Traffic\_light\_with\_Delay\_units

**Code ;**

module traffic\_light(

input clk,rst,

output reg ns\_g,ns\_r,ns\_y,ew\_g,ew\_y,ew\_r);

//internal states

parameter nsgewr=3'b000;

parameter nsyewr=3'b001;

parameter nsrewr=3'b010;

parameter nsrewg=3'b011;

parameter nsrewy=3'b100;

parameter nsrewr\_1=3'b101;

//internal registers

reg [7:0]counter;

reg [2:0]current\_state,next\_state;

//counter\_delay

parameter green=8'd15;

parameter yellow=8'd3;

parameter red=8'd3;

always@(posedge clk)

begin

if(rst)

begin

current\_state<=nsgewr;

counter<=8'd0;

end

else

begin

if (counter != 0)

counter <= counter - 1;

else

//counter logic

begin

current\_state<=next\_state;

case(current\_state)

nsgewr:counter<=green;

nsyewr:counter<=yellow;

nsrewr:counter<=red;

nsrewg:counter<=green;

nsrewy:counter<=yellow;

nsrewr\_1:counter<=red;

default:counter<=green;

endcase

end

end

end

//next\_state

always@(\*)

begin

case(current\_state)

nsgewr:next\_state<=nsyewr;

nsyewr:next\_state<=nsrewr;

nsrewr:next\_state<=nsrewg;

nsrewg:next\_state<=nsrewy;

nsrewy:next\_state<=nsrewr\_1;

nsrewr\_1:next\_state<=nsgewr;

default:next\_state<=nsgewr;

endcase

end

always@(\*)

begin

ns\_r<=0;

ns\_y<=0;

ew\_g<=0;

ew\_y<=0;

ew\_r<=0;

ns\_g<=0;

case(current\_state)

nsgewr:

begin

ns\_g<=1;

ew\_r<=1;

end

nsyewr:

begin

ns\_y<=1;

ew\_r<=1;

end

nsrewr:

begin

ns\_r<=1;

ew\_r<=1;

end

nsrewg:

begin

ns\_r<=1;

ew\_g<=1;

end

nsrewy:

begin

ns\_r<=1;

ew\_y<=1;

end

nsrewr\_1:

begin

ns\_r<=1;

ew\_r<=1;

end

endcase

end

endmodule

Verified with DigitalJS

