Lab 5 & 6: Assignment 1 (10 Marks)

Date for Showing the output to Instructor (No Deduction): 17/09/2021 between 11AM-1PM
Due Date for final submission through CMS: 17/09/2021 9:00 PM

Name: Vishwas Vasuki Gautam
ID No: 2019A3PS0443H

Problem Statement: Design a state machine to control the tail lights of a 1965 Ford Thunder bird (Figure 1). The tail lights are composed of three lights on each side which operate for the turns as shown in figure 2. The state machine has two inputs **(LEFT, RIGHT)** and 6 outputs **(LC, LB, LA, RA, RB and RC)**. When **(RIGHT=0 and LEFT=0)** or when **(RIGHT=1 and LEFT=1)** no lights will turn ON. If **(RIGHT=0 and LEFT=1)** then lights LC, LB, and LA will be ON as shown in figure 2(a) indicating **LEFT** turn. If **(RIGHT=1 and LEFT=0)** then lights RA, RB, and RC will be ON as shown in figure 2(b) indicating **RIGHT** turn. In addition to LEFT and RIGHT there are two more inputs **Clk** and **Reset** for normal operation of FSM. When **Reset** is enabled all lights will be OFF**.** The flashing rate of LEDs is 2Hz (i.e. the time between two successive states is 0.5s).

PIN Assignment:

Inputs: **RIGHT F22; LEFT G22;**

Outputs: **LC U14; LB U19; LA W22; RA U22; RB T21; RC T22;**

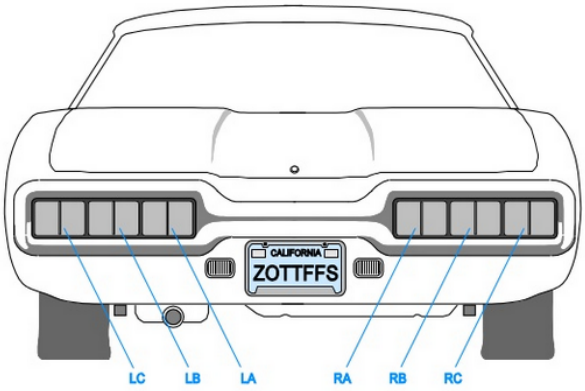


Figure 1

Left Turn **Right Turn**

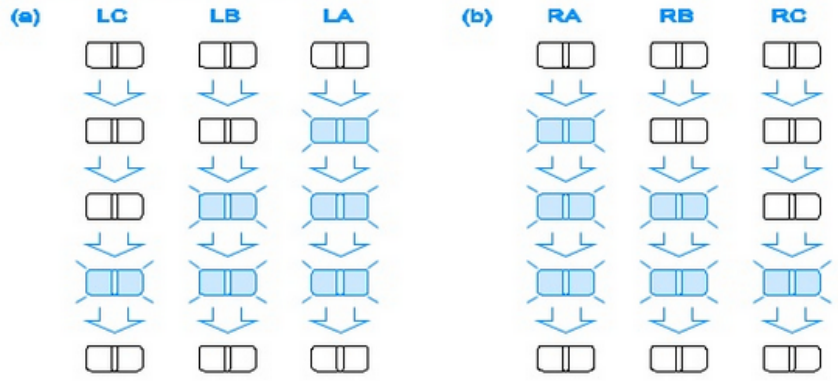


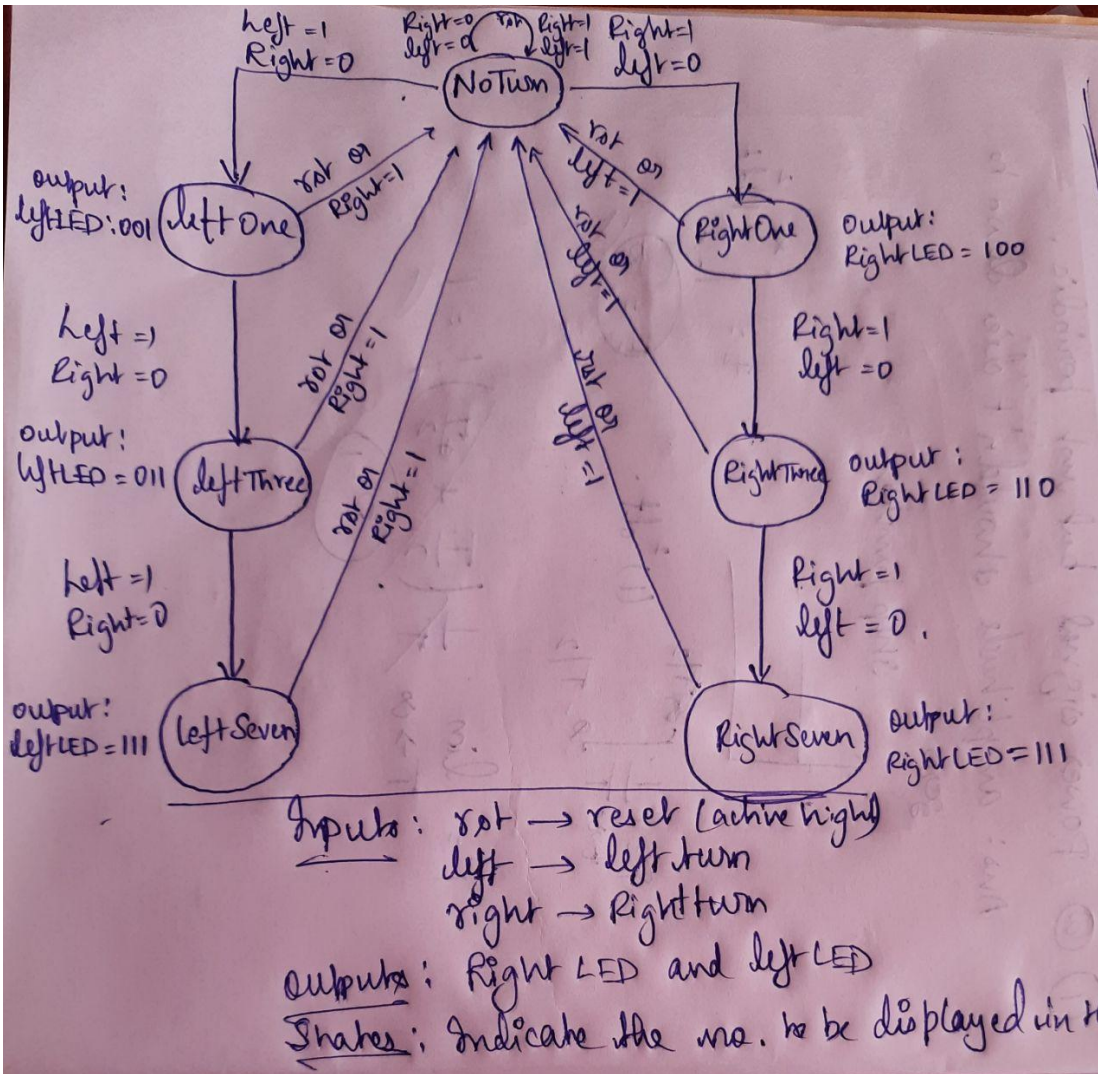
Figure 2(a) **Figure 2(b)**

Figure 2

(Please refer to the file named "**Vivado_Design_Flow_All_Steps.pdf**" for a review of all the steps in the design flow)

1. **<u>Question</u>: Draw the FSM (can be an image) with proper description.**

Answer:



2. Create a Vivado Project and write Verilog code (**Car_FSM.v** with comments) for implementing the above FSM.

<u>Question</u>: Paste the image of verilog code Car_FSM.v.

Answer:

:

Car_FSM.v

F:/CompArch/FPGA_Lab/Lab5and6/project_2/project_2.srcs/sources_1/new/Car_FSM.v

```
22  // VISHWAS VASUKI GAUTAM - 2019A3PS0443H
23  module Car_FSM(
24      input clk,
25      input rst,
26      input right,
27      input left,
28      output reg [2:0] rightLED,
29      output reg [2:0] leftLED
30      );
31
32  reg [25:0] clkDiv;
33
34  parameter noTurn = 3'b000;
35  parameter leftOne = 3'b001, leftThree = 3'b010, leftSeven = 3'b011; // Indicates the states for left turn
36  parameter rightOne = 3'b100, rightThree = 3'b101, rightSeven = 3'b110; // Indicates the states for right turn
37
38  reg [2:0] presentState, nextState;
39
40  initial begin
41      clkDiv <= 0;
42  end
43
44  always @(posedge clk)                  // Clock division
45      begin
46          clkDiv <= clkDiv + 1;
47      end
48
49  always @(posedge clkDiv[25], posedge rst)     // State register for initialisation
50      begin
51          if(rst)
52              presentState <= noTurn;
53          else
54              presentState <= nextState;
55      end
```

Car_FSM.v

F:/CompArch/FPGA_Lab/Lab5and6/project_2/project_2.srcs/sources_1/new/Car_FSM.v

```
57  always @(presentState, right, left)            // Calculating the next state based on the inputs and the present state
58      begin
59          case(presentState)
60              noTurn:
61                  if(left ^ right == 0)
62                      nextState <= noTurn;
63                  else if(left && ~right)
64                      nextState <= leftOne;
65                  else if(~left && right)
66                      nextState <= rightOne;
67
68              leftOne:
69                  nextState <= (left && ~right)? leftThree: noTurn;
70
71              leftThree:
72                  nextState <= (left && ~right)? leftSeven: noTurn;
73
74              leftSeven:
75                  nextState <= noTurn;
76
77              rightOne:
78                  nextState <= (~left && right)? rightThree: noTurn;
79
80              rightThree:
81                  nextState <= (~left && right)? rightSeven: noTurn;
82
83              rightSeven:
84                  nextState <= noTurn;
85
86          endcase
87      end
88
```

```
always @(presentState)//Calculating the output based on the present state
    begin
        case(presentState)
            noTurn: begin
                    rightLED <= 3'b000;
                    leftLED <= 3'b000;
                end
            leftOne: begin
                     rightLED <= 3'b000;
                     leftLED <= 3'b001;
                end
            leftThree: begin
                    rightLED <= 3'b000;
                    leftLED <= 3'b011;
                end
            leftSeven: begin
                    rightLED <= 3'b000;
                    leftLED <= 3'b111;
                end
            rightOne: begin
                    rightLED <= 3'b100;
                    leftLED <= 3'b000;
                end
            rightThree: begin
                    rightLED <= 3'b110;
                    leftLED <= 3'b000;
                end
            rightSeven: begin
                    rightLED <= 3'b111;
                    leftLED <= 3'b000;
                end
        endcase
    end
endmodule
```

3. Write the test bench **Test_Car_FSM.v** and simulate your design to check the functionality.

<u>Question</u>: Paste the image of test bench verilog code Test_Car_FSM.v
Answer:

Car_FSM_test.v

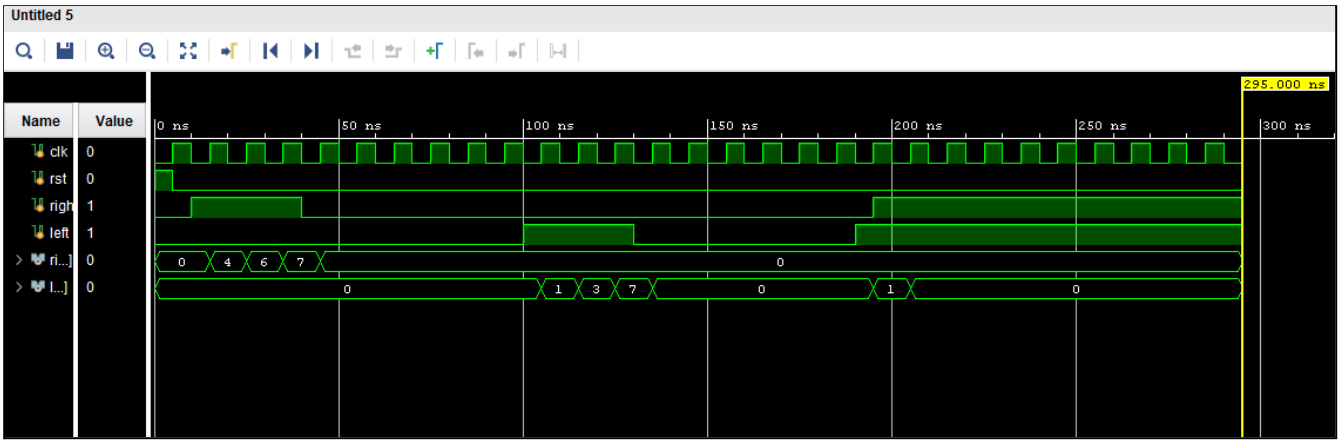
F:/CompArch/FPGA_Lab/Lab5and6/project_2/project_2.srcs/sim_1/new/Car_FSM_test.v

```
23  module Car_FSM_test(
24
25      );
26      reg clk, rst, right, left;
27      wire [2:0] rightLED, leftLED;
28      // module instantiation
29      Car_FSM car(.clk(clk), .rst(rst), .right(right), .left(left), .leftLED(leftLED), .rightLED(rightLED));
30
31      always #5 clk = ~clk;
32
33      initial             // Checking all the possible inputs
34      begin
35      clk <= 0;
36      rst <= 1;
37      right <= 0;
38      left <= 0;
39      #5 rst <= 0;
40      #5 right <= 1;
41      #30 right <= 0;
42      #60 left <= 1;
43      #30 left <= 0;
44      #60 left <= 1;
45      #5 right <= 1;
46
47      #100 $finish;
48
```

:

<u>Question</u>: Paste the image showing the simulated waveforms for FSM (Behavioral Simulation). Clearly show the LEFT turn and RIGHT turn Cases.

Answer:



4. Add clock division code to **Car_FSM.v** such that the actual input **Clk (Y9 pin with frequency of 100MHz)** is converted to Clock of frequency 2Hz. This 2Hz signal is used as clock for running the FSM.

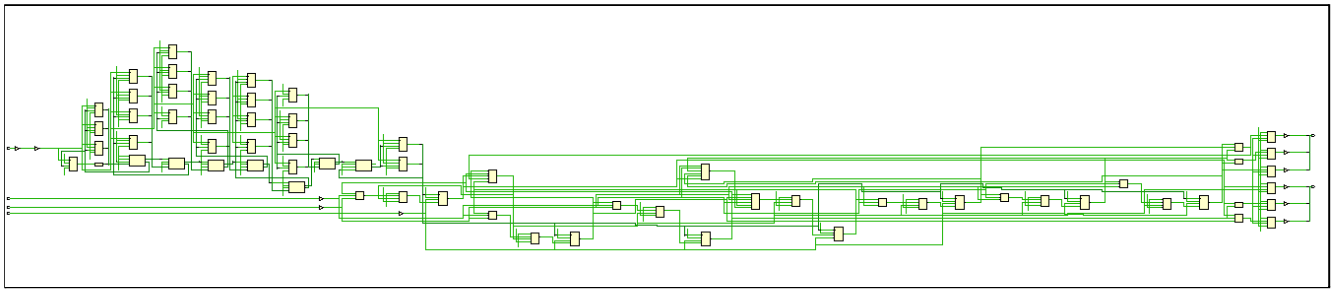
5. Plan your I/O mapping (using **I/O planning** option) such that actual input **Clk** is connected to internal clock pin **Y9**, **Reset** is connected to push button switch, other inputs (**LEFT** and

RIGHT) are connected to DIP switches and outputs are connected to LEDs. In the ZedBoard, the pin numbers indicating the DIP switches, LEDs and internal clock are listed in table uploaded in CMS. Save the mapping information as **Car_FSM.xdc**.

6. Synthesize (**Run Synthesis**).

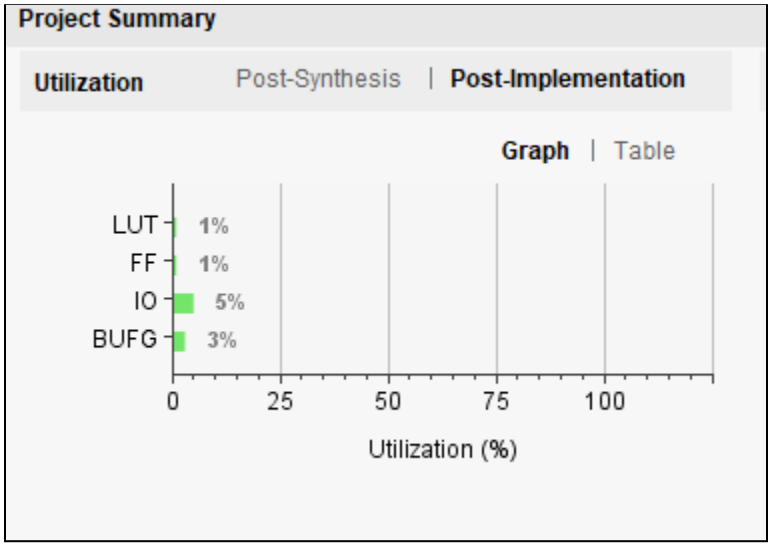
Question: Paste the image showing the schematic after synthesis.

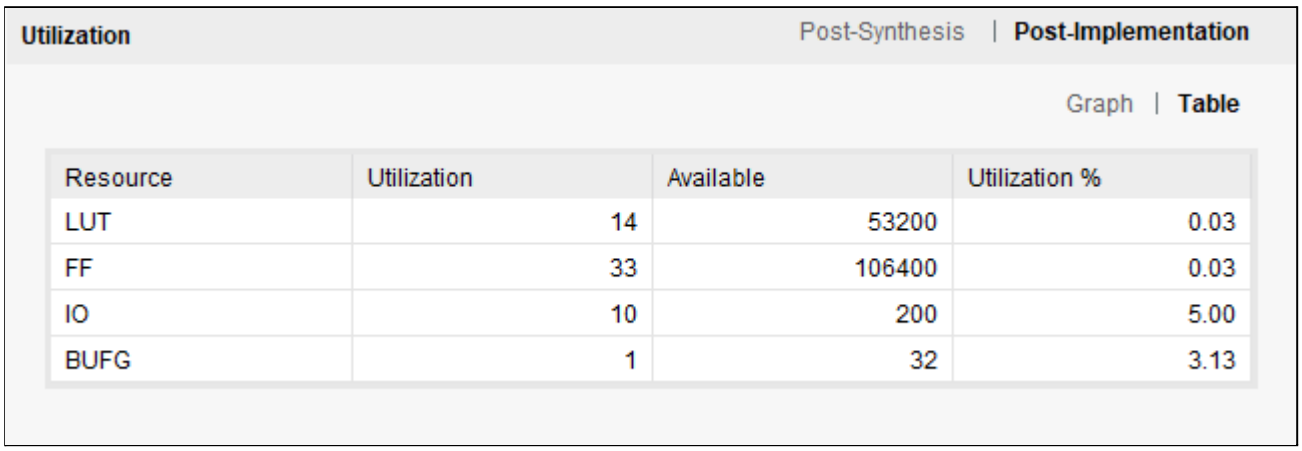
Answer:



Question: Check the summary report and report hardware utilization for the FSM implementation.

Answer: Hardware utilization





Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	14	53200	0.03
FF	33	106400	0.03
IO	10	200	5.00
BUFG	1	32	3.13

7. Implement the design (**Run Implementation)**.

8. **Generate Bitstream** and port your design on to FPGA (**Open Hardware Manager New Target** … **Program Device**)

9. **Check the output on FPGA.**

10. **Show the output to the instructor.**

11. **Submit following files as a Zipped folder with file name as <Student1_ID_No>_<Name>.zip through CMS before due date.**
    1) Completed Document
    2) Car_FSM.v (with proper comments)
    3) Test_Car_FSM.v (with proper comments)
    4) Car_FSM.xdc.
    5) Car_FSM.bit