Lab 3: Car tail lights design

Starts: starting September 24, 2017 – attend your scheduled lab session
Demo: at the end of your scheduled lab session
Points (100 points): Pre-lab (15 points) and Lab (85 points)
Code submission: Submit all your .v files on D2L under "Lab 3" by 11.59 pm on Sep 28, 2018.
Complete the overall percent effort that you and your team member contribute to the lab 3 assignment in D2L -> **Surveys (Lab 3 Work effort)** by 11.59 pm on Sep 28, 2018.
No overall percent effort completion, 20 points deduction for that team member.
References:
Lecture Notes: FSM 1 (in Unit 2) and verilog_lecture2_seq (in Unit: Verilog folder)
Objectives:

- Design and implement **circuit of the car tail lights**
- Practice writing testbench for sequential circuits
- Use Structural Verilog style to connect multiple modules to implement a digital circuit.

Pre-lab: (15 pts): Draw a **state diagram of car taillight circuit.**

Car tail lights design

Figure 1 below shows the tail lights of a car. There are 3 lights on each side and they operate in sequence to show the turning direction and emergency as shown in Figure 2

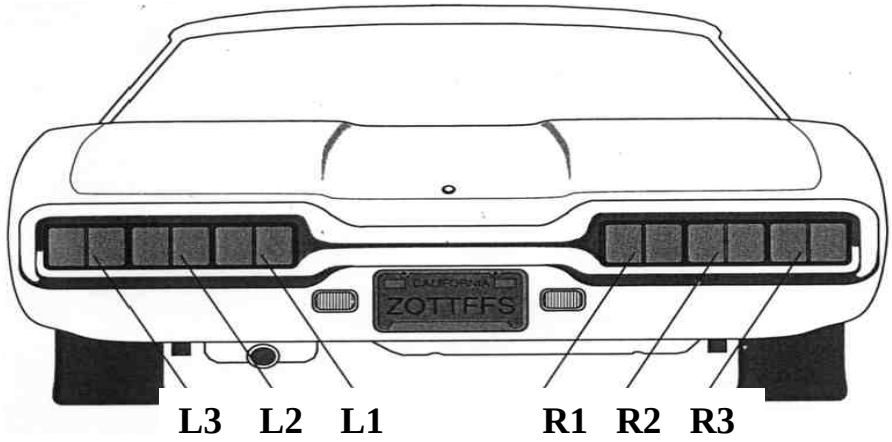
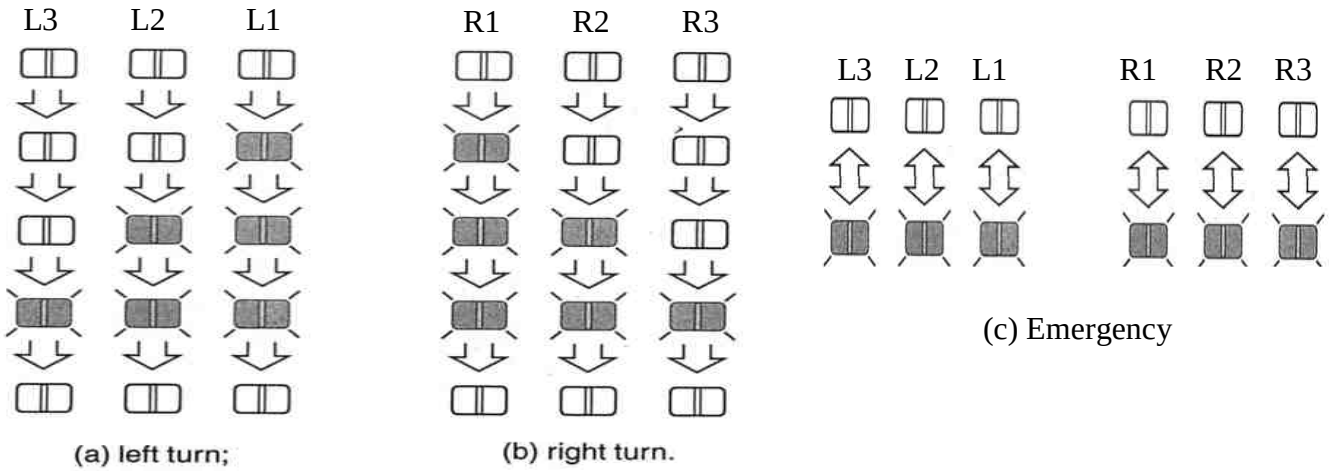


Figure 1: car tail lights

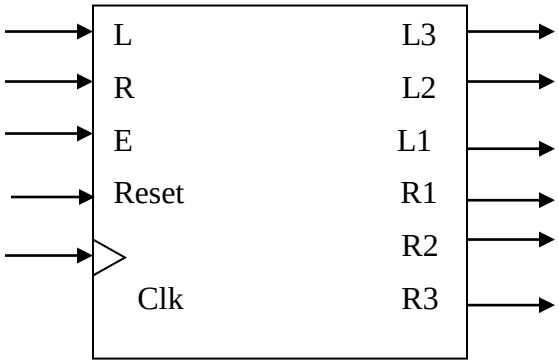


Note: Figures above are from John F. Wakerly, "Digital Design: Principles & Practices," 3rd ed., Prentice Hall, 2000.

 shows that the light is on

Figure 2: Flashing sequence for the tail lights

Prelab: You will **design a state machine** (draw its state diagram) that controls the car tail lights



Requirement Specifications:

- The FSM has 3 inputs, ***L, R,*** and ***E*** and 6 outputs ***L3, L2, L1, R1, R2,*** and ***R3***.
  - **L** is for the driver's request of a **Left turn.** The flashing sequence for a left turn is shown in Figure 2(a). **This left-turn sequence repeats as long as L = *1*** (and E and R = 0)
  - **R** is for the driver's request of a **Right turn.** The flashing sequence for a right turn is shown in Figure 2(b). **This right-turn sequence repeats as long as R = *1*** (and E and L = 0)
  - **E** is for the driver's request of an **Emergency light**. The tail lights will operate in an emergency mode – all six lights (outputs) will flash on and off in unison as shown in Figure 2(c). This emergency sequence repeats as long as E = 1 or both L = R = 1.
- The first state in your designed FSM should be an ***initial state*** where all lights are 0.

  If ***E*** is activated (=1), the lights flash in an emergency mode.

  If ***L*** is activated (=1) and E = 0, R = 0, the lights flash in the left-turn mode in which the left-turn sequence will be complete and repeated as long as L = 1 (and E = R = 0). However, if ***E*** is activated or ***both L and R*** are activated in the middle of the left-turn sequence, the lights will go to the emergency mode right away.

  If ***R*** is activated (=1) and E = L = 0, the lights flash in the right-turn mode in which the right-turn sequence will be complete and repeated as long as R = 1 (and E = L = 0). However, if ***E*** is activated or ***both L and R*** are activated in the middle of the right-turn sequence, the lights will go to the emergency mode right away.

  If ***both L*** and ***R*** are activated at the same time (L = R= 1), the lights will go into the emergency mode.

Read carefully the requirement specification above and **design this sequential circuit** (draw its state diagram).

Step 1: Write **Verilog code** for your design

Step 2: **Simulate** (Run Post-synthesis functional simulation. Note: start with Behavioral simulation) your designed FSM. Write **Testbench** to show several scenarios. Your testbench must show **at least 4 different scenarios**.
The scenarios are when

E = 0, R = 0 and L = 0 (all lights are off)
E = 0, R = 0 and L = 1 (the sequence of left turn should be completed)
E = 0, R = 0 and L = 0 (the sequence of right turn should be completed)
E = 0, R = 1 and L = 1 (the sequence of emergency should be completed)
E = 1, R = 0 and L = 0 (the sequence of emergency should be completed)
E = 1, R = 0 and L = 1 (the sequence of emergency should be completed)
E = 1, R = 1 and L = 0 (the sequence of emergency should be completed)
E = 1, R = 1 and L = 1 (the sequence of emergency should be completed)

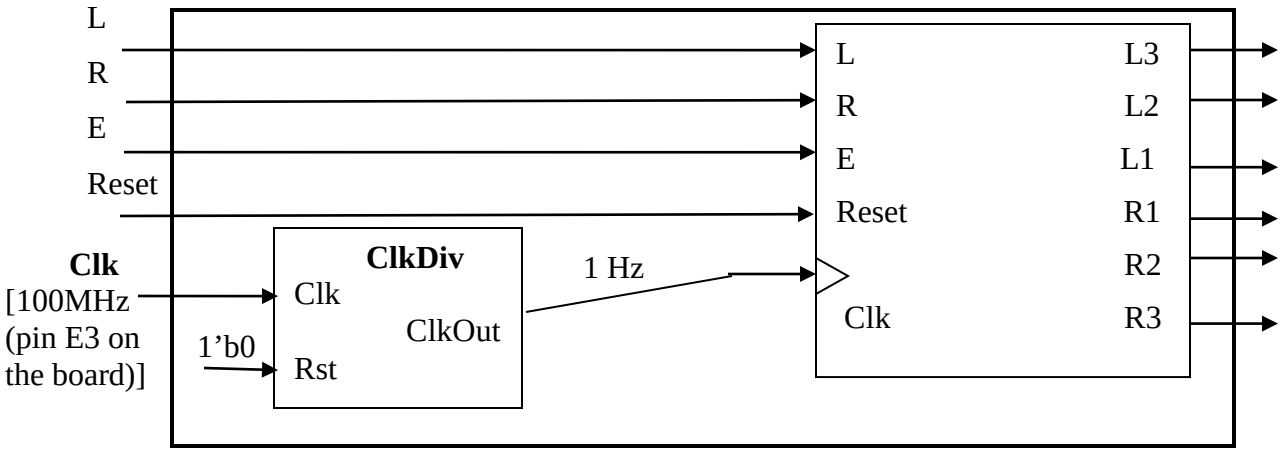
For Testbench, the code below might be useful to make input(s) stay at that value for several clock cycles

```
integer i;

#5 L <= 1'b0; R <= 1'b0; E <= 1'b1;

for (i = 0; i < 6; i = i + 1) begin
      @(posedge Clk);
end

#5 L <= 1'b0; R <= 1'b1; E <= 1'b0;
```

This is equivalent to

```
@(posedge Clk);
@(posedge Clk);
@(posedge Clk);
@(posedge Clk);
@(posedge Clk);
@(posedge Clk);
```

Step 3: Write a **top-level design code (.v file)** using the figure below for connection. Use **ClkDiv code** (see its description and code on the next page) with a frequency of **1 Hz.**



Write the .xdc file. Decide which pins you want to use for your inputs and let TA or the instructor know at the time of demo. **The following must be used for the outputs**

On the board	**LD5**	**LD4**	**LD3**	**LD2**	**LD1**	**LD0 (the rightmost LED on the board)**
Your design	**L3**	**L2**	**L1**	**R1**	**R2**	**R3**

Step 4: Download to the board. When working, demo to TA or your instructor.

ClkDiv:

two inputs: Clk and Rst, and one output: ClkOut.

Given the 100 MHz clock provided by the Nexys-4 DDR FPGA board, the ClkDiv component given below will generate a 1-Hz clock on its output, ClkOut. Below is the Verilog code for ClkDiv module (ClkDiv.v is given on D2L (the zipped folder lab 3)).

```
module ClkDiv(Clk, Rst, ClkOut);
  input Clk, Rst;
  output reg ClkOut;
 //to create 1 Hz clock from 100-MHz on the board
  parameter DivVal = 50000000;
  reg [25:0] DivCnt;
  reg ClkInt;

  always @(posedge Clk) begin
    if( Rst == 1 )begin
      DivCnt <= 0;
      ClkOut <= 0;
      ClkInt <= 0;
    end
    else begin
      if( DivCnt == DivVal ) begin
                ClkOut <= ~ClkInt;
                ClkInt <= ~ClkInt;
                DivCnt <= 0;
      end
      else begin
                ClkOut <= ClkInt;
                ClkInt <= ClkInt;
                DivCnt <= DivCnt + 1;
      end
    end
  end
endmodule
```

Points Distribution (100 pts):

Pre lab: State diagram	15 pts
Step 1: Verilog code	25 pts
Step 2: Testbench with correct Post-Synthesis functional simulation	25 pts
Step 3: Top level code and .xdc file	25 pts
Step 4: Download and the circuit works on the board	10 pts

After finishing step 4, **complete the overall percent effort** that you and your team member contribute to the lab 3 assignment in D2L -> **Surveys (Lab 3 Work effort)** by 11.59 pm on Sep 28, 2018. **No overall percent effort completion, 20 points deduction for that team member.**