

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

```

1  `timescale 1ns / 1ps
2  //Danish Vaid, Dylan Goldsworthy, Anubhav Basak
3  //What the code does:
4  //This code is written to create the circuit for a Ford Thunderbird
5  //tail light controller
6  //Inputs: clk, reset, brake, hazard, left, right [listed in our files as x0x1x2x3x4]
7  //Outputs: Lc, Lb, La, Ra, Rb, Rc
8
9  //-----
10 // Main Function
11 //-----
12 module taillightControllerMain(dimClk, lights, clk, reset, brake, hazard, left, right, Lc, Lb, La, Ra, Rb, Rc);
13 input dimClk, lights, clk, reset, brake, hazard, left, right;
14 output Lc, Lb, La, Ra, Rb, Rc;
15 wire[2:0] Lcba; //Temp Memory Storage
16 wire[2:0] Rabc;
17 taillightControllerStateMachine f1(clk, reset, brake, hazard, left, right, Lcba, Rabc);
18 taillightControllerDimmer d1(dimClk, lights, Lcba, Rabc, Lc, Lb, La, Ra, Rb, Rc);
19 endmodule
20 //-----
21 // Controls when the lights get checked for dim or not
22 //-----
23 module taillightControllerDimmer(input dimClk, input lights, input[2:0] Lcba, input[2:0] Rabc, output reg Lc, output reg Lb,
24 output reg La, output reg Ra, output reg Rb, output reg Rc);
25 reg toggle; //Dimming Helper
26 always@(posedge dimClk) begin
27 if(lights) begin
28 toggle <= ~toggle;
29
30 if(Lcba[2]) begin
31 Lc = 1'b1;
32 end
33 else begin
34 Lc = toggle;
35 end
36
37 if(Lcba[1]) begin
38 Lb = 1'b1;
39 end
40 else begin
41 Lb = toggle;
42 end
43
44 if(Lcba[0]) begin
45 La = 1'b1;
46 end
47 else begin
48 La = toggle;
49 end
50
51 if(Rabc[1]) begin
52 Ra = 1'b1;
53 end
54 else begin
55 Ra = toggle;
56 end
57
58 if(Rabc[1]) begin
59 Rb = 1'b1;
60 end
61 else begin
62 Rb = toggle;
63 end
64
65 if(Rabc[0]) begin
66 Rc = 1'b1;
67 end
68 else begin
69 Rc = toggle;
70 end
71
72 end
73 else begin
74 Lc = Lcba[2];
75 Lb = Lcba[1];
76 La = Lcba[0];
77 Ra = Rabc[2];
78 Rb = Rabc[1];
79 Rc = Rabc[0];
80 end
81 endmodule
82

```

```

83 //-----
84 // State machine for the circuit
85 //-----
86 module taillightControllerStateMachine(input clk, input reset, input brake, input hazard, input left, input right, output
reg[2:0] Lcba, output reg[2:0] Rabc);
87 //-----
88 // Local Parameters All possible states
89 //-----
90 `define state_off 4'd0
91 `define state_brake 4'd1
92 `define state_l1 4'd2
93 `define state_l2 4'd3
94 `define state_l3 4'd4
95 `define state_r1 4'd5
96 `define state_r2 4'd6
97 `define state_r3 4'd7
98 `define state_bl1 4'd8
99 `define state_bl2 4'd9
100 `define state_br1 4'd10
101 `define state_br2 4'd11
102 `define state_hazard 4'd12
103 //-----
104 // Registry States
105 //-----
106 reg[3:0] currentState; //Current state value
107 reg[3:0] nextState; //Next state value
108 //-----
109 // Outputs
110 //-----
111 always@( * ) begin
112 case(currentState)
113 `state_off: begin
114 Lcba = 3'b000; //Can change to be Lc = 1_b0
115 Rabc = 3'b000;
116 end
117 `state_brake: begin
118 Lcba = 3'b111;
119 Rabc = 3'b111;
120 end
121 `state_l1: begin
122 Lcba = 3'b001;
123 Rabc = 3'b000;
124 end
125 `state_l2: begin
126 Lcba = 3'b011;
127 Rabc = 3'b000;
128 end
129 `state_l3: begin
130 Lcba = 3'b111;
131 Rabc = 3'b000;
132 end
133 `state_r1: begin
134 Lcba = 3'b000;
135 Rabc = 3'b100;
136 end
137 `state_r2: begin
138 Lcba = 3'b000;
139 Rabc = 3'b110;
140 end
141 `state_r3: begin
142 Lcba = 3'b000;
143 Rabc = 3'b111;
144 end
145 `state_bl1: begin
146 Lcba = 3'b001;
147 Rabc = 3'b111;
148 end
149 `state_bl2: begin
150 Lcba = 3'b011;
151 Rabc = 3'b111;
152 end
153 `state_br1: begin
154 Lcba = 3'b111;
155 Rabc = 3'b100;
156 end
157 `state_br2: begin
158 Lcba = 3'b111;
159 Rabc = 3'b110;
160 end
161 `state_hazard: begin
162 Lcba = 3'b111;
163 Rabc = 3'b111;
164 end
165 endcase
166 end

```

```

167 //-----
168 // Change Current to Next State
169 //-----
170 always@(posedge clk) begin
171   if(reset) currentState <= `state_off;
172   else currentState <= nextState;
173 end
174 //-----
175 // Next State Selector [xxxxx]
176 //-----
177 always@( * ) begin
178   nextState = currentState;
179   if(reset) begin //If 1xxxx
180     nextState = `state_off;
181   end
182   else if(!reset && brake && !left && !right) begin //If 01x00
183     nextState = `state_brake;
184   end
185   else if(!reset && brake && left && right) begin //If 01x11
186     nextState = `state_brake;
187   end
188   else if(!reset && !brake && hazard && (currentState != `state_hazard)) begin
189     nextState = `state_hazard; //If 001xx
190   end
191   else if(!reset && !brake && !hazard && left && right && (currentState != `state_hazard)) begin
192     nextState = `state_hazard; //If 00011
193   end
194   else if(!reset && !brake && !hazard && !left && !right) begin //If 00000
195     nextState = `state_off;
196   end
197   else begin
198     case (currentState)
199       `state_off: begin //state_off
200         if(!reset && brake && !left && right) nextState = `state_br1;
201         if(!reset && brake && left && !right) nextState = `state_bl1;
202         if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
203         if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
204       end
205       `state_brake: begin
206         if(!reset && brake && !left && right) nextState = `state_br1;
207         if(!reset && brake && left && !right) nextState = `state_bl1;
208         if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
209         if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
210       end
211       `state_l1: begin
212         if(!reset && brake && !left && right) nextState = `state_br1;
213         if(!reset && brake && left && !right) nextState = `state_bl2;
214         if(!reset && !brake && !hazard && left && !right) nextState = `state_l2;
215         if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
216       end
217       `state_l2: begin
218         if(!reset && brake && !left && right) nextState = `state_br1;
219         if(!reset && brake && left && !right) nextState = `state_brake;
220         if(!reset && !brake && !hazard && left && !right) nextState = `state_l3;
221         if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
222       end
223       `state_l3: begin
224         if(!reset && brake && !left && right) nextState = `state_br1;
225         if(!reset && brake && left && !right) nextState = `state_r3;
226         if(!reset && !brake && !hazard && left && !right) nextState = `state_off;
227         if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
228       end
229       `state_r1: begin
230         if(!reset && brake && !left && right) nextState = `state_br2;
231         if(!reset && brake && left && !right) nextState = `state_bl1;
232         if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
233         if(!reset && !brake && !hazard && !left && right) nextState = `state_r2;
234       end
235       `state_r2: begin
236         if(!reset && brake && !left && right) nextState = `state_brake;
237         if(!reset && brake && left && !right) nextState = `state_bl1;
238         if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
239         if(!reset && !brake && !hazard && !left && right) nextState = `state_r3;
240       end
241       `state_r3: begin
242         if(!reset && brake && !left && right) nextState = `state_l3;
243         if(!reset && brake && left && !right) nextState = `state_bl1;
244         if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
245         if(!reset && !brake && !hazard && !left && right) nextState = `state_off;
246       end
247       `state_bl1: begin
248         if(!reset && brake && !left && right) nextState = `state_br1;
249         if(!reset && brake && left && !right) nextState = `state_bl2;
250         if(!reset && !brake && !hazard && left && !right) nextState = `state_l2;
251         if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;

```

```
252     end
253     `state_b12: begin
254     if(!reset && brake && !left && right) nextState = `state_br1;
255     if(!reset && brake && left && !right) nextState = `state_brake;
256     if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
257     if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
258     end
259     `state_br1: begin
260     if(!reset && brake && !left && right) nextState = `state_br2;
261     if(!reset && brake && left && !right) nextState = `state_b11;
262     if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
263     if(!reset && !brake && !hazard && !left && right) nextState = `state_r2;
264     end
265     `state_br2: begin
266     if(!reset && brake && !left && right) nextState = `state_brake;
267     if(!reset && brake && left && !right) nextState = `state_b11;
268     if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
269     if(!reset && !brake && !hazard && !left && right) nextState = `state_r3;
270     end
271     `state_hazard: begin
272     if(!reset && brake && !left && right) nextState = `state_br1;
273     if(!reset && brake && left && !right) nextState = `state_b11;
274     if(!reset && !brake && !hazard && left && !right) nextState = `state_l1;
275     if(!reset && !brake && !hazard && !left && right) nextState = `state_r1;
276     if(!reset && !brake && hazard) nextState = `state_off;
277     if(!reset && !brake && !hazard && left && right) nextState = `state_off;
278     end
279     //default: nextState = state_off;    //Not sure whether to implement
280
281     endcase
282     end
283     end
284 endmodule
```