06/02/16 06:05:11 C:\Users\Danish\Files\Dropbox\College\2nd Year\Q3\ECE 152A\Lab #3\finalCode.v

```
`timescale 1ns / 1ps
//Danish Vaid, Dylan Goldsworthy, Anubhav Basak
//What the code does:
//This code is written to create the circuit for a Ford Thunderbird
//tail light controller
//Inputs: clk, reset, brake, hazard, left, right [listed in our files as x0x1x2x3x4]
//Ouputs: Lc, Lb, La, Ra, Rb, Rc

//-----------------------------------------------------------------------------------------------------
// Main Function
//-----------------------------------------------------------------------------------------------------
module tailLightControllerMain(dimClk, lights, clk, reset, brake, hazard, left, right, Lc, Lb, La, Ra, Rb, Rc);
input dimClk, lights, clk, reset, brake, hazard, left, right;
output Lc, Lb, La, Ra, Rb, Rc;
wire[2:0] Lcba;             //Temp Memory Storage
wire[2:0] Rabc;
tailLightControllerStateMachine f1(clk, reset, brake, hazard, left, right, Lcba, Rabc);
tailLightControllerDimmer   d1(dimClk, lights, Lcba, Rabc, Lc, Lb, La, Ra, Rb, Rc);
endmodule
//-----------------------------------------------------------------------------------------------------
// Controls when the lights get checked for dim or not
//-----------------------------------------------------------------------------------------------------
module tailLightControllerDimmer(input dimClk, input lights, input[2:0] Lcba, input[2:0] Rabc, output reg Lc, output reg Lb,
output reg La, output reg Ra,output reg Rb, output reg Rc);
reg toggle;             //Dimming Helper
always@(posedge dimClk ) begin
if(lights) begin
toggle <= ~toggle;

if(Lcba[2]) begin
Lc = 1'b1;
end
else begin
Lc = toggle;
end

if(Lcba[1]) begin
Lb = 1'b1;
end
else begin
Lb = toggle;
end

if(Lcba[0]) begin
La = 1'b1;
end
else begin
La = toggle;
end

if(Rabc[1]) begin
Ra = 1'b1;
end
else begin
Ra = toggle;
end

if(Rabc[1]) begin
Rb = 1'b1;
end
else begin
Rb = toggle;
end

if(Rabc[0]) begin
Rc = 1'b1;
end
else begin
Rc = toggle;
end

end
else begin
Lc = Lcba[2];
Lb = Lcba[1];
La = Lcba[0];
Ra = Rabc[2];
Rb = Rabc[1];
Rc = Rabc[0];
end
end
endmodule

```

```verilog
//------------------------------------------------------------------------------------------------------------
// State machine for the circuit
//------------------------------------------------------------------------------------------------------------
module tailLightControllerStateMachine(input clk, input reset, input brake, input hazard, input left, input right, output
reg[2:0] Lcba, output reg[2:0] Rabc);
//------------------------------------------------------------------------------------------
// Local Parameters All possible states
//------------------------------------------------------------------------------------------
`define state_off   4'd0
`define state_brake   4'd1
`define state_l1   4'd2
`define state_l2   4'd3
`define state_l3   4'd4
`define state_r1   4'd5
`define state_r2   4'd6
`define state_r3   4'd7
`define state_bl1   4'd8
`define state_bl2   4'd9
`define state_br1   4'd10
`define state_br2   4'd11
`define state_hazard  4'd12
//------------------------------------------------------------------------------------------
// Registry States
//------------------------------------------------------------------------------------------
reg[3:0] currentState;            //Current state value
reg[3:0] nextState;             //Next state value
//------------------------------------------------------------------------------------------
// Outputs
//------------------------------------------------------------------------------------------
always@( * ) begin
case(currentState)
`state_off: begin
    Lcba = 3'b000;            //Can change to be Lc = 1_b0
    Rabc = 3'b000;
    end
    `state_brake: begin
    Lcba = 3'b111;
    Rabc = 3'b111;
    end
    `state_l1: begin
    Lcba = 3'b001;
    Rabc = 3'b000;
    end
    `state_l2: begin
    Lcba = 3'b011;
    Rabc = 3'b000;
    end
    `state_l3: begin
    Lcba = 3'b111;
    Rabc = 3'b000;
    end
    `state_r1: begin
    Lcba = 3'b000;
    Rabc = 3'b100;
    end
    `state_r2: begin
    Lcba = 3'b000;
    Rabc = 3'b110;
    end
    `state_r3: begin
    Lcba = 3'b000;
    Rabc = 3'b111;
    end
    `state_bl1: begin
    Lcba = 3'b001;
    Rabc = 3'b111;
    end
    `state_bl2: begin
    Lcba = 3'b011;
    Rabc = 3'b111;
    end
    `state_br1: begin
    Lcba = 3'b111;
    Rabc = 3'b100;
    end
    `state_br2: begin
    Lcba = 3'b111;
    Rabc = 3'b110;
    end
    `state_hazard: begin
    Lcba = 3'b111;
    Rabc = 3'b111;
    end
    endcase
    end
```

```
//--------------------------------------------------------------------------------------------------
// Change Current to Next State
//--------------------------------------------------------------------------------------------------
always@(posedge clk) begin
if(reset) currentState <= `state_off;
else currentState <= nextState;
end
//--------------------------------------------------------------------------------------------------
// Next State Selector [xxxxx]
//--------------------------------------------------------------------------------------------------
always@( * ) begin
nextState = currentState;
  if(reset) begin                //If 1xxxx
  nextState = `state_off;
  end
  else if(!reset && brake && !left && !right) begin     //If 01x00
  nextState = `state_brake;
  end
  else if(!reset && brake && left && right) begin      //If 01x11
  nextState = `state_brake;
  end
  else if(!reset && !brake && hazard && (currentState != `state_hazard)) begin
   nextState = `state_hazard;            //If 001xx
   end
   else if(!reset && !brake && !hazard && left && right && (currentState != `state_hazard)) begin
   nextState = `state_hazard;            //If 00011
   end
  else if(!reset && !brake && !hazard && !left && !right) begin //If 00000
  nextState = `state_off;
  end
  else begin
  case (currentState)
    `state_off: begin              //state_off
    if(!reset && brake && !left && right) nextState = `state_br1;
    if(!reset && brake && left && !right) nextState = `state_bl1;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
    end
    `state_brake: begin
    if(!reset && brake && !left && right) nextState = `state_br1;
    if(!reset && brake && left && !right) nextState = `state_bl1;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
    end
    `state_l1: begin
    if(!reset && brake && !left && right) nextState = `state_br1;
    if(!reset && brake && left && !right) nextState = `state_bl2;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_l2;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
    end
    `state_l2: begin
    if(!reset && brake && !left && right) nextState = `state_br1;
    if(!reset && brake && left && !right) nextState = `state_brake;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_l3;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
    end
    `state_l3: begin
    if(!reset && brake && !left && right) nextState = `state_br1;
    if(!reset && brake && left && !right) nextState = `state_r3;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_off;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
    end
    `state_r1: begin
    if(!reset && brake && !left && right) nextState = `state_br2;
    if(!reset && brake && left && !right) nextState = `state_bl1;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_r2;
    end
    `state_r2: begin
    if(!reset && brake && !left && right) nextState = `state_brake;
    if(!reset && brake && left && !right) nextState = `state_bl1;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_r3;
    end
    `state_r3: begin
    if(!reset && brake && !left && right) nextState = `state_l3;
    if(!reset && brake && left && !right) nextState = `state_bl1;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_off;
    end
    `state_bl1: begin
    if(!reset && brake && !left && right) nextState = `state_br1;
    if(!reset && brake && left && !right) nextState = `state_bl2;
    if(!reset && !brake && !hazard && left && !right) nextState = `state_l2;
    if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
```

```
end
`state_bl2: begin
if(!reset && brake && !left && right) nextState = `state_br1;
if(!reset && brake && left && !right) nextState = `state_brake;
if(!reset && !brake && !hazard && left && !right) nextState = `state_l3;
if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
end
`state_br1: begin
if(!reset && brake && !left && right) nextState = `state_br2;
if(!reset && brake && left && !right) nextState = `state_bl1;
if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
if(!reset && !brake && !hazard && !left && right) nextState = `state_r2;
end
`state_br2: begin
if(!reset && brake && !left && right) nextState = `state_brake;
if(!reset && brake && left && !right) nextState = `state_bl1;
if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
if(!reset && !brake && !hazard && !left && right) nextState = `state_r3;
end
`state_hazard: begin
if(!reset && brake && !left && right) nextState = `state_br1;
if(!reset && brake && left && !right) nextState = `state_bl1;
if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
if(!reset && !brake && hazard) nextState = `state_off;
if(!reset && !brake && !hazard && left && right) nextState = `state_off;
end
//default: nextState = state_off;     //Not sure whether to implement

endcase
end
end
endmodule
```