= red;

HS2= red;

FS1= green;

FS2= red;

end

s11:begin

HS1= red;

HS2= red;

FS1= yellow;

FS2= redyellow;

end

s12:begin

HS1= red;

HS2= red;

FS1= red;

FS2= green;

end

s13:begin

HS1= red;

HS2= red;

FS1= red;

FS2= yellow;

end

s14:begin

HS1= red;

HS2= red;

FS1= red;

FS2= red;

end

s15:begin

HS1= red;

HS2= redyellow;

FS1= red;

FS2= red;

end

s16:begin

HS1= red;

HS2= green;

FS1= red;

FS2= red;

end

s17:begin

HS1= red;

HS2= yellow;

FS1= red;

FS2= red;

end

endcase

end

endmodule

//generator module to save the expected values in memory to be able to compare them in the analyzer module

module Traffic\_Light\_Controller\_Generator(RST, CLK, GO, HS1EXP, HS2EXP, FS1EXP, FS2EXP);

input RST, GO, CLK;

output reg [1:0] HS1EXP, HS2EXP, FS1EXP, FS2EXP;

integer k=0;

//unpacked memory array to save the outputs of all states and assign them to the outputs

reg [1:0] mem [0:71]= {2'b10,2'b10,2'b10,2'b10,2'b11,2'b11,2'b10,2'b10,2'b00,2'b00,2'b10,2'b10,2'b00,2'b01,2'b10,2'b10,2'b00,2'b10,2'b10,2'b10,2'b01,2'b10,2'b10,2'b10,2'b10,2'b10,2'b10,2'b10,2'b10,2'b10,2'b11,2'b11,2'b10,2'b10,2'b00,2'b00,2'b10,2'b10,2'b00,2'b01,2'b10,2'b10,2'b00,2'b10,2'b10,2'b10,2'b01,2'b11,2'b10,2'b10,2'b10,2'b00,2'b10,2'b10,2'b10,2'b01,2'b10,2'b10,2'b10,2'b10,2'b10,2'b11,2'b10,2'b10,2'b10,2'b00,2'b10,2'b10,2'b10,2'b01,2'b10,2'b10};

always@ (posedge CLK or negedge RST or GO)

begin

HS1EXP= mem[k];

HS2EXP= mem[k+1];

FS1EXP= mem[k+2];

FS2EXP= mem[k+3];

assign k=k+4;

end

endmodule

//compare the expected results with the outputs of the traffic module and raise error is not matching

module Traffic\_Light\_Controller\_Analyzer (RST, CLK, GO, HS1, HS2, FS1, FS2, HS1EXP, HS2EXP, FS1EXP, FS2EXP, ERROR);

input CLK, RST, GO;

input [1:0] HS1, HS2, FS1, FS2, HS1EXP, HS2EXP, FS1EXP, FS2EXP;

output reg ERROR;

always @(posedge CLK or negedge RST or GO)begin

if (RST)

ERROR=1'b0;

else

begin

if (HS1 != HS1EXP || HS2 != HS2EXP || FS1 != FS1EXP || FS2 != FS2EXP) //if not matched then error is 1

ERROR=1'b1;

else

ERROR=1'b0; //if matching then error is 0

end

if (ERROR == 1'b1)

$display("FAIL, VALUES ARE NOT MATCHED!");

else

$display("PASS");

end

endmodule

//test bench of the traffic light controller module

module Traffic\_Light\_Controller\_tb;

reg RST, GO, CLK;

wire [1:0] HS1, HS2, FS1, FS2, HS1EXP, HS2EXP, FS1EXP, FS2EXP;

wire ERROR;

//calling the previous modules to examine their outputs and therefore validity

Traffic\_Light\_Controller TLC (RST, CLK, GO, HS1, HS2, FS1, FS2);

Traffic\_Light\_Controller\_Generator TLCG (RST, CLK, GO, HS1EXP, HS2EXP, FS1EXP, FS2EXP);

Traffic\_Light\_Controller\_Analyzer TLCA (RST, CLK, GO, HS1, HS2, FS1, FS2, HS1EXP, HS2EXP, FS1EXP, FS2EXP, ERROR);

initial

begin

RST=0; GO=1; CLK=0;

$monitor("Highway HS1 = %b, Highway HS2 = %b, Farm Road FS1 = %b, Farm Road FS2 = %b at time %d", HS1, HS2, FS1, FS2, $time);

#20

RST=1;

#30

RST=1;

#20;

RST=0;

GO=0;

#40;

GO=1;

end

always CLK = #5ns ~CLK;

endmodule