Lab 7: Thunderbird Turn Signal

Lina Mi

ID:@01377283

Purpose:

1) Design Finite State Machine according to the function of project and draw a state transition diagram by analyzing the situation.
2) Try to use System Verilog to implement the designed finite state machine. design a Finite State Machine to implement an adventure game, including draw state transition diagram for the two parts of game, namely Room FSM and Sword FSM.
3) Use VHDL to program to implement the Adventure Game.
4) Use ModelSim software to simulate the procedure of designed game, analyze and correct the programming problems, obtain the simulation results (simulation waves).

Content:

1) **Thunderbird tail lights control :**

There are three lights on each side that operate in sequence to indicate the direction of a turn. The Figure 1 shows the sequence of tail lights flashing:

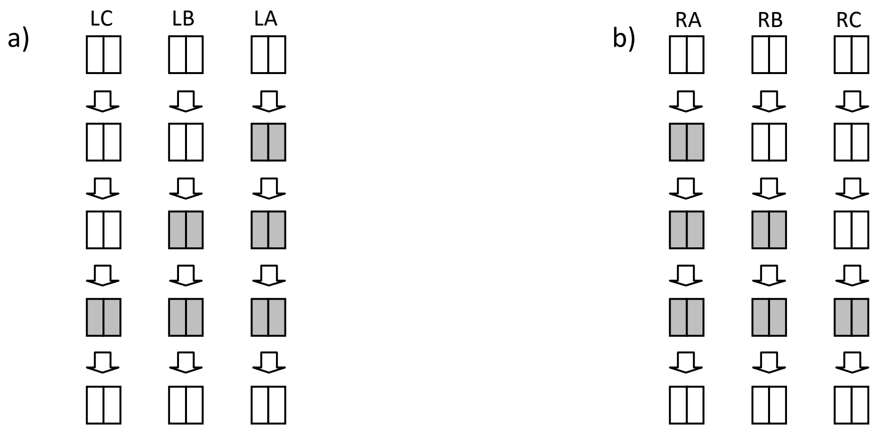


Figure 1 Flashing Sequence (LC,LB, LA are left tail lights; RC, RB, RA are right tail lights; the shaded lights are illuminated)

2) **State encode table, State transition diagrams, state transition and output tables :**

The states which the thunderbird tail lights should be in are listed in following table according to the lights condition.

Table 1 State Expression

State	Thunderbird Tail Lights					
	LC	LB	LA	RA	RB	RC
Q0	0	0	0	0	0	0
Q1	0	0	1	0	0	0
Q2	0	1	1	0	0	0
Q3	1	1	1	0	0	0
Q4	0	0	0	1	0	0
Q5	0	0	0	1	1	0
Q6	0	0	0	1	1	1

From previous description on the sequence of tail lights flashing, the diagram of Tail Light FSM is drawn as Figure 2 In the diagram, ‘Res’ is input signal ‘Reset’, ‘R’ means ‘right turn’ signal, ‘L’ represents ‘left turn’ signal

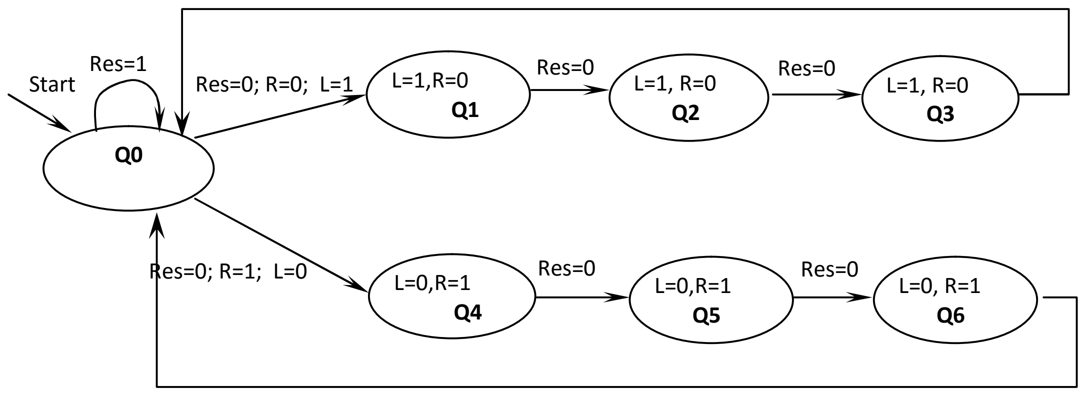


Figure 2 State Transition Diagram for Tail Light FSM

According the Figure1, the corresponding state transition table for Tail Light FSM with direction input is shown as following

Current State	CLK	Reset	Direction Inputs		Next State
			L	R	
X	X	1	X	X	Q0
Q0	1	0	0	0	Q0
Q0	1	0	1	1	Q0
Q0	1	0	1	0	Q1
Q0	1	0	0	1	Q4
Q1	1	0	X	X	Q2
Q2	1	0	X	X	Q3
Q3	1	0	X	X	Q0
Q4	1	0	X	X	Q5
Q5	1	0	X	X	Q6
Q6	1	0	X	X	Q0

3) Verilog programming

In this lab, we program with Verilog to implement the above state transition table for Thunderbird Tail Lights FSM in Altera Quartus software.

The following is the written program:

```
module Thunderbird (input logic clk,

 input logic reset,

 input logic left,right,

 output logic la,lb,lc,  ra,  rb,  rc);

enumint unsigned {f0, f1, f2, f3, f4, f5, f6} state, next_state;

always_comb

 begin



 case(state)

          f0:

                    begin

                    la= 1'b0;

                    lb= 1'b0;

                    lc= 1'b0;

                    ra= 1'b0;

                    rb= 1'b0;

                    rc= 1'b0;

                    if ((left == 1'b0 && right == 1'b0)||(left == 1'b1 &&right == 1'b1))

                    next_state<= f0;

                    else if (left==1'b1 && right==1'b0)

                    next_state<= f1;
                    else
                    next_state<= f4;
                    end
          f1:
                    begin
                    la= 1'b1;
                    lb= 1'b0;
                    lc= 1'b0;
                    ra= 1'b0;
                    rb= 1'b0;
                    rc= 1'b0;
                    next_state<= f2;
                    end
          f2:
                    begin
                    lb = 1'b1;
                    la= 1'b1;
                    lc= 1'b0;
                    ra= 1'b0;
                    rb= 1'b0;
                    rc= 1'b0;
                    next_state<= f3;
                    end
          f3:
                    begin
```

```
            lc = 1'b1;

            la= 1'b1;

            lb= 1'b1;

            ra= 1'b0;

            rb= 1'b0;

            rc= 1'b0;

            next_state<= f0;

            end

f4:

            begin

            ra = 1'b1;

            rb= 1'b0;

            rc= 1'b0;

            la= 1'b0;

            lb= 1'b0;

            lc= 1'b0;

            next_state<= f5;

            end

f5:

            begin

            rb = 1'b1;

            ra= 1'b1;

            rc= 1'b0;

            la= 1'b0;

            lb= 1'b0;
```

```
                lc= 1'b0;
                next_state<= f6;
                end
        f6:
                begin
                rc = 1'b1;
                ra= 1'b1;
                rb= 1'b1;
                la= 1'b0;
                lb= 1'b0;
                lc= 1'b0;
                next_state<= f0;
                end
        default :
                begin
                la= 1'b0;
                lb= 1'b0;
                lc= 1'b0;
                ra= 1'b0;
                rb= 1'b0;
                rc= 1'b0;
                next_state<= f0;
                end
endcase
 end
```

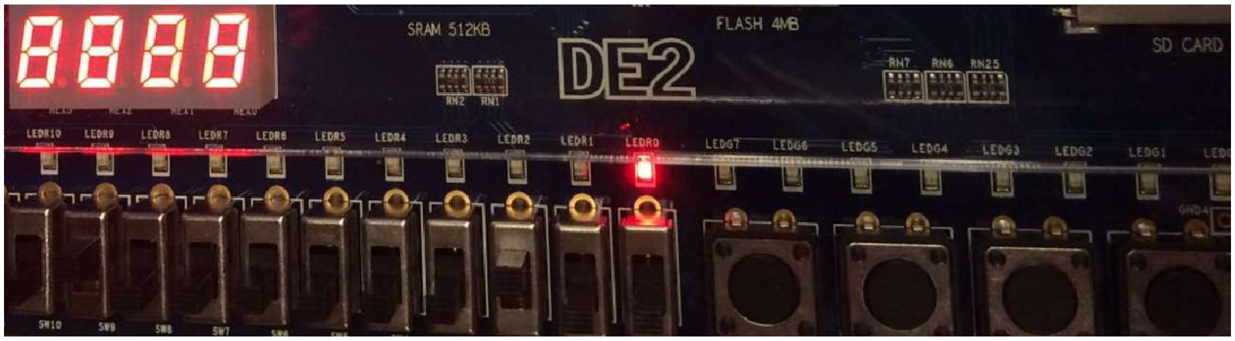
```
always_ff@(posedgeclk)

 begin

            if(reset == 1'b1)

            state <= f0;

            else

            state <= next_state;

 end

endmodule
```

4) **Implementation on DE2 board**
The designed Thunderbird Tail Lights using above Verilog program was implemented on DE2 board to verify its correctness
The input signals (reset, left turn, right turn) and output signals (LC, LB, LA and RC, RB, RC) are assigned to corresponding switches and LEDs on DE2 board, the assignment is shown in following wrapper.

```
module wrapper(input  logic [2:0] SW,

input  logic [0:0]   KEY,

                 output logic [2:0] LEDR,

                 output logic [7:5] LEDG);


  // Use Key0 for clk

  // switches for inputs

  // red and green LEDs for output


  Thunderbird Thunderbird(.clk(KEY[0]), .reset(SW[0]), .left(SW[2]), .right(SW[1]),

                .la(LEDR[0]), .lb(LEDR[1]), .lc(LEDR[2]),

.ra(LEDG[7]), .rb(LEDG[6]), .rc(LEDG[5]));
```

endmodule

The implementation results are shown in Figure 3 to 8:



· Figure 3 SW2=1(L=1), SW1=0 (R=0), SW0=0(Reset=0), left turn first state Q1 (LA is illuminated)

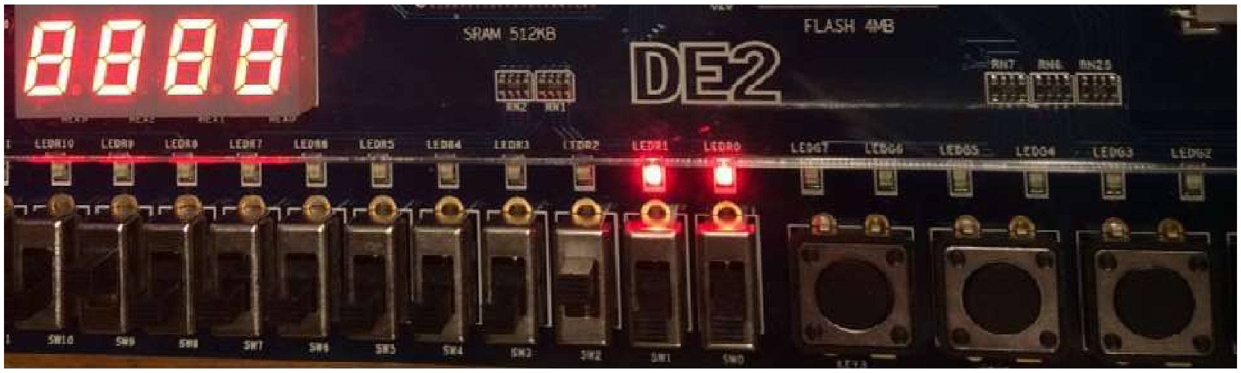


Figure 4 SW2=1(L=1), SW1=0 (R=0), SW0=0(Reset=0), left turn second state Q2 (both LB, LA are illuminated)

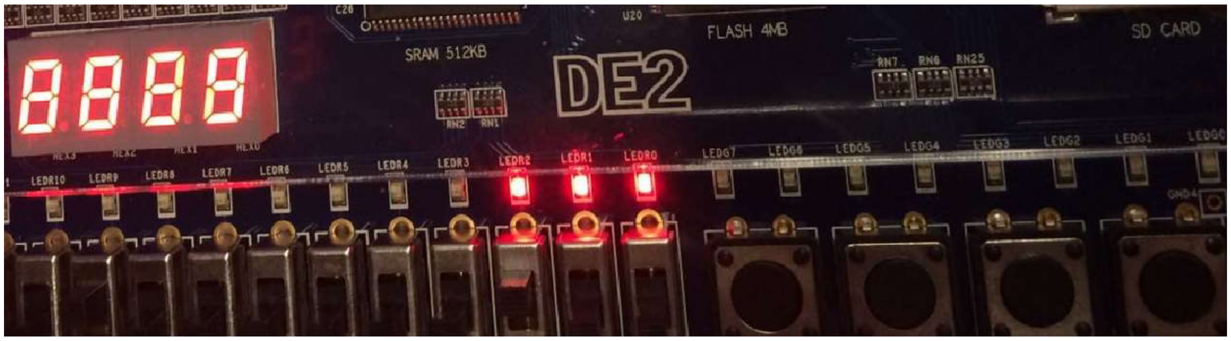


Figure 5 SW2=1(L=1), SW1=0 (R=0), SW0=0(Reset=0), left turn third state Q3 (LC, LB, LA are illuminated)

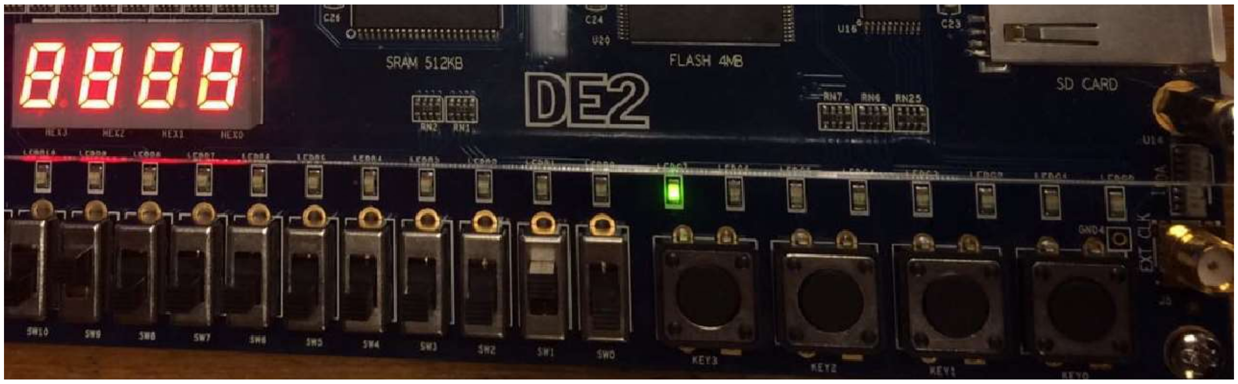


Figure 6 SW2=0(L=0), SW1=0=1 (R=1), SW0=0(Reset=0),right turn first state Q4 (RA is illuminated)

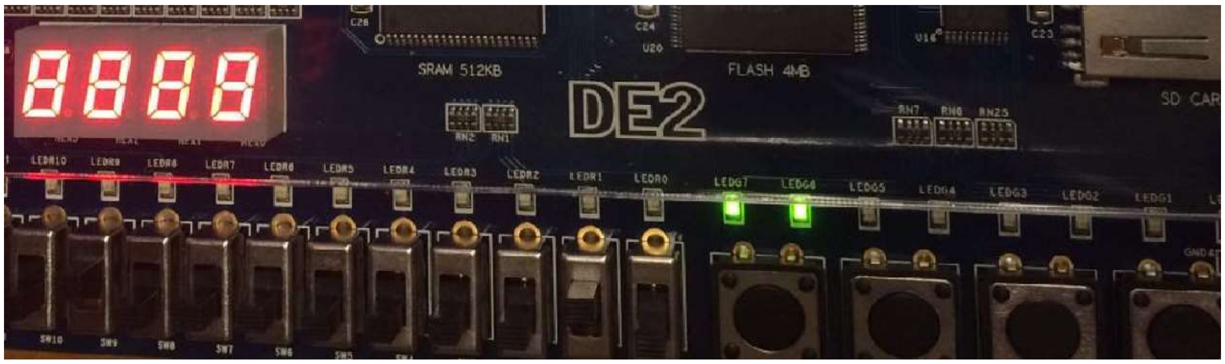


Figure 7 SW2=0(L=0), SW1=0=1 (R=1), SW0=0(Reset=0), right turn second state Q5 (RB, RA are illuminated)

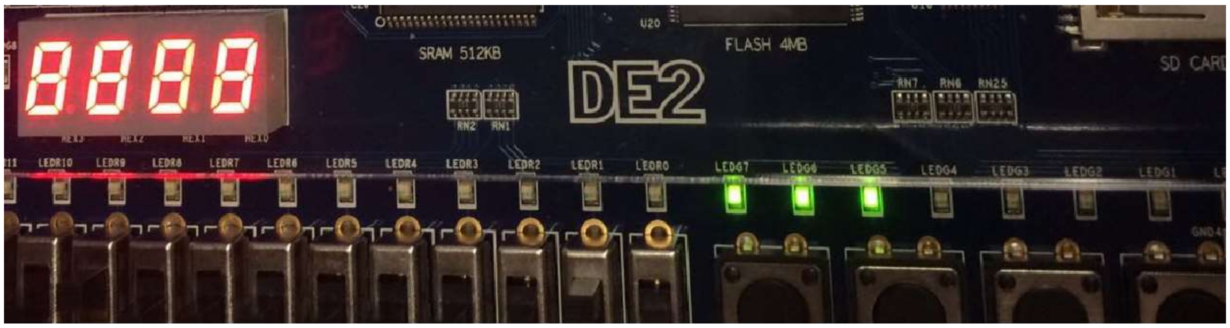


Figure 8 SW2=0(L=0), SW1=0=1 (R=1), SW0=0(Reset=0), right turn third state Q6 (RC,RB, RA are illuminated)

From Figure3 to 8, it can shown that the sequences of tail lights flashing are correctly realized when the left turn and right turn signal are input respectively by the designed FSM. So it can concluded that the design is correct.

5) Conclusions: (my lab partners and I used 2 hours to finish the Lab 7)

From the lab7,

1) Based on the description of thunderbird tail light, I am able to encode the state, draw state transition diagram for the project, and list the table of state transition under different current state and inputs;
2) I can use the Verilog to implement the FSM;
3) I can implement the designed FSM on DE2 board by assigning the input signals and output signals to the switches and LEDs of DE2 board and check the correctness of designed project by running the FSM on DE2 board.

What I did in Lab 6

1) complete the state transition diagram for thunderbird tail lights FSM;
2) List the table of state representation and table of state transition for designed FSM;
3) programming with Verilog language to implement the state transition table of Tail lights FSM;
4) Debug the program with lab partner;
5) Implement designed FSM on DE2 board with my lab partner;

What my lab partner did:

1) Complete the state transition diagram for tail light FSM and;
2) List the state transition table for tail light FSM;
3) Programming with Verilog language together with me to implement designed FSM;
4) Debug the program and compile the program successfully;
5) Implement the designed FSM on DE2 board and record the operation results;