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: Lakshmi Mounika Chaturvedula

ID No: 2019A8PS0621H

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:

<u>Inputs:</u> RIGHT→ F22; LEFT→G22;

<u>Outputs:</u> LC→U14; LB→U19; LA→W22; RA→U22; RB→T21; RC→T22;

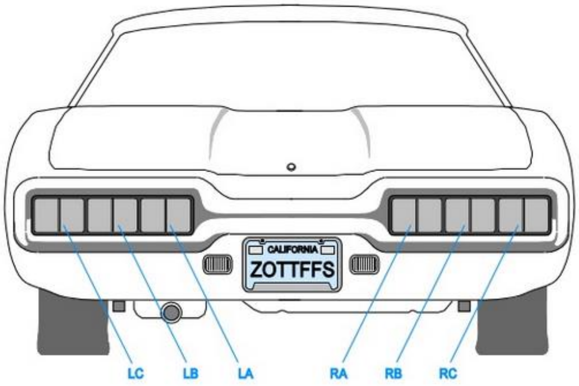


Figure 1

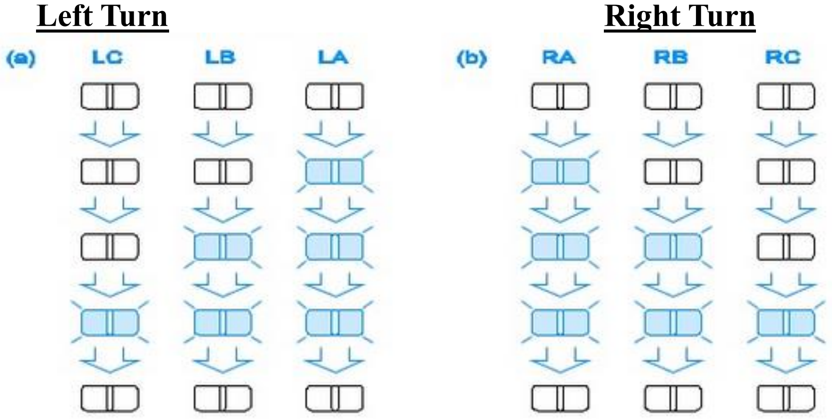


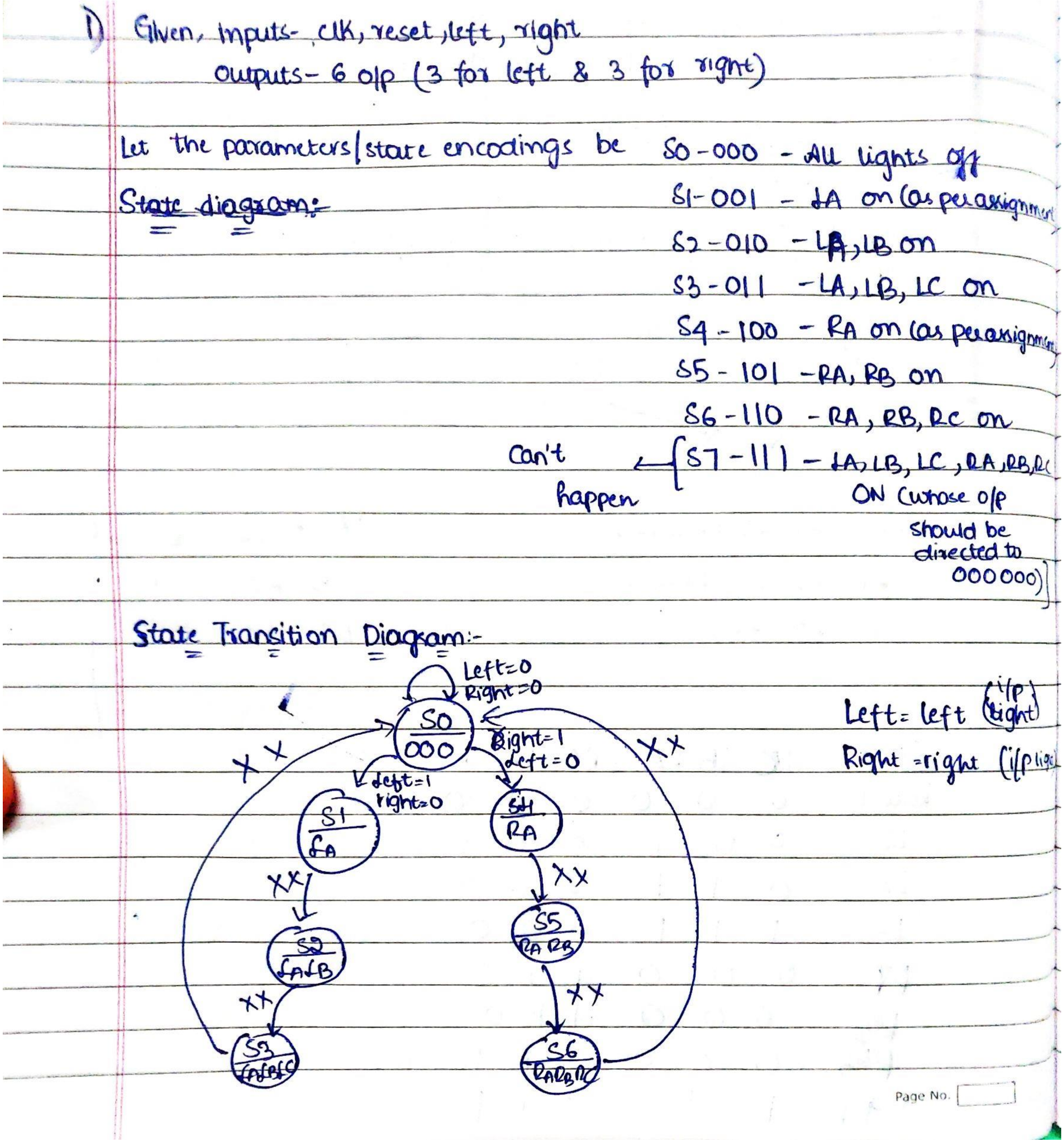
Figure 2(a) **Figure 2(b)**

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:



1) Given, inputs- clk, reset, left, right
outputs- 6 o/p (3 for left & 3 for right)

Let the parameters/state encodings be S0-000 - All lights off
S1-001 - LA on (as per assignment
S2-010 - LA, LB on
S3-011 - LA, LB, LC on
S4-100 - RA on (as per assignment)
S5-101 - RA, RB on
S6-110 - RA, RB, RC on
Can't happen ← [S7-111 - LA, LB, LC, RA, RB, RC ON (whose o/p should be directed to 000000)]

State diagram:

State Transition Diagram:-

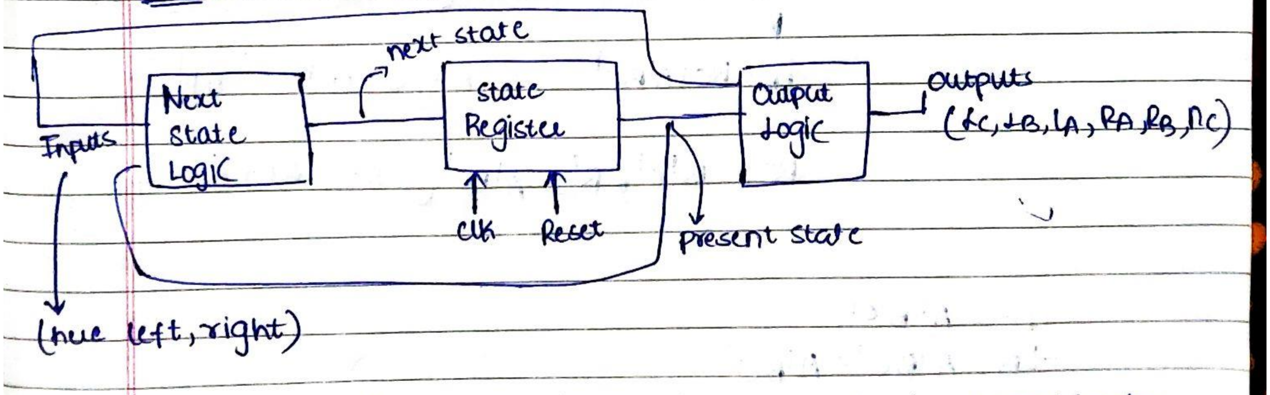
Left= left (i/p light)
Right =right (i/p lig

Page No.

State Transition Table (with the encodings):

Present State	Input left	Input right	Next State	Output out LC LB LA RA RB RC
S0 (000)	0	0	S0 (000)	000 000
S0 (000)	1	0	S1 (001)	001 000
S1 (001)	X	X	S2 (010)	011 000
S2 (010)	X	X	S3 (011)	111 000
S3 (011)	X	X	S0 (000)	000 000
S0 (000)	0	1	S4 (100)	000 100
S4 (100)	X	X	S5 (101)	000 110
S5 (101)	X	X	S6 (110)	000 111
S6 (110)	X	X	S0 (000)	000 000

7SM:



* The description for next state, present state is mentioned in the table

* If Reset = = 1 then present state is S0
(positive edge only)
else present state is next state

* Even the clock is takes the posedge (clock) into verilog code

Date ___ / ___ / ______

* We can use Dflipflops ~~to de~~ for next states & output.

So, Dflipflop inputs will simply be next state values i.e,

(ABC) Present state	(L) Input (left)	(R) Input (right)	(A'B'C') Next state	Dflipflop i/p $D_A D_B D_C$	Output $L_C L_B L_A R_A R_B R_C$
000	0	0	000	000	000 000
000	0	1	100	100	000100
000	1	0	001	001	000000
000	X	X	000	000	000000
001	X	X	010	010	011000
010	X	X	011	011	111000
011	X	X	000	000	000000
100	X	X	101	101	000110
101	X	X	110	110	000111
110	X	X	000	000	000000

And any other state o/p - 000000

For Next states ⇒ $D_A = AB'C' + B'C'L'R$ $\left(\because A'B'C'L'R + \underbrace{AB'C'L'R}_{\text{from don't care}} + AB'C\right)$

$D_B = B'C + A'BC'$ $\left(\because A'B'C + A'BC' + AB'C\right)$

$D_C = B'C'LR' + A'BC'$ $\left(\because A'B'C'L'R + A'BC' + \underbrace{AB'C'LR'}_{\text{from don't care}}\right)$

$L_C = A'BC'$

$L_B = A'BC' + A'B'C$

$L_A = A'BC' + A'B'C + A'B'C'LR'$

$= A'BC' + A'B'[\cancel{(C+C')(C+LR')}]$

Similarly

$R_C = AB'C$

$R_B = AB'C + AB'C' = AB'$

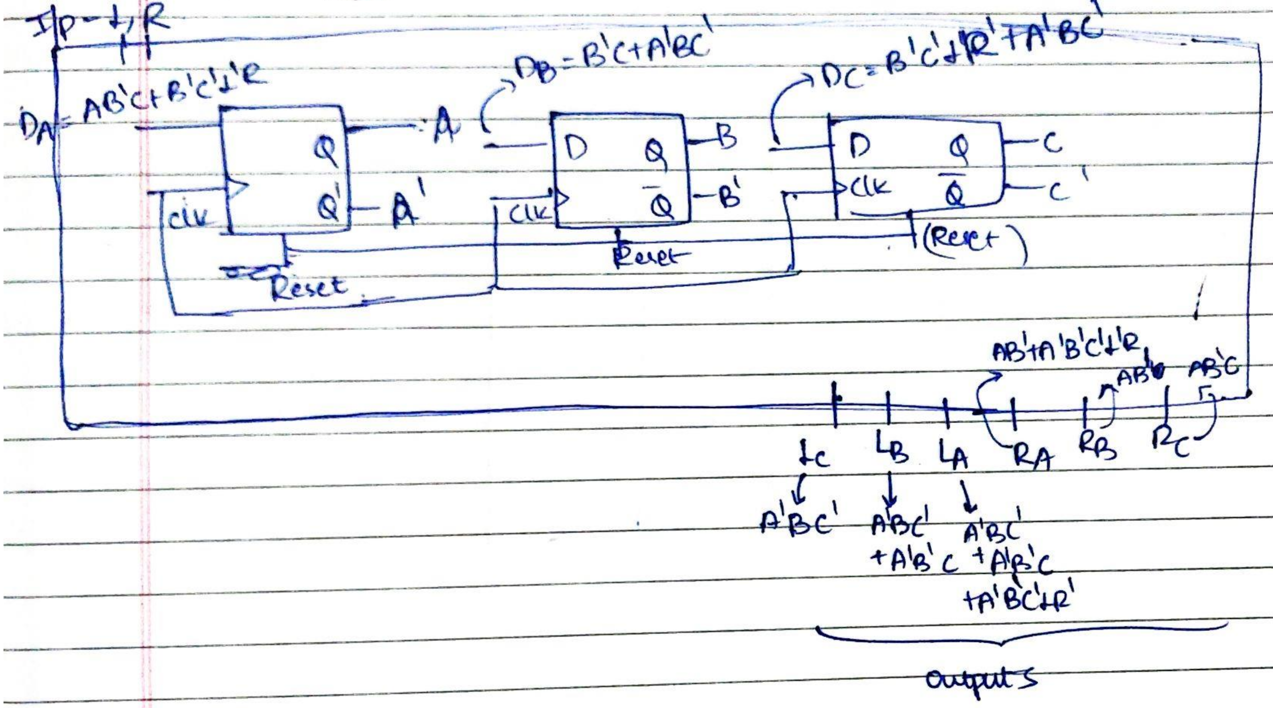
$R_A = AB'C + AB'C' + A'B'C'L'R$

$= AB' + A'B'C'L'R$

Page No

Date ___ /___ /______

Representation of FSM using D-FF

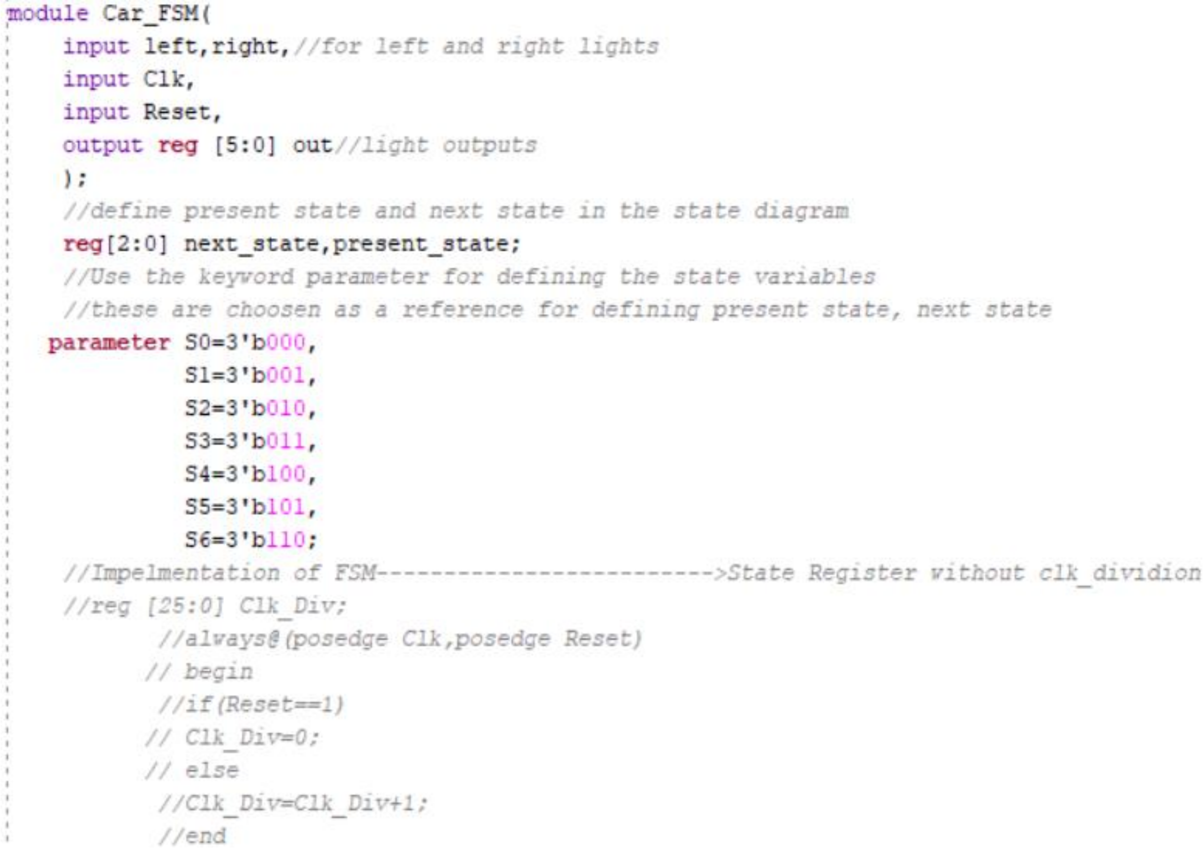


2. Create a Vivado Project and write Verilog code (**Car_FSM.v** with comments) for implementing

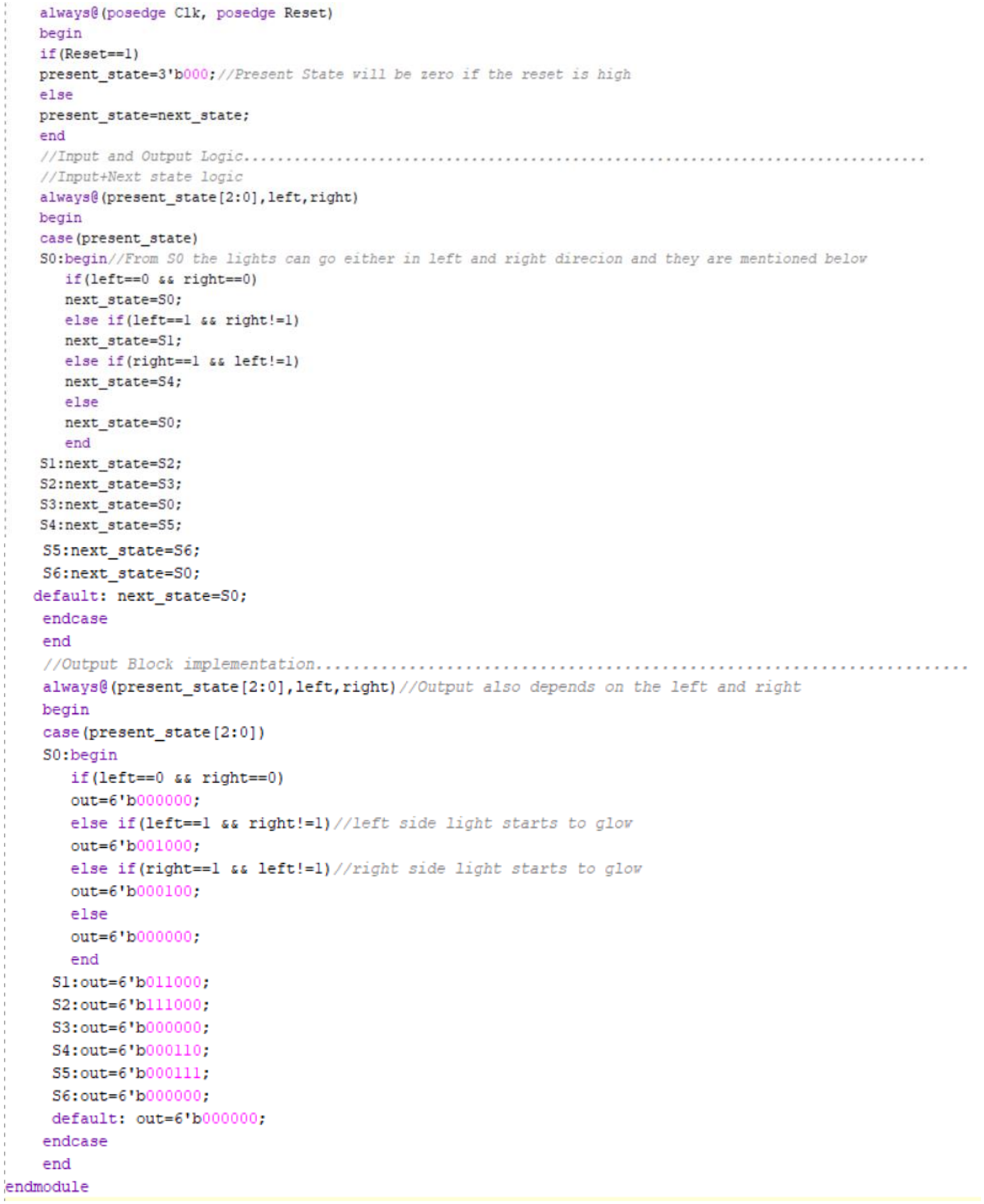
the above FSM.

Question: Paste the image of verilog code Car_FSM.v.

Answer: Without the clock division



```
module Car_FSM(
    input left,right,//for left and right lights
    input Clk,
    input Reset,
    output reg [5:0] out//light outputs
    );
    //define present state and next state in the state diagram
    reg[2:0] next_state,present_state;
    //Use the keyword parameter for defining the state variables
    //these are choosen as a reference for defining present state, next state
   parameter S0=3'b000,
             S1=3'b001,
             S2=3'b010,
             S3=3'b011,
             S4=3'b100,
             S5=3'b101,
             S6=3'b110;
    //Impelmentation of FSM------------------------->State Register without clk_dividion
    //reg [25:0] Clk_Div;
           //always@(posedge Clk,posedge Reset)
          // begin
           //if(Reset==1)
          // Clk_Div=0;
          // else
           //Clk_Div=Clk_Div+1;
           //end
```

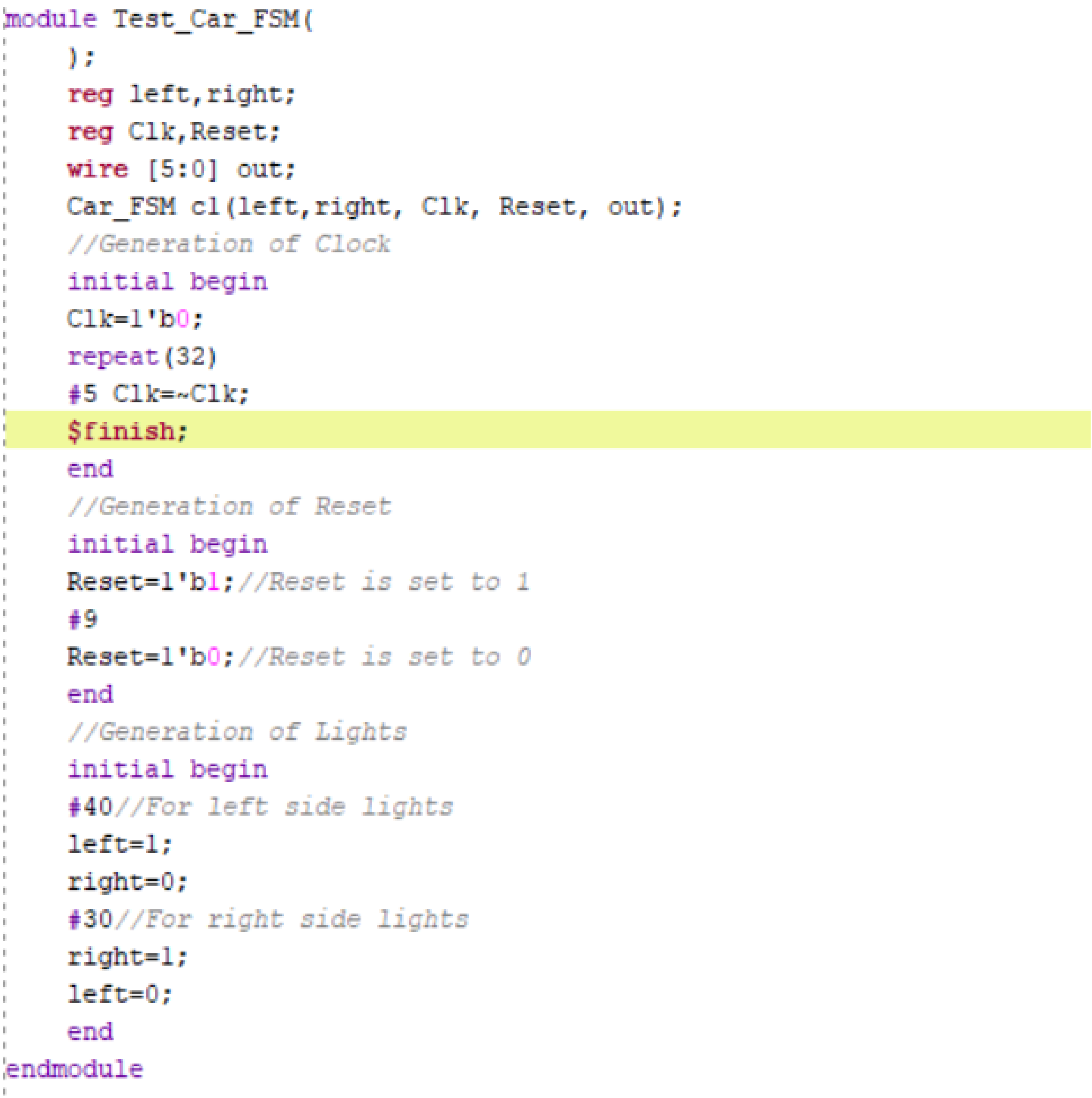


```
    always@(posedge Clk, posedge Reset)
    begin
    if(Reset==1)
    present_state=3'b000;//Present State will be zero if the reset is high
    else
    present_state=next_state;
    end
    //Input and Output Logic.............................................................................
    //Input+Next state logic
    always@(present_state[2:0],left,right)
    begin
    case(present_state)
    S0:begin//From S0 the lights can go either in left and right direcion and they are mentioned below
       if(left==0 && right==0)
       next_state=S0;
       else if(left==1 && right!=1)
       next_state=S1;
       else if(right==1 && left!=1)
       next_state=S4;
       else
       next_state=S0;
       end
    S1:next_state=S2;
    S2:next_state=S3;
    S3:next_state=S0;
    S4:next_state=S5;
    S5:next_state=S6;
    S6:next_state=S0;
   default: next_state=S0;
    endcase
    end
    //Output Block implementation.......................................................................
    always@(present_state[2:0],left,right)//Output also depends on the left and right
    begin
    case(present_state[2:0])
    S0:begin
       if(left==0 && right==0)
       out=6'b000000;
       else if(left==1 && right!=1)//left side light starts to glow
       out=6'b001000;
       else if(right==1 && left!=1)//right side light starts to glow
       out=6'b000100;
       else
       out=6'b000000;
       end
     S1:out=6'b011000;
     S2:out=6'b111000;
     S3:out=6'b000000;
     S4:out=6'b000110;
     S5:out=6'b000111;
     S6:out=6'b000000;
     default: out=6'b000000;
    endcase
    end
endmodule
```

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

Question: Paste the image of test bench verilog code Test_Car_FSM.v.

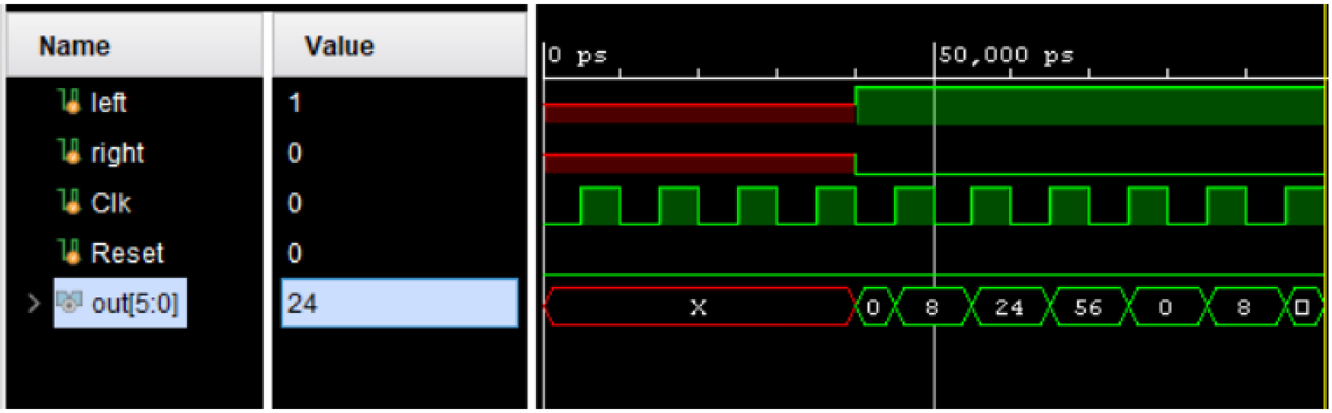
Answer:



```
module Test_Car_FSM(
    );
    reg left,right;
    reg Clk,Reset;
    wire [5:0] out;
    Car_FSM c1(left,right, Clk, Reset, out);
    //Generation of Clock
    initial begin
    Clk=1'b0;
    repeat(32)
    #5 Clk=~Clk;
    $finish;
    end
    //Generation of Reset
    initial begin
    Reset=1'b1;//Reset is set to 1
    #9
    Reset=1'b0;//Reset is set to 0
    end
    //Generation of Lights
    initial begin
    #40//For left side lights
    left=1;
    right=0;
    #30//For right side lights
    right=1;
    left=0;
    end
endmodule
```

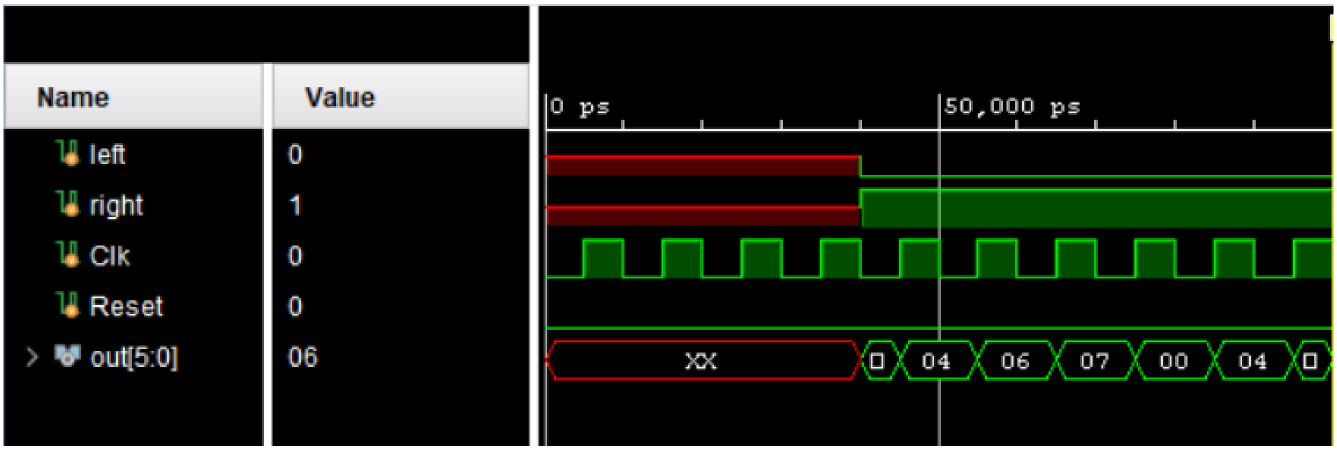
Question: Paste the image showing the simulated waveforms for FSM (Behavioral Simulation). Clearly show the LEFT turn and RIGHT turn Cases.

Answer: LEFT TURN: 0001000-8; 011000-24; 111000-56 (Put Left=1 and Right=0 in the Test Bench)



Name	Value
left	1
right	0
Clk	0
Reset	0
out[5:0]	24

RIGHT TURN: 000100-4;000110-6;000111-7 (Put Right= 1 and Left=0 in the Test Bench)



Name	Value
left	0
right	1
Clk	0
Reset	0
out[5:0]	06

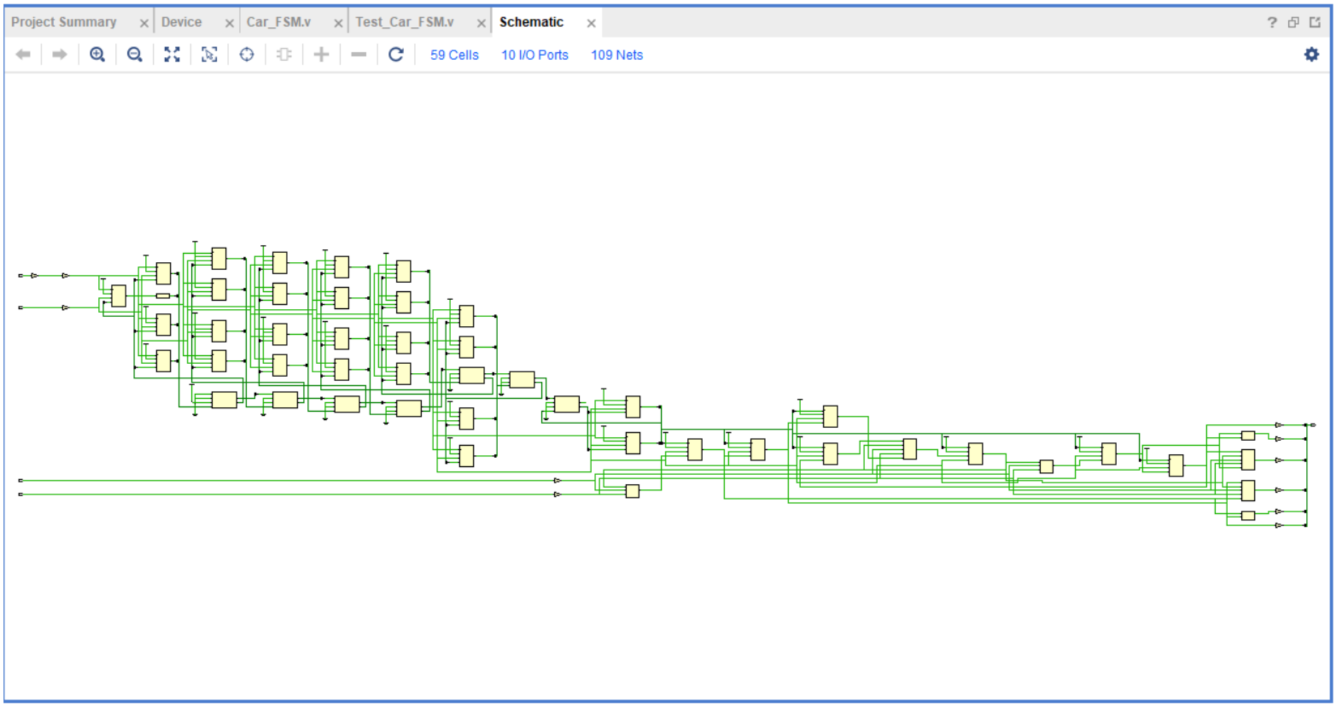
4. Add clock division code to **Car_FSM.v** such that the actual input **Clk (Y9 pin with frequency of 100MHz)** is converted to Clock of frequency 2Hz. This 2Hz signal is used as clock for running the FSM.

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**).

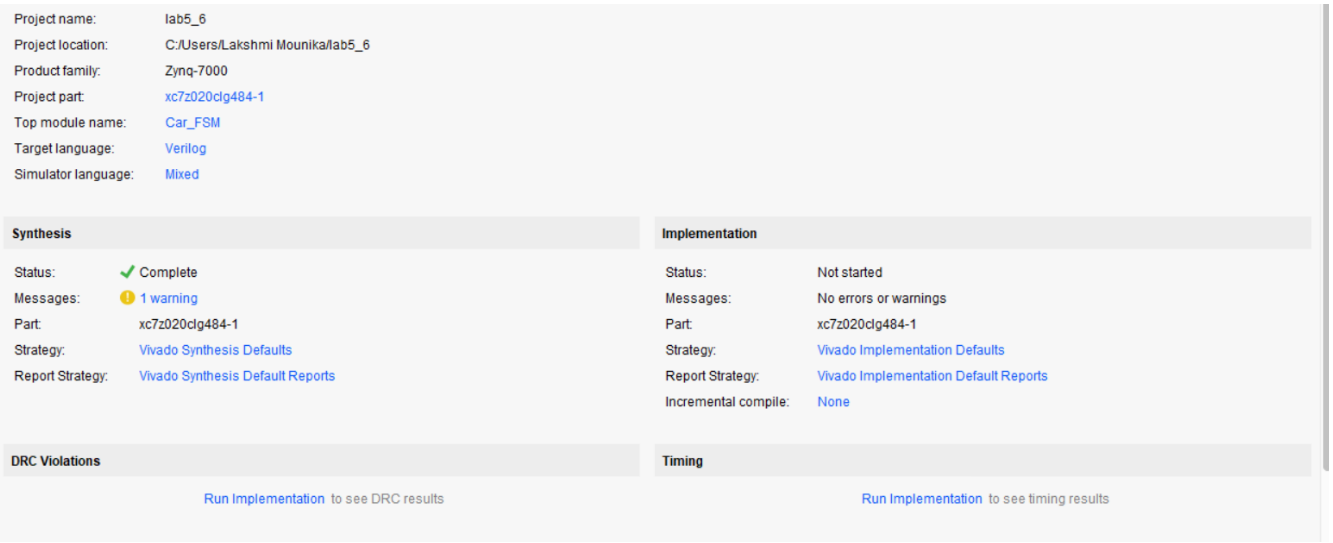
   Question: Paste the image showing the schematic after synthesis.

Answer:



<u>Question</u>: Check the summary report and report hardware utilization for the FSM implementation.

Answer: Hardware Utilization=5+3.13+0.03+0.02=8.18%



Project name:	lab5_6
Project location:	C:/Users/Lakshmi Mounika/lab5_6
Product family:	Zynq-7000
Project part:	xc7z020clg484-1
Top module name:	Car_FSM
Target language:	Verilog
Simulator language:	Mixed

Synthesis

Status:	✓ Complete
Messages:	1 warning
Part:	xc7z020clg484-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports

Implementation

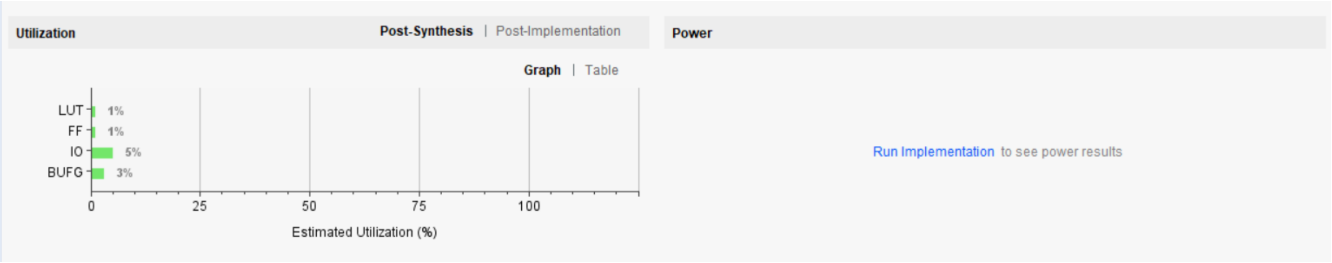
Status:	Not started
Messages:	No errors or warnings
Part:	xc7z020clg484-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental compile:	None

DRC Violations

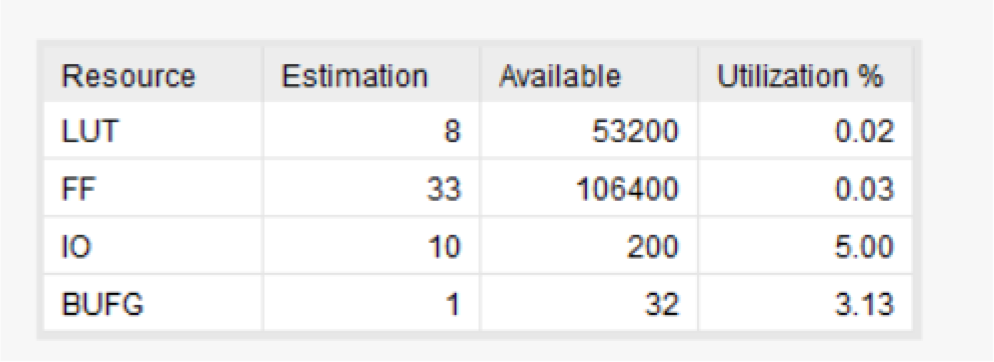
Run Implementation to see DRC results

Timing

Run Implementation to see timing results



Run Implementation to see power results



Resource	Estimation	Available	Utilization %
LUT	8	53200	0.02
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