Traffic Lights Controller
Part 1

Ain Shams University
Faculty of Engineering
Electronics and Communication Engineering Department
ECE 413: ASIC

Team Members:
Amira Bahey Eldin Mahmoud (2000067)
Rahaf Abdullah Atef Abdullah (2000099)
Asmaa Abdelmotlb Yousseff (2000051)
Abdallah Karim Motwea (2000993)

Contribution:
We all contributed to the design and verification of the project.

December 10, 2024

Contents

1 System Design **2**
- 1.1 Purpose . . . 2
- 1.2 Inputs and Outputs . . . 2
- 1.3 Goals . . . 2
- 1.4 Functional Requirements: . . . 2
- 1.5 The functionality of the RTL code: . . . 3
- 1.6 Traffic Light Algorithm . . . 4
- 1.7 Controller Specifications . . . 5

2 FSM Design **6**
- 2.1 FSM Design . . . 6
- 2.2 FSM diagrams . . . 8

3 RTL Code **9**
- 3.1 Overview . . . 9
- 3.2 Code . . . 9

4 Testbench **18**
- 4.1 Purpose . . . 18
- 4.2 Testbench Code . . . 18

5 Simulation **24**
- 5.1 Waveform Analysis . . . 24
- 5.2 Snippets results with identification for each test case on wave form and transcript 27

6 Conclusion **30**

CHAPTER 1

System Design

1.1 Purpose

The objective of this project is to design a traffic light control system for four intersections (TF_1, TF_2, TF_3, TF_4). The system ensures efficient traffic management while preventing conflicting green lights at perpendicular intersections.

1.2 Inputs and Outputs

Inputs

- **clk**: System clock for timing.
- **rst**: Reset signal to initialize the system.
- **s_tf1, s_tf2, s_tf3, s_tf4**: Sensors indicating vehicle presence at each traffic light.

Outputs

- **TF_1, TF_2, TF_3, TF_4**: Traffic light signals with 3-bit states (RED, YELLOW, GREEN).

1.3 Goals

- **Efficient traffic management.**
- **Conflict-free operation:** No two perpendicular lights should be GREEN simultaneously.

1.4 Functional Requirements:

- Lights cycle through RED → YELLOW → GREEN → YELLOW → RED.
- Support priority overrides based on sensor inputs.
- Handle edge cases like inactive sensors or conflicting requests.

1.5 The functionality of the RTL code:

- The RTL (Register Transfer Level) code implements the traffic light control system for four intersections.
- Each traffic light transition between the states RED, YELLOW, and GREEN, represented as 3-bit signals, based on a defined timing sequence and priority determined by sensor inputs.
- The system ensures safe and conflict-free operation by preventing perpendicular traffic lights from being GREEN simultaneously.
- It incorporates extended GREEN time for intersections with higher sensor priority, dynamically adjusts the light cycle based on real-time input, and ensures all lights default to RED during inactive periods.
- The design uses a finite state machine (FSM) for state transitions, synchronized with a clock (clk), and includes a reset signal (rst) to initialize or reset the system.

1.6 Traffic Light Algorithm

Purpose:

- Control the transition of traffic light states based on timing and sensor inputs.

1. **States:**
   - **RED**: `000`
   - **YELLOW**: `001`
   - **GREEN**: `010`

2. **Transition Rules:**
   - **Default Cycle:**
     * **RED** → **YELLOW** → **GREEN** → **YELLOW** → **RED**
   - **Green Extension:**
     * If sensor input (e.g., `s_tfX = 2'b11`) indicates heavy traffic, the corresponding light remains **GREEN** for a longer duration.
   - **Priority Handling:**
     * Sensor inputs determine which traffic light gets the **GREEN** state when multiple intersections request it.
   - **Idle State:**
     * All traffic lights remain **RED** when no sensor inputs are active.

3. **timing:**
   - Default durations for GREEN, YELLOW, and RED (e.g., GREEN = 70 units, YELLOW = 30 units, extended green = 100 units).
   - Adjustable durations for extended GREEN based on sensor inputs.

1.7 Controller Specifications

Purpose: Define the technical requirements for the traffic light controller.

1 Inputs:

- **clk**: System clock signal.
- **rst**: System reset signal.
- **s_tf1, s_tf2, s_tf3, s_tf4**: Sensor signals for each traffic flow.

2 Outputs:

- **TF_1, TF_2, TF_3, TF_4**: Traffic light states represented as `3-bit` signals.

3 Performance:

- **Reaction Time:** The system responds to sensor inputs within a fixed cycle.
- **Accuracy:** Ensures no conflicting **G**REEN states are active simultaneously.

4 State Machine:

- **Finite State Machine (FSM):** Controls state transitions for traffic light states.
- **Reset Handling:** The FSM handles the reset signal and ensures a smooth return to the initial state.
- **Sensor Override:** Sensor inputs can override the current state to prioritize specific traffic flows.

5 Additional Features:

- **Extended GREEN Time:** If priority sensors are active, the FSM extends the GREEN light duration for that intersection.
- **Safety:** Ensures at least one complete cycle of RED between perpendicular directions to avoid conflicts.

CHAPTER 2

FSM Design

2.1 FSM Design

Purpose:

– Design a finite state machine (FSM) to control the traffic light states.

1 States:

* **RED:** Default state where traffic flow is stopped. Represented as `TF_X = 3'b000`.
* **YELLOW:** Transition state before **RED** or **GREEN**. Represented as `TF_X = 3'b001`.
* **GREEN:** Active traffic flow state. Represented as `TF_X = 3'b010`.

2 Transitions:

* **Default Cycle: RED → YELLOW → GREEN → YELLOW → RED**
* **Sensor-Based Priority:** If a sensor (e.g., `s_tfX`) is active, the FSM transitions to the **GREEN** state for the corresponding traffic light.

3 Inputs:

* **clk:** Clock signal for synchronization.
* **rst:** Reset signal to initialize or reinitialize the system.
* **s_tf1, s_tf2, s_tf3, s_tf4:** Sensor inputs for each intersection, which influence the state transitions.

4 Outputs:

* **TF_1, TF_2, TF_3, TF_4:** Traffic light state signals, each represented as a `3-bit` signal.

5 Conflict Handling:

* When two perpendicular sensors request **GREEN** simultaneously, the FSM gives priority based on predefined rules.
* **Priority Order:** `TF_1 > TF_2 > TF_3 > TF_4`

6 State Diagram:

* The state diagram visually represents all possible states and transitions.
* The diagram helps in understanding, debugging, and verifying the system design.

2.2 FSM diagrams

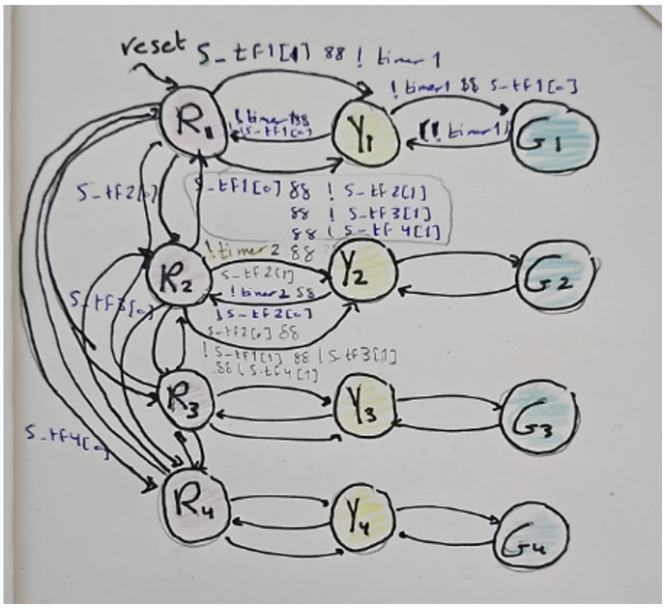
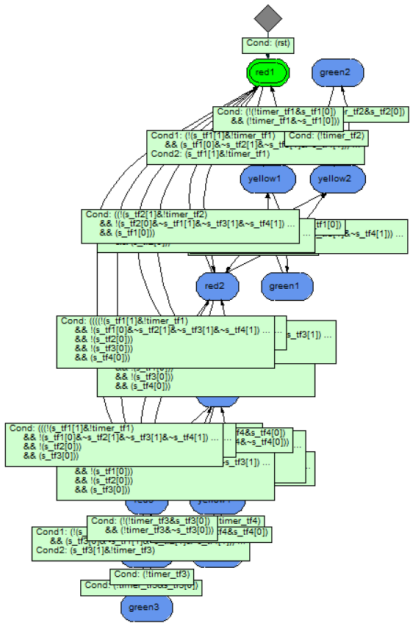


Figure 2.1: The FSM handwritten diagram

States: RED (Default State), YELLOW (Transition State), GREEN (Active Traffic Flow)

(a) The FSM debugging diagram



(b) The FSM debugging diagram with conditions

Figure 2.2: FSM debugging diagrams

CHAPTER 3

RTL Code

3.1 Overview

The RTL (Register Transfer Level) code implements the control logic for four traffic lights. The traffic lights operate according to a state machine that transitions through the following states: RED, YELLOW, and GREEN.

3.2 Code

```
1  module traffic_light (
2
3  input  wire       clk ,rst ,
4  input  wire [1:0] s_tf1 , s_tf2 , s_tf3 , s_tf4 ,
5  output reg  [2:0] TF_1 , TF_2 , TF_3 , TF_4
6
7  );
8
9  typedef enum {
10      red1,
11      yellow1,
12      green1,
13      red2,
14      yellow2,
15      green2,
16      red3,
17      yellow3,
18      green3,
19      red4,
20      yellow4,
21      green4
22
23 } state_;
24
25 state_ current_state , next_state;
26
27 //parameters decelration
28 parameter yellow_time         = 3 ;     // yellow time is small as
       usual
29 parameter default_green_time  = 7 ;     // if s1 only is on
```

```
30 parameter extended_green_time = 10 ;    // if s2 is on
31
32 integer       timer_tf1 , timer_tf2 , timer_tf3 , timer_tf4 ;
33
34 always@(posedge clk or posedge rst)
35  begin
36   if(rst)
37    begin
38     current_state<= red1 ;
39    end
40   else
41    begin
42     current_state<=next_state;
43    end
44  end
45
46
47
48 always@(*)
49  begin
50   case(current_state)
51
52    red1 :begin
53           if((s_tf1[1]) && !timer_tf1 )
54             begin
55              next_state= yellow1;
56             end
57            else if(s_tf1[0] && !s_tf2[1] && !s_tf3[1] && !s_tf4[1])
58             begin
59              next_state= yellow1;
60             end
61            else if(s_tf2[0])
62             begin
63              next_state= red2;
64             end
65            else if(s_tf3[0])
66             begin
67              next_state= red3;
68             end
69            else if(s_tf4[0])
70             begin
71              next_state= red4;
72             end
73            else
74              begin
75               next_state= red1;
76              end
77             end
78
79    yellow1:begin
80            if(!timer_tf1 && s_tf1[0]==1 )
81             begin
82              next_state= green1;
83             end
84            else if(!timer_tf1 && s_tf1[0]==0)
85             begin
86              next_state= red1;
87             end
```

```
88            else
89              begin
90               next_state= yellow1;
91              end
92             end
93
94    green1 :begin
95             if(!timer_tf1)
96              begin
97               next_state= yellow1 ;
98              end
99             else
100             begin
101              next_state= green1;
102             end
103           end
104   red2 :begin
105          if((s_tf2[1]) && !timer_tf2 )
106            begin
107             next_state= yellow2;
108            end
109           else if(s_tf2[0] && !s_tf1[1] && !s_tf3[1] && !s_tf4[1])
110            begin
111             next_state= yellow2;
112            end
113           else if(s_tf1[0])
114            begin
115             next_state= red1;
116            end
117           else if(s_tf3[0])
118            begin
119             next_state= red3;
120            end
121           else if(s_tf4[0])
122            begin
123             next_state= red4;
124            end
125           else
126             begin
127              next_state= red2;
128             end
129            end
130
131  yellow2:begin
132           if(!timer_tf2 && s_tf2[0]==1 )
133            begin
134             next_state= green2;
135            end
136           else if(!timer_tf2 && s_tf2[0]==0)
137            begin
138             next_state= red2;
139            end
140           else
141             begin
142              next_state= yellow2;
143             end
144            end
145   green2 :begin
```

```
146             if(!timer_tf2)
147              begin
148               next_state= yellow2 ;
149              end
150             else
151              begin
152               next_state= green2;
153              end
154            end
155    red3 :begin
156           if((s_tf3[1]) && !timer_tf3 )
157             begin
158              next_state= yellow3;
159             end
160            else if(s_tf3[0] && !s_tf1[1] && !s_tf2[1] && !s_tf4[1])
161             begin
162              next_state= yellow3;
163             end
164            else if(s_tf1[0])
165             begin
166              next_state= red1;
167             end
168            else if(s_tf2[0])
169             begin
170              next_state= red2;
171             end
172            else if(s_tf4[0])
173             begin
174              next_state= red4;
175             end
176            else
177              begin
178               next_state= red3;
179              end
180           end
181
182     yellow3:begin
183            if(!timer_tf3 && s_tf3[0]==1 )
184             begin
185              next_state= green3;
186             end
187            else if(!timer_tf3 && s_tf3[0]==0)
188             begin
189              next_state= red3;
190             end
191            else
192              begin
193               next_state= yellow3;
194              end
195             end
196
197    green3 :begin
198             if(!timer_tf3)
199              begin
200               next_state= yellow3 ;
201              end
202             else
203              begin
```

```
204             next_state= green3;
205            end
206          end
207
208   red4 :begin
209         if((s_tf4[1]) && !timer_tf4 )
210           begin
211            next_state= yellow4;
212           end
213          else if(s_tf4[0] && !s_tf1[1] && !s_tf2[1] && !s_tf3[1])
214           begin
215            next_state= yellow4;
216           end
217          else if(s_tf1[0])
218           begin
219            next_state= red1;
220           end
221          else if(s_tf2[0])
222           begin
223            next_state= red2;
224           end
225          else if(s_tf3[0])
226           begin
227            next_state= red3;
228           end
229          else
230            begin
231             next_state= red4;
232            end
233         end
234    yellow4:begin
235             if(!timer_tf4 && s_tf4[0]==1 )
236           begin
237            next_state= green4;
238           end
239          else if(!timer_tf4 && s_tf4[0]==0)
240           begin
241            next_state= red4;
242           end
243          else
244            begin
245             next_state= yellow4;
246            end
247           end
248
249   green4 :begin
250            if(!timer_tf4)
251             begin
252              next_state= yellow4 ;
253             end
254            else
255             begin
256              next_state= green4 ;
257             end
258           end
259
260
261   default : begin
```

```
262                next_state= red1;
263               end
264     endcase
265   end
266
267 always@(*)
268   begin
269     TF_1 = 3'b001;
270     TF_2 = 3'b001;
271     TF_3 = 3'b001;
272     TF_4 = 3'b001;
273
274     case(current_state)
275
276
277     red1 :begin
278             TF_1 = 3'b001 ;
279            end
280
281     yellow1:begin
282              if ((timer_tf1 == 0) && (s_tf1[0]==1))
283                                  begin
284                                      TF_1 = 3'b100;
285                                  end
286                             else if ((timer_tf1 == 0) && (s_tf1[0]==0))
287                                  begin
288                                      TF_1 = 3'b001;
289                                  end
290                             else
291                                  begin
292                                      TF_1 = 3'b010 ;
293                                  end
294               end
295
296     green1 :begin
297               TF_1 = 3'b100;
298              end
299     red2 :begin
300             TF_2 = 3'b001 ;
301            end
302
303    yellow2:begin
304              if ((timer_tf2 == 0) && (s_tf2[0]==1))
305                                  begin
306                                      TF_2 = 3'b100;
307                                  end
308                             else if ((timer_tf2 == 0) && (s_tf2[0]==0))
309                                  begin
310                                      TF_2 = 3'b001;
311                                  end
312                             else
313                                  begin
314                                      TF_2 = 3'b010 ;
315                                  end
316               end
317     green2 :begin
318               TF_2 = 3'b100;
319              end
```

```
320     red3 :begin
321             TF_3 = 3'b001 ;
322            end
323
324      yellow3:begin
325                if ((timer_tf3 == 0) && (s_tf3[0]==1))
326                                begin
327                                    TF_3 = 3'b100;
328                                end
329                            else if ((timer_tf3 == 0) && (s_tf3[0]==0))
330                                begin
331                                    TF_3 = 3'b001;
332                                end
333                            else
334                                begin
335                                    TF_3 = 3'b010 ;
336                                end
337               end
338
339     green3 :begin
340               TF_3 = 3'b100;
341              end
342
343     red4 :begin
344             TF_4 = 3'b001 ;
345            end
346      yellow4:begin
347                if ((timer_tf4 == 0) && (s_tf4[0]==1))
348                                begin
349                                    TF_4 = 3'b100;
350                                end
351                            else if ((timer_tf4 == 0) && (s_tf4[0]==0))
352                                begin
353                                    TF_4 = 3'b001;
354                                end
355                            else
356                                begin
357                                    TF_4 = 3'b010 ;
358                                end
359               end
360
361     green4 :begin
362               TF_4 = 3'b100;
363              end
364
365
366     default : begin
367                  TF_1 = 3'b001;
368                  TF_2 = 3'b001;
369                  TF_3 = 3'b001;
370                  TF_4 = 3'b001;
371               end
372     endcase
373   end
374
375
376
377 always @( posedge clk or posedge rst) begin
```

```
378
379     if (rst)
380         begin
381             timer_tf1 <= 0 ;
382             timer_tf2 <= 0 ;
383             timer_tf3 <= 0 ;
384             timer_tf4 <= 0 ;
385         end
386     else begin
387
388             if (next_state== green1 )
389                 begin
390                     if (s_tf1 [1] == 1 && s_tf1 [0] == 1)
391                          begin
392                             timer_tf1 <= (timer_tf1 == 0) ?
                                    extended_green_time : timer_tf1 - 1;
393                          end
394                     else
395                          begin
396                             timer_tf1 <= (timer_tf1 == 0) ?
                                    default_green_time : timer_tf1 - 1;
397                          end
398                 end
399             else if (next_state == yellow1 )
400                 begin
401                     timer_tf1 <= (timer_tf1 == 0) ? yellow_time :
                           timer_tf1 - 1;
402                 end
403
404             if (next_state == green2 )
405                 begin
406                     if (s_tf2 [1] == 1 && s_tf2 [0] == 1)
407                         begin
408                             timer_tf2 <= (timer_tf2 == 0) ?
                                    extended_green_time : timer_tf2 - 1;
409                         end
410                     else
411                         begin
412                             timer_tf2 <= (timer_tf2 == 0) ?
                                    default_green_time : timer_tf2 - 1;
413                         end
414                 end
415             else if (next_state == yellow2 )
416                 begin
417                    timer_tf2 <= (timer_tf2 == 0) ? yellow_time :
                          timer_tf2 - 1;
418                 end
419
420             if (next_state == green3 )
421                 begin
422                     if (s_tf3 [1] == 1 && s_tf3 [0] == 1)
423                         begin
424                             timer_tf3 <= (timer_tf3 == 0) ?
                                    extended_green_time : timer_tf3 - 1;
425                         end
426                     else
427                         begin
```

```
428                             timer_tf3 <= (timer_tf3 == 0) ?
                                   default_green_time : timer_tf3 - 1;
429                         end
430                 end
431             else if (next_state == yellow3 )
432                 begin
433                    timer_tf3 <= (timer_tf3 == 0) ? yellow_time :
                          timer_tf3 - 1;
434                 end
435
436
437             if (next_state== green4 )
438             begin
439                 if (s_tf4 [1] == 1 && s_tf4 [0] == 1)
440                     begin
441                         timer_tf4 <= (timer_tf4 == 0) ?
                              extended_green_time : timer_tf4 - 1;
442                     end
443                 else
444                     begin
445                         timer_tf4 <= (timer_tf4 == 0) ?
                              default_green_time : timer_tf4 - 1;
446                     end
447             end
448             else if (next_state == yellow4 )
449                 begin
450                    timer_tf4 <= (timer_tf4 == 0) ? yellow_time :
                          timer_tf4 - 1;
451                 end
452     end
453 end
454 endmodule
```

Listing 3.1: RTL Code for Traffic Light Controller

CHAPTER 4

Testbench

4.1 Purpose

The testbench verifies the functionality of the RTL design by simulating various test scenarios. It validates the following:

* Initial conditions (all RED state).
* State transitions.
* Sensor-driven priority.
* Conflict-free operation.

4.2 Testbench Code

```
1  module traffic_light_tb_2;
2
3  // Declare testbench signals
4  reg         clk, rst;
5  reg  [1:0] s_tf1, s_tf2, s_tf3, s_tf4;
6  wire [2:0] TF_1, TF_2, TF_3, TF_4;
7
8  // Instantiate the traffic light module
9  traffic_light uut (
10     .clk(clk),
11     .rst(rst),
12     .s_tf1(s_tf1),
13     .s_tf2(s_tf2),
14     .s_tf3(s_tf3),
15     .s_tf4(s_tf4),
16     .TF_1(TF_1),
17     .TF_2(TF_2),
18     .TF_3(TF_3),
19     .TF_4(TF_4)
20 );
21
22 ////////////////////////////////////////////////////////////////////
   Clock generation (10-unit period)
   ////////////////////////////////////////////
23 always begin
```

```
24      #5 clk = ~clk;
25  end
26
27  // Initialize the signals
28  initial begin
29      // Initialize signals
30      clk = 0;
31      rst = 0;
32      s_tf1 = 2'b00;
33      s_tf2 = 2'b00;
34      s_tf3 = 2'b00;
35      s_tf4 = 2'b00;
36
37      ////////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
38      #5 rst = 1;
39      #10 rst = 0;
40
41      ////////////////////////////////////////////////////////////////////
            Test 1: Initial state
            ////////////////////////////////////////////////////
42
43      $display("\nTest 1: Initial state");
44      $monitor("Time=%0t: TF_1=%b, TF_2=%b, TF_3=%b, TF_4=%b", $time,
            TF_1, TF_2, TF_3, TF_4);
45      #100;
46
47      ////////////////////////////////// Test 2: TF_1 , TF_2 , TF_3 ,
            TF_4 transitions from RED     YELLOW     GREEN     YELLOW
            RED//////////////////////
48
49      $display("\nTest 2: TF_1 transition sequence");
50      s_tf1 = 2'b01;
51      #10;
52      $display("Step 1 (TF_1 YELLOW)");
53      #30;
54      $display("Step 2 (TF_1 GREEN)");
55      #70;
56      $display("Step 3 (TF_1 YELLOW)");
57      s_tf1 = 2'b00;
58      #30;
59      $display("Step 4 (TF_1 RED)");
60
61      ////////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
62      #50 rst = 1;
63      #10 rst = 0;
64
65
66      $display("\nTest 2: TF_2 transition sequence");
67      s_tf2 = 2'b01;
68      s_tf1 = 2'b00;
69      s_tf3 = 2'b00;
70      s_tf4 = 2'b00;
71      #10;
72      $display("Step 1 (TF_2 YELLOW)");
```

```
73      #30;
74      $display("Step 2 (TF_2 GREEN)");
75      #70;
76      $display("Step 3 (TF_2 YELLOW)");
77      s_tf2 = 2'b00;
78      #30;
79      $display("Step 4 (TF_2 RED)");
80
81      ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
82      #50 rst = 1;
83      #10 rst = 0;
84
85
86      $display("\nTest 2: TF_3 transition sequence");
87      s_tf2 = 2'b00;
88      s_tf1 = 2'b00;
89      s_tf3 = 2'b01;
90      s_tf4 = 2'b00;
91      #10;
92      $display("Step 1 (TF_3 YELLOW)");
93      #30;
94      $display("Step 2 (TF_3 GREEN)");
95      #70;
96      $display("Step 3 (TF_3 YELLOW)");
97      s_tf3 = 2'b00;
98      #30;
99      $display("Step 4 (TF_3 RED)");
100
101     ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
102     #50 rst = 1;
103     #10 rst = 0;
104
105
106     $display("\nTest 2: TF_4 transition sequence");
107     s_tf2 = 2'b00;
108     s_tf1 = 2'b00;
109     s_tf3 = 2'b00;
110     s_tf4 = 2'b01;
111     #10;
112     $display("Step 1 (TF_4 YELLOW)");
113     #30;
114     $display("Step 2 (TF_4 GREEN)");
115     #70;
116     $display("Step 3 (TF_4 YELLOW)");
117     s_tf4 = 2'b00;
118     #30;
119     $display("Step 4 (TF_4 RED)");
120
121
122     ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
123     #50 rst = 1;
124     #10 rst = 0;
```

```
125
126
127
128     ////////////////////////////////////////////////// Test 3:
            transitions with extended GREEN time
            ////////////////////////////////////////////
129
130
131     $display("\nTest 3: TF_1 extended GREEN sequence");
132     s_tf2 = 2'b00;
133     s_tf1 = 2'b11;
134     s_tf3 = 2'b00;
135     s_tf4 = 2'b00;
136     #10;
137     $display("Step 1: TF_1 YELLOW");
138     #30;
139     $display("Step 2: TF_1 GREEN");
140     #100;
141     $display("Step 3: TF_1 YELLOW");
142     s_tf1 = 2'b00;
143     #30;
144     $display("Step 4: TF_1 RED");
145
146     ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
147     #50 rst = 1;
148     #10 rst = 0;
149
150
151     $display("\nTest 3: TF_2 extended GREEN sequence");
152     s_tf2 = 2'b11;
153     s_tf1 = 2'b00;
154     s_tf3 = 2'b00;
155     s_tf4 = 2'b00;
156     #10;
157     $display("Step 1: TF_2 YELLOW");
158     #30;
159     $display("Step 2: TF_2 GREEN");
160     #100;
161     $display("Step 3: TF_2 YELLOW");
162     #30;
163     $display("Step 4: TF_2 RED");
164     s_tf2 = 2'b00;
165
166     ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
167     #50 rst = 1;
168     #10 rst = 0;
169
170     $display("\nTest 3: TF_3 extended GREEN sequence");
171     s_tf2 = 2'b00;
172     s_tf1 = 2'b00;
173     s_tf3 = 2'b11;
174     s_tf4 = 2'b00;
175     #10;
176     $display("Step 1: TF_3 YELLOW");
```

```
177     #30;
178     $display("Step 2: TF_3 GREEN");
179     #100;
180     $display("Step 3: TF_3 YELLOW");
181     #30;
182     $display("Step 4: TF_3 RED");
183     s_tf3 = 2'b00;
184
185     ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
186     #50 rst = 1;
187     #10 rst = 0;
188
189
190     $display("\nTest 3: TF_4 extended GREEN sequence");
191     s_tf2 = 2'b00;
192     s_tf1 = 2'b00;
193     s_tf3 = 2'b00;
194     s_tf4 = 2'b11;
195     #10;
196     $display("Step 1: TF_4 YELLOW");
197     #30;
198     $display("Step 2: TF_4 GREEN");
199     #100;
200     $display("Step 3: TF_4 YELLOW");
201     #30;
202     $display("Step 4: TF_4 RED");
203     s_tf4 = 2'b00;
204
205     ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
206     #50 rst = 1;
207     #10 rst = 0;
208
209
210     ////////////////////////////////////////////////// Test 4: All
            sensors inactive
            ////////////////////////////////////////////////////////
211
212     $display("\nTest 4: All sensors inactive");
213     s_tf1 = 2'b00; s_tf2 = 2'b00; s_tf3 = 2'b00; s_tf4 = 2'b00;
214     #100;
215     $display("All lights RED : TF_1=%b, TF_2=%b, TF_3=%b, TF_4=%b",
            TF_1, TF_2, TF_3, TF_4);
216
217     ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
218     #50 rst = 1;
219     #10 rst = 0;
220
221
222     ////////////////////////////////////// Test 5: Conflicting
            priority sensors (TF_2 and TF_3 active)
            //////////////////////////////////////
223
```

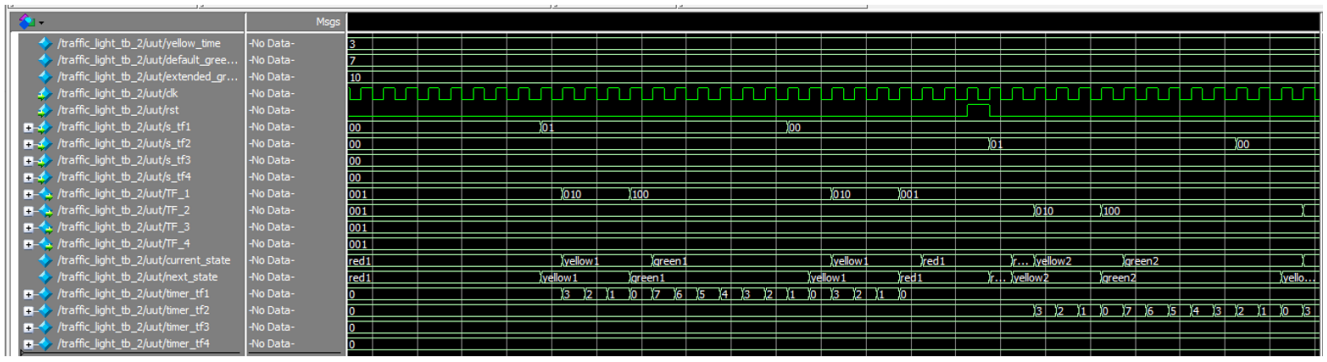
```
224     $display("\nTest 5: Conflicting priority sensors");
225     s_tf2 = 2'b01;
226     s_tf3 = 2'b11;
227     #30;
228     $display("Step 1: TF_3 YELLOW");
229     #100;
230     $display("Step 2: TF_3 GREEN");
231     s_tf3 = 2'b00;
232     #30;
233     $display("Step 3: TF_3 YELLOW");
234     #30;
235     $display("Step 4: TF_2 YELLOW");
236     #100;
237     $display("Step 5: TF_2 GREEN");
238     s_tf2 = 2'b00;
239     #30;
240     $display("Step 6: TF_2 YELLOW");
241     #100;
242
243     ////////////////////////////////////////////////////////////////
            Apply reset
            ////////////////////////////////////////////////////////////////
244     #50 rst = 1;
245     #10 rst = 0;
246
247
248     ////////////////////////////////////// Test 6: Conflicting
            priority sensors (TF_1 and TF_4 active)
            //////////////////////////////////////
249
250     $display("\nTest 6: Conflicting priority sensors");
251     s_tf1 = 2'b01;
252     s_tf4 = 2'b11;
253     #30;
254     $display("Step 1: TF_1 YELLOW");
255     #100;
256     $display("Step 2: TF_1 GREEN");
257     s_tf4 = 2'b00;
258     #30;
259     $display("Step 3: TF_1 YELLOW");
260     #30;
261     $display("Step 4: TF_4 YELLOW");
262     #100;
263     $display("Step 5: TF_4 GREEN");
264     s_tf1 = 2'b00;
265     #30;
266     $display("Step 6: TF_4 YELLOW");
267     #100
268
269     $stop;
270 end
271
272 endmodule
```

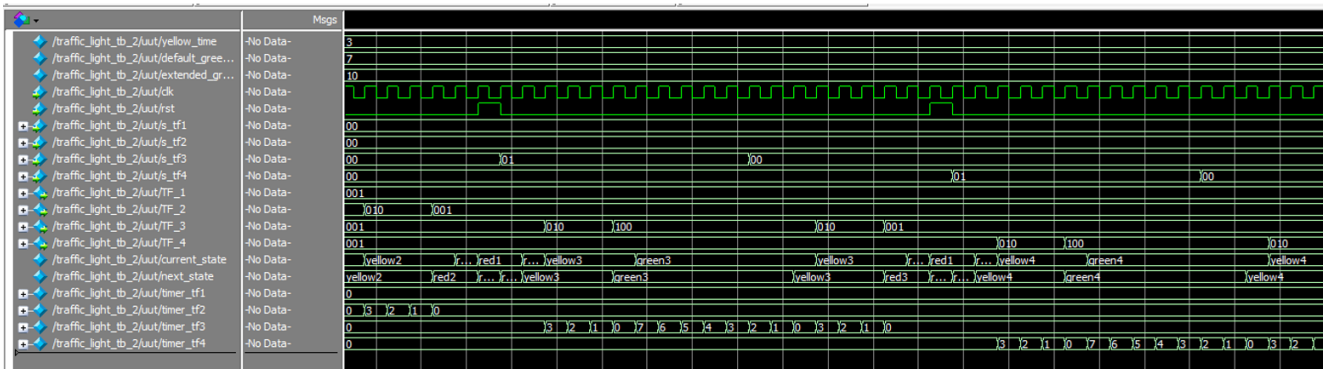
Listing 4.1: Testbench Code for Traffic Light Controller

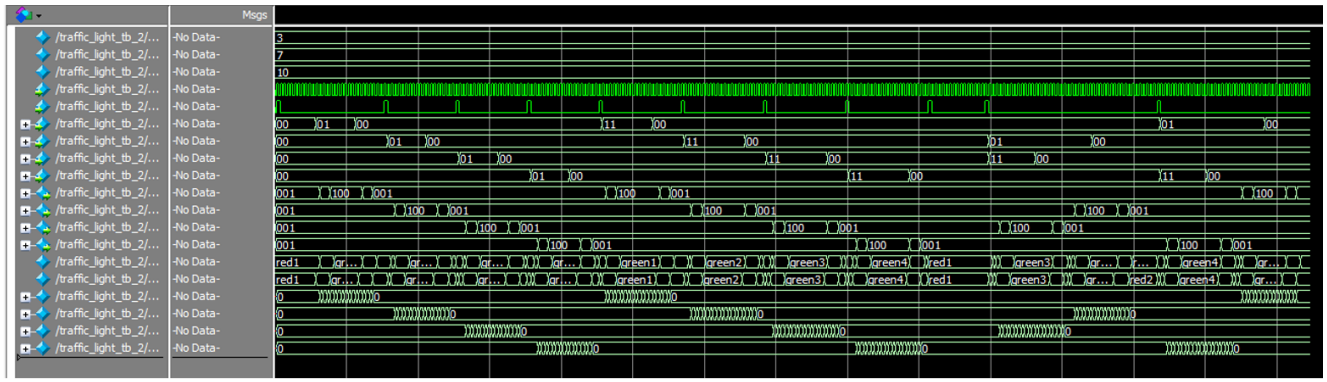
CHAPTER 5

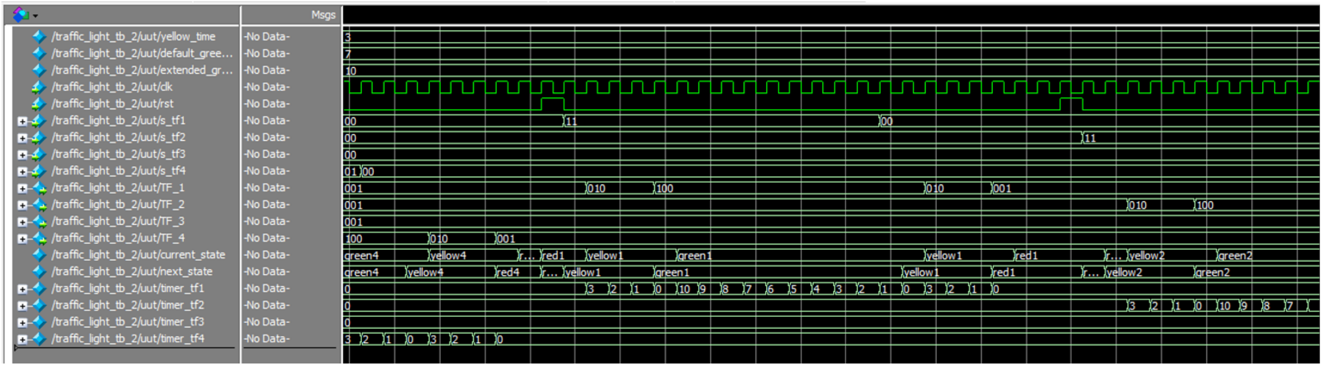
Simulation

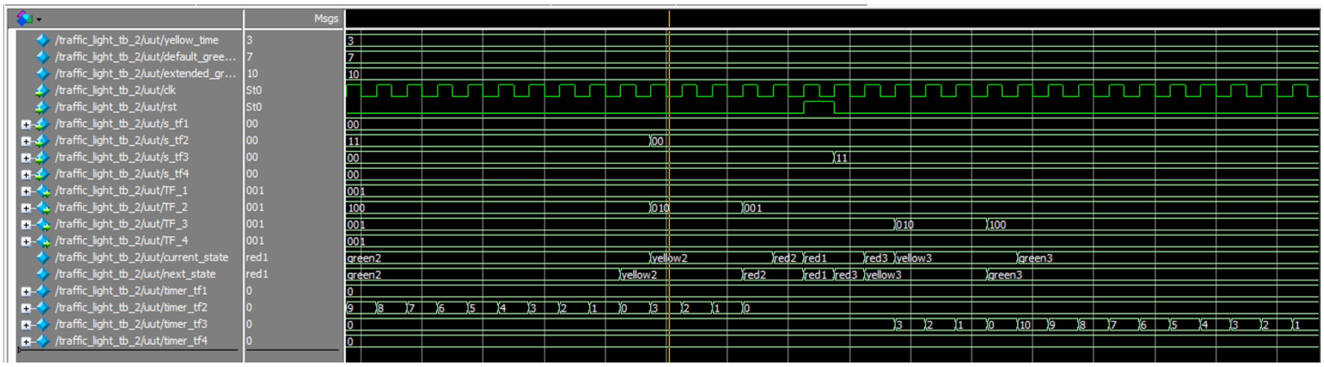
5.1 Waveform Analysis

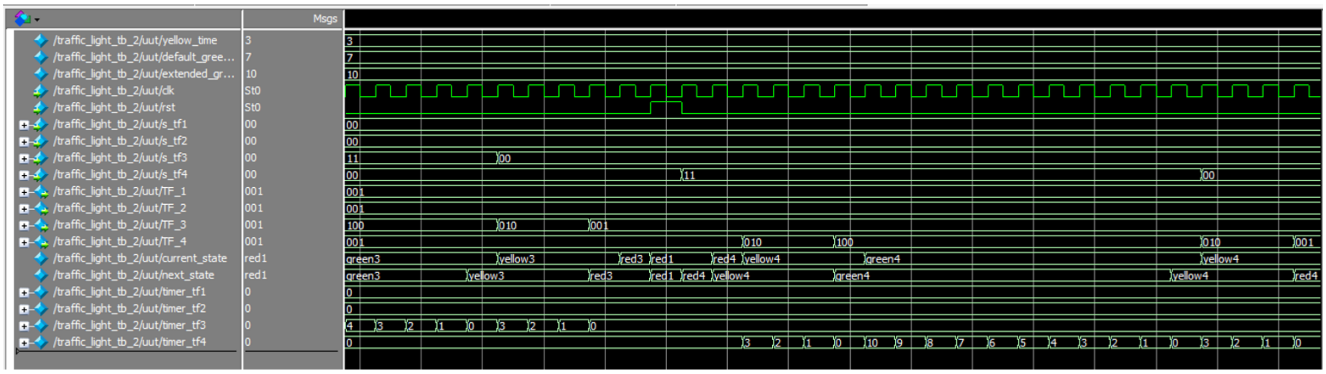


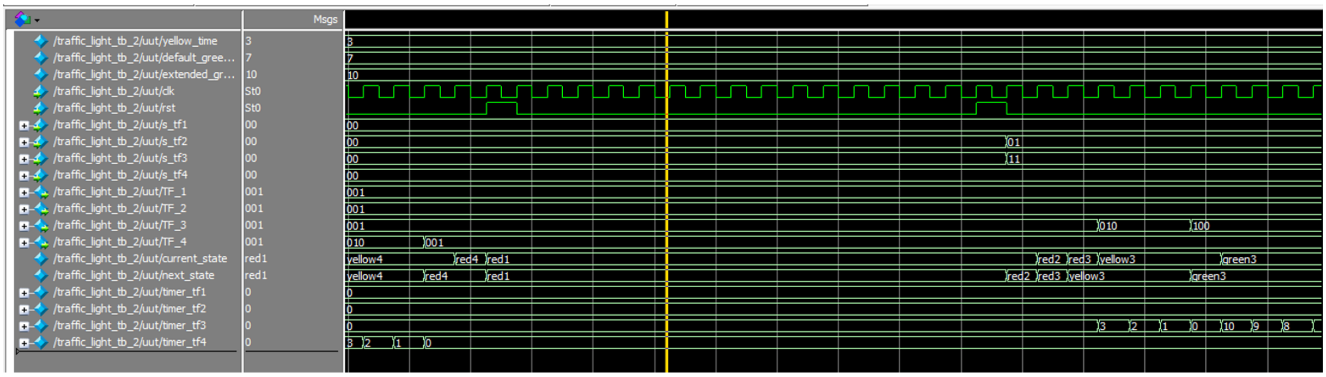


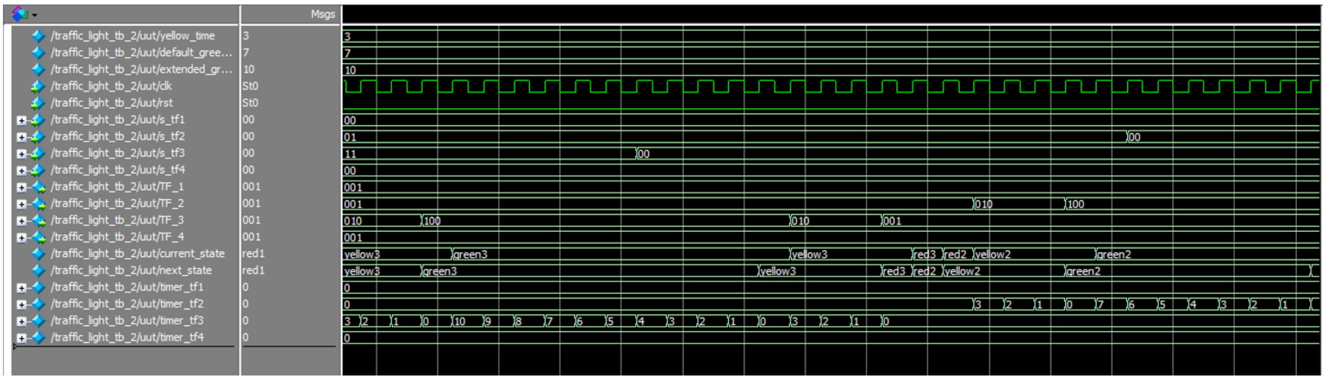


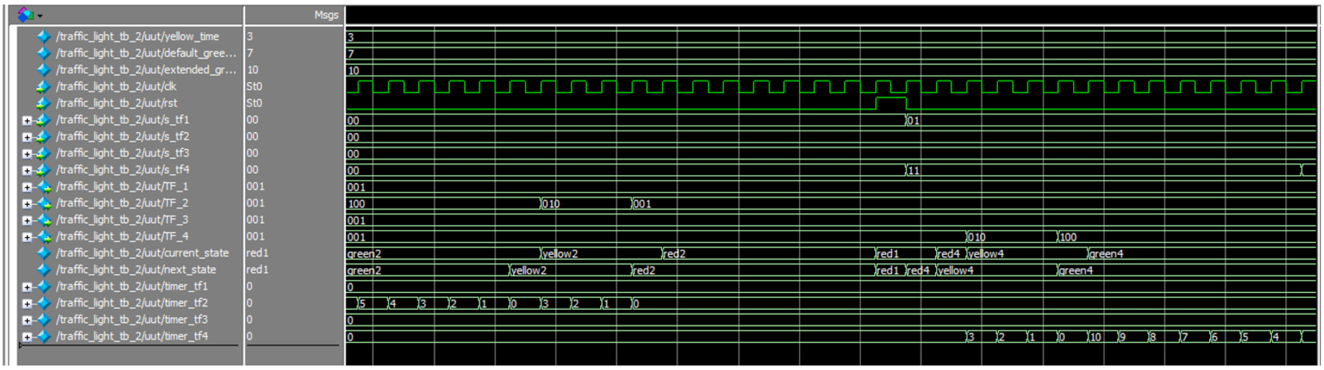

Msgs
/traffic_light_tb_2/uut/yellow_time -No Data-
/traffic_light_tb_2/uut/default_gree... -No Data-
/traffic_light_tb_2/uut/extended_gr... -No Data-
/traffic_light_tb_2/uut/clk -No Data-
/traffic_light_tb_2/uut/rst -No Data-
/traffic_light_tb_2/uut/s_tf1 -No Data-
/traffic_light_tb_2/uut/s_tf2 -No Data-
/traffic_light_tb_2/uut/s_tf3 -No Data-
/traffic_light_tb_2/uut/s_tf4 -No Data-
/traffic_light_tb_2/uut/TF_1 -No Data-
/traffic_light_tb_2/uut/TF_2 -No Data-
/traffic_light_tb_2/uut/TF_3 -No Data-
/traffic_light_tb_2/uut/TF_4 -No Data-
/traffic_light_tb_2/uut/current_state -No Data-
/traffic_light_tb_2/uut/next_state -No Data-
/traffic_light_tb_2/uut/timer_tf1 -No Data-
/traffic_light_tb_2/uut/timer_tf2 -No Data-
/traffic_light_tb_2/uut/timer_tf3 -No Data-
/traffic_light_tb_2/uut/timer_tf4 -No Data-

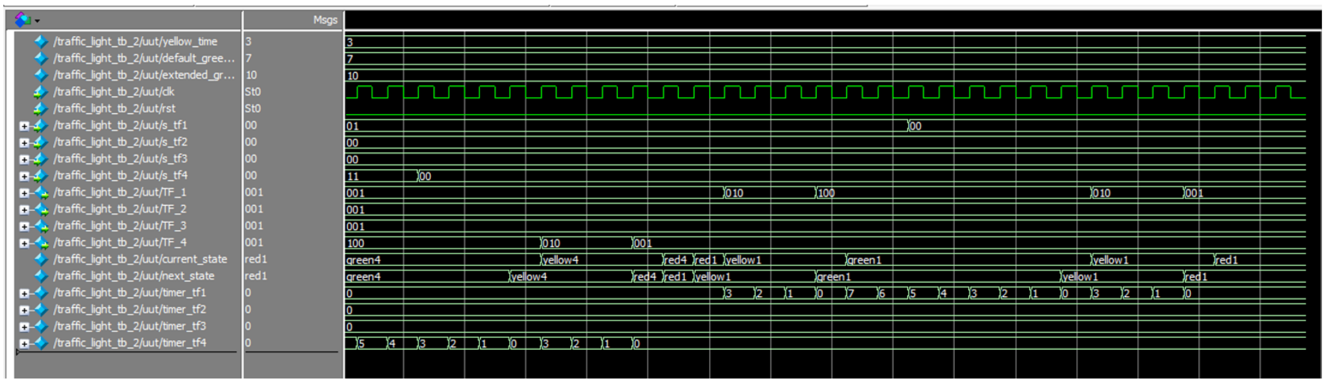

Msgs
/traffic_light_tb_2/uut/yellow_time 3
/traffic_light_tb_2/uut/default_gree... 7
/traffic_light_tb_2/uut/extended_gr... 10
/traffic_light_tb_2/uut/clk St0
/traffic_light_tb_2/uut/rst St0
/traffic_light_tb_2/uut/s_tf1 00
/traffic_light_tb_2/uut/s_tf2 00
/traffic_light_tb_2/uut/s_tf3 00
/traffic_light_tb_2/uut/s_tf4 00
/traffic_light_tb_2/uut/TF_1 001
/traffic_light_tb_2/uut/TF_2 001
/traffic_light_tb_2/uut/TF_3 001
/traffic_light_tb_2/uut/TF_4 001
/traffic_light_tb_2/uut/current_state red1
/traffic_light_tb_2/uut/next_state red1
/traffic_light_tb_2/uut/timer_tf1 0
/traffic_light_tb_2/uut/timer_tf2 0
/traffic_light_tb_2/uut/timer_tf3 0
/traffic_light_tb_2/uut/timer_tf4 0


Msgs
/traffic_light_tb_2/uut/yellow_time 3
/traffic_light_tb_2/uut/default_gree... 7
/traffic_light_tb_2/uut/extended_gr... 10
/traffic_light_tb_2/uut/clk St0
/traffic_light_tb_2/uut/rst St0
/traffic_light_tb_2/uut/s_tf1 00
/traffic_light_tb_2/uut/s_tf2 00
/traffic_light_tb_2/uut/s_tf3 00
/traffic_light_tb_2/uut/s_tf4 00
/traffic_light_tb_2/uut/TF_1 001
/traffic_light_tb_2/uut/TF_2 001
/traffic_light_tb_2/uut/TF_3 001
/traffic_light_tb_2/uut/TF_4 001
/traffic_light_tb_2/uut/current_state red1
/traffic_light_tb_2/uut/next_state red1
/traffic_light_tb_2/uut/timer_tf1 0
/traffic_light_tb_2/uut/timer_tf2 0
/traffic_light_tb_2/uut/timer_tf3 0
/traffic_light_tb_2/uut/timer_tf4 0


Msgs
/traffic_light_tb_2/uut/yellow_time 3
/traffic_light_tb_2/uut/default_gree... 7
/traffic_light_tb_2/uut/extended_gr... 10
/traffic_light_tb_2/uut/clk St0
/traffic_light_tb_2/uut/rst St0
/traffic_light_tb_2/uut/s_tf1 00
/traffic_light_tb_2/uut/s_tf2 00
/traffic_light_tb_2/uut/s_tf3 00
/traffic_light_tb_2/uut/s_tf4 00
/traffic_light_tb_2/uut/TF_1 001
/traffic_light_tb_2/uut/TF_2 001
/traffic_light_tb_2/uut/TF_3 001
/traffic_light_tb_2/uut/TF_4 001
/traffic_light_tb_2/uut/current_state red1
/traffic_light_tb_2/uut/next_state red1
/traffic_light_tb_2/uut/timer_tf1 0
/traffic_light_tb_2/uut/timer_tf2 0
/traffic_light_tb_2/uut/timer_tf3 0
/traffic_light_tb_2/uut/timer_tf4 0


Msgs
/traffic_light_tb_2/uut/yellow_time 3
/traffic_light_tb_2/uut/default_gree... 7
/traffic_light_tb_2/uut/extended_gr... 10
/traffic_light_tb_2/uut/clk St0
/traffic_light_tb_2/uut/rst St0
/traffic_light_tb_2/uut/s_tf1 00
/traffic_light_tb_2/uut/s_tf2 00
/traffic_light_tb_2/uut/s_tf3 00
/traffic_light_tb_2/uut/s_tf4 00
/traffic_light_tb_2/uut/TF_1 001
/traffic_light_tb_2/uut/TF_2 001
/traffic_light_tb_2/uut/TF_3 001
/traffic_light_tb_2/uut/TF_4 001
/traffic_light_tb_2/uut/current_state red1
/traffic_light_tb_2/uut/next_state red1
/traffic_light_tb_2/uut/timer_tf1 0
/traffic_light_tb_2/uut/timer_tf2 0
/traffic_light_tb_2/uut/timer_tf3 0
/traffic_light_tb_2/uut/timer_tf4 0


Msgs
/traffic_light_tb_2/uut/yellow_time 3
/traffic_light_tb_2/uut/default_gree... 7
/traffic_light_tb_2/uut/extended_gr... 10
/traffic_light_tb_2/uut/clk St0
/traffic_light_tb_2/uut/rst St0
/traffic_light_tb_2/uut/s_tf1 00
/traffic_light_tb_2/uut/s_tf2 00
/traffic_light_tb_2/uut/s_tf3 00
/traffic_light_tb_2/uut/s_tf4 00
/traffic_light_tb_2/uut/TF_1 001
/traffic_light_tb_2/uut/TF_2 001
/traffic_light_tb_2/uut/TF_3 001
/traffic_light_tb_2/uut/TF_4 001
/traffic_light_tb_2/uut/current_state red1
/traffic_light_tb_2/uut/next_state red1
/traffic_light_tb_2/uut/timer_tf1 0
/traffic_light_tb_2/uut/timer_tf2 0
/traffic_light_tb_2/uut/timer_tf3 0
/traffic_light_tb_2/uut/timer_tf4 0


Msgs
/traffic_light_tb_2/uut/yellow_time 3
/traffic_light_tb_2/uut/default_gree... 7
/traffic_light_tb_2/uut/extended_gr... 10
/traffic_light_tb_2/uut/clk St0
/traffic_light_tb_2/uut/rst St0
/traffic_light_tb_2/uut/s_tf1 00
/traffic_light_tb_2/uut/s_tf2 00
/traffic_light_tb_2/uut/s_tf3 00
/traffic_light_tb_2/uut/s_tf4 00
/traffic_light_tb_2/uut/TF_1 001
/traffic_light_tb_2/uut/TF_2 001
/traffic_light_tb_2/uut/TF_3 001
/traffic_light_tb_2/uut/TF_4 001
/traffic_light_tb_2/uut/current_state red1
/traffic_light_tb_2/uut/next_state red1
/traffic_light_tb_2/uut/timer_tf1 0
/traffic_light_tb_2/uut/timer_tf2 0
/traffic_light_tb_2/uut/timer_tf3 0
/traffic_light_tb_2/uut/timer_tf4 0

5.2 Snippets results with identification for each test case on wave form and transcript

```
# Test 1: Initial state
# Time=15: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 2: TF_1 transition sequence
# Step 1 (TF_1 YELLOW)
# Time=125: TF_1=010, TF_2=001, TF_3=001, TF_4=001
# Step 2 (TF_1 GREEN)
# Time=155: TF_1=100, TF_2=001, TF_3=001, TF_4=001
# Step 3 (TF_1 YELLOW)
# Time=245: TF_1=010, TF_2=001, TF_3=001, TF_4=001
# Step 4 (TF_1 RED)
# Time=275: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 2: TF_2 transition sequence
# Step 1 (TF_2 YELLOW)
# Time=335: TF_1=001, TF_2=010, TF_3=001, TF_4=001
# Step 2 (TF_2 GREEN)
# Time=365: TF_1=001, TF_2=100, TF_3=001, TF_4=001
# Step 3 (TF_2 YELLOW)
# Step 4 (TF_2 RED)
# Time=455: TF_1=001, TF_2=010, TF_3=001, TF_4=001
# Time=485: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 2: TF_3 transition sequence
# Step 1 (TF_3 YELLOW)
# Time=535: TF_1=001, TF_2=001, TF_3=010, TF_4=001
# Step 2 (TF_3 GREEN)
# Time=565: TF_1=001, TF_2=001, TF_3=100, TF_4=001
# Step 3 (TF_3 YELLOW)
# Step 4 (TF_3 RED)
# Time=655: TF_1=001, TF_2=001, TF_3=010, TF_4=001
# Time=685: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 2: TF_4 transition sequence
# Step 1 (TF_4 YELLOW)
# Time=735: TF_1=001, TF_2=001, TF_3=001, TF_4=010
# Step 2 (TF_4 GREEN)
# Time=765: TF_1=001, TF_2=001, TF_3=001, TF_4=100
# Step 3 (TF_4 YELLOW)
# Step 4 (TF_4 RED)
# Time=855: TF_1=001, TF_2=001, TF_3=001, TF_4=010
```

```
# Time=855: TF_1=001, TF_2=001, TF_3=001, TF_4=010
# Time=885: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 3: TF_1 extended GREEN sequence
# Step 1: TF_1 YELLOW
# Time=925: TF_1=010, TF_2=001, TF_3=001, TF_4=001
# Step 2: TF_1 GREEN
# Time=955: TF_1=100, TF_2=001, TF_3=001, TF_4=001
# Step 3: TF_1 YELLOW
# Time=1075: TF_1=010, TF_2=001, TF_3=001, TF_4=001
# Step 4: TF_1 RED
# Time=1105: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 3: TF_2 extended GREEN sequence
# Step 1: TF_2 YELLOW
# Time=1165: TF_1=001, TF_2=010, TF_3=001, TF_4=001
# Step 2: TF_2 GREEN
# Time=1195: TF_1=001, TF_2=100, TF_3=001, TF_4=001
# Step 3: TF_2 YELLOW
# Step 4: TF_2 RED
# Time=1315: TF_1=001, TF_2=010, TF_3=001, TF_4=001
# Time=1345: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 3: TF_3 extended GREEN sequence
# Step 1: TF_3 YELLOW
# Time=1395: TF_1=001, TF_2=001, TF_3=010, TF_4=001
# Step 2: TF_3 GREEN
# Time=1425: TF_1=001, TF_2=001, TF_3=100, TF_4=001
# Step 3: TF_3 YELLOW
# Step 4: TF_3 RED
# Time=1545: TF_1=001, TF_2=001, TF_3=010, TF_4=001
# Time=1575: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 3: TF_4 extended GREEN sequence
# Step 1: TF_4 YELLOW
# Time=1625: TF_1=001, TF_2=001, TF_3=001, TF_4=010
# Step 2: TF_4 GREEN
# Time=1655: TF_1=001, TF_2=001, TF_3=001, TF_4=100
# Step 3: TF_4 YELLOW
# Step 4: TF_4 RED
# Time=1775: TF_1=001, TF_2=001, TF_3=001, TF_4=010
```

```
# Time=1775: TF_1=001, TF_2=001, TF_3=001, TF_4=010
# Time=1805: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 4: All sensors inactive
# All lights RED : TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 5: Conflicting priority sensors
# Step 1: TF_3 YELLOW
# Time=2025: TF_1=001, TF_2=001, TF_3=010, TF_4=001
# Time=2055: TF_1=001, TF_2=001, TF_3=100, TF_4=001
# Step 2: TF_3 GREEN
# Step 3: TF_3 YELLOW
# Time=2175: TF_1=001, TF_2=001, TF_3=010, TF_4=001
# Step 4: TF_2 YELLOW
# Time=2205: TF_1=001, TF_2=001, TF_3=001, TF_4=001
# Time=2235: TF_1=001, TF_2=010, TF_3=001, TF_4=001
# Time=2265: TF_1=001, TF_2=100, TF_3=001, TF_4=001
# Step 5: TF_2 GREEN
# Step 6: TF_2 YELLOW
# Time=2355: TF_1=001, TF_2=010, TF_3=001, TF_4=001
# Time=2385: TF_1=001, TF_2=001, TF_3=001, TF_4=001
#
# Test 6: Conflicting priority sensors
# Time=2495: TF_1=001, TF_2=001, TF_3=001, TF_4=010
# Step 1: TF_1 YELLOW
# Time=2525: TF_1=001, TF_2=001, TF_3=001, TF_4=100
# Step 2: TF_1 GREEN
# Step 3: TF_1 YELLOW
# Time=2645: TF_1=001, TF_2=001, TF_3=001, TF_4=010
# Step 4: TF_4 YELLOW
# Time=2675: TF_1=001, TF_2=001, TF_3=001, TF_4=001
# Time=2705: TF_1=010, TF_2=001, TF_3=001, TF_4=001
# Time=2735: TF_1=100, TF_2=001, TF_3=001, TF_4=001
# Step 5: TF_4 GREEN
# Step 6: TF_4 YELLOW
# Time=2825: TF_1=010, TF_2=001, TF_3=001, TF_4=001
# Time=2855: TF_1=001, TF_2=001, TF_3=001, TF_4=001
# Break in Module traffic_light_tb_2 at C:/altera/13.0spl/yaaa_tb.v line 269
```

CHAPTER 6

Conclusion

The traffic light control system successfully achieves conflict-free traffic management at four intersections. The FSM logic ensures safe and efficient flow based on sensor inputs. The RTL and testbench were verified using simulation, and the design met all functional requirements.