Rohan Jhaveri

830962238

Lab Report 9

Goal: The objective of this lab is to gain experience with state machine design by using the six-step design process. From the project specifications, you will:

Develop a state diagram,

- Reduce the number of states using the implication chart.
- Assign the reduced states the optimal binary values.
- Develop the excitation table using T flip-flops.
- Design the logic and implement the finite state machine.

Function: Design a variation of the classical traffic light controller. The intersection is shown in the following diagram. "A" street runs north-south, "B" street runs east-west, and "C" street enters the intersection from the southeast. "A" street is quite busy, and it is frequently difficult for cars heading south on "A" to make the left turn onto either "B" or "C". In addition, cars rarely enter the intersection from "C" street. Design a traffic light state diagram for this three-way intersection to the following specifications:

a. There are five sets of traffic lights, facing cars coming from "A" north, "A" south, "B" east, "B" west, and "C" street.

b. The red, yellow, and green lights facing cars from "A" north are augmented with a left turn arrow that can be lit up as either green or yellow or not lit up at all.

c. The normal sequencing of lights facing the cars coming from "A" north is arrow green, arrow yellow, traffic light green, traffic light yellow, traffic light red, and repeat. In other words, the left arrow light is illuminated in every complete cycle of the lights.

d. However, it should be possible for traffic going from north to south on "A" street to cross the intersection even when the left turn arrow is illuminated. Therefore, the traffic light green should also be illuminated while the turn arrow is lit up.

e. Cars traveling from south to north on "A" street (and all directions on "B" and "C" streets) must see a red light while the left turn arrow is illuminated for the traffic heading south.

f. A car sensor C is embedded in "C" street to detect whether a car is waiting to enter the intersection from the southeast.

g. A timer generates a long interval signal, Time Long (TL), and a short interval signal, Time Short (TS) when set by a Short Time (ST) signal.

h. Red and green lights are lit up for at least a TL unit of time. Yellow lights, the green arrow, and the yellow arrow are lit up for exactly a TS unit of time.

i. The "C" street lights cycle from red to green only if the embedded car sensor indicates that a car is waiting. The lights cycle to yellow and then red as soon as no cars are waiting. Under no circumstances is the "C" street green light to be lit for longer than a TL unit of time.

Inputs: Sensor C, short and long time intervals TS and TL.

Outputs:
"A" street north arrow green/yellow/off (three separate signals)

"A" street north light green/yellow/red

"A" street south light green/yellow/red

"B" street east light green/yellow/red

"B" street west light green/yellow/red

"C" street southeast green/yellow/red and the timer set signal ST

Steps:

1. Made state diagram (FSM) and transition table. (Refer Prelab)

2. Implemented the circuit in Verilog.

3. Implemented the circuit in Cadence.

Verilog Code:

```
module traffic(clk, c, ts, tl, current_state, next_state);

input clk, c, ts, tl;

output current_state, next_state;

reg [2:0] current_state, next_state;


localparam s0=3'd0,

s1=3'd1,

s2=3'd2,
```

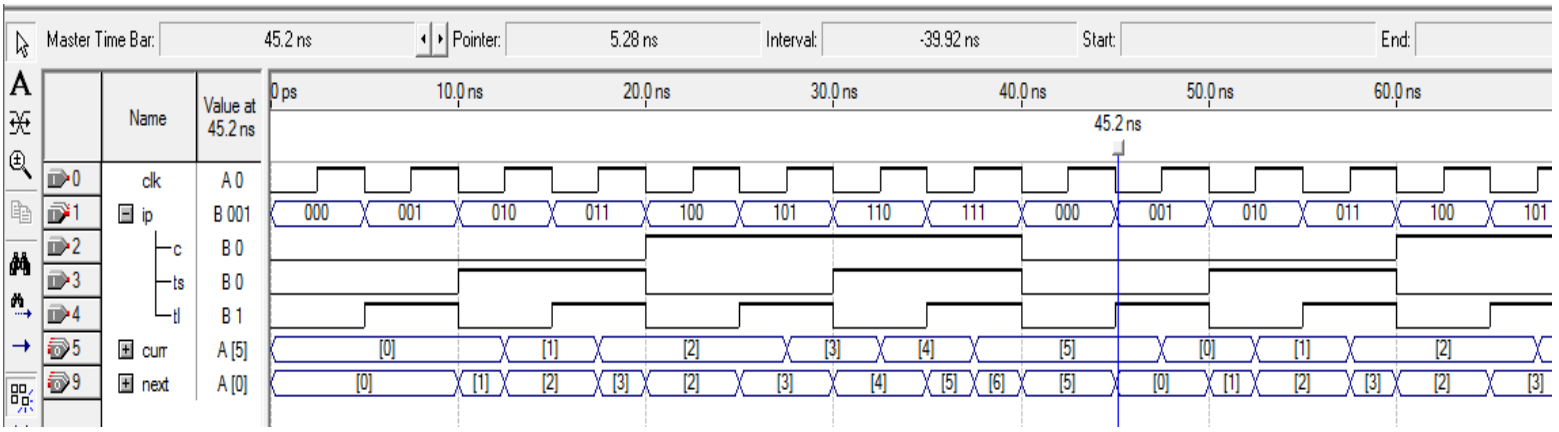
```
s3=3'd3,
s4=3'd4,
s5=3'd5,
s6=3'd6,
s7=3'd7;

always@(posedge clk)
current_state<=next_state;

always@(*)
begin
next_state=current_state;
case(current_state)
s0:
begin
if(ts==0)
current_state=s0;
else if(ts==1)
next_state=s1;
end
s1:
begin
if(ts==0)
next_state=s1;
else if(ts==1)
next_state=s2;
end
```

```
s2:
begin
if(tl==0)
next_state=s2;
else if(tl==1)
next_state=s3;
end
s3:
begin
if(ts==0)
next_state=s3;
else if(tl==0 & ts==1)
next_state=s4;
end
s4:
begin
if(tl==0)
next=s4;
else if(tl==1)
next=s5;
end
s5:
begin
if(tl==0)
next_state=s5;
else if(c==0 & tl==1)
next_state=s0;
```

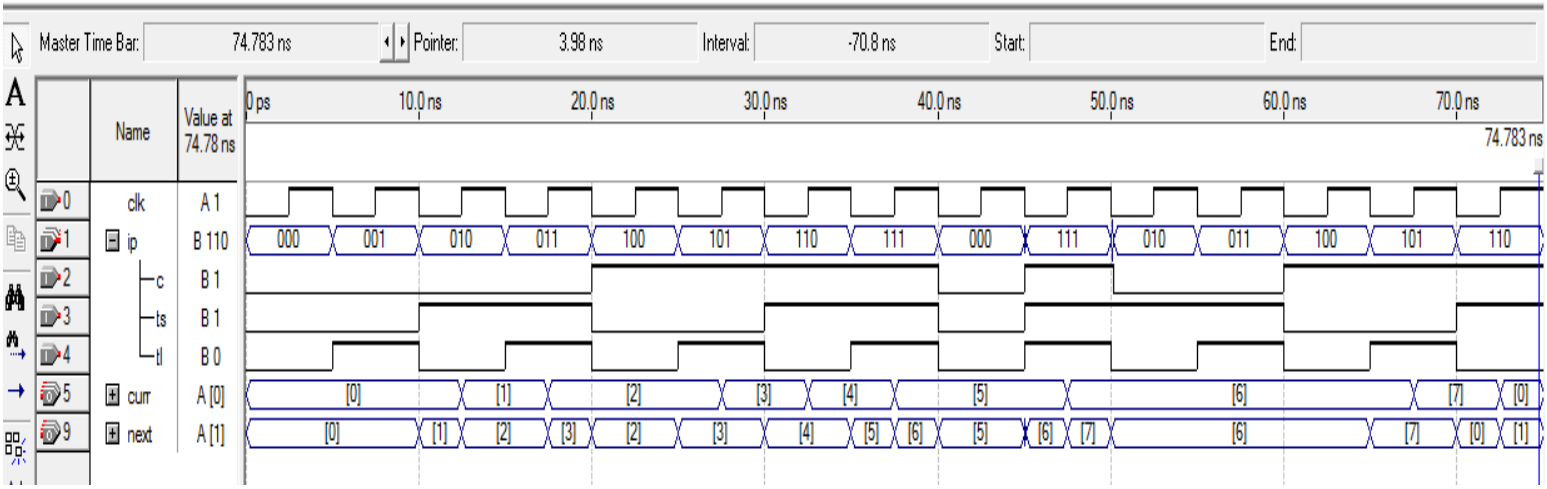
```
else if(c==1 & tl==1)
next_state=s6;
end
s6:
begin
if(c==1 & tl==0)
next_state=s6;
else if(c==1 & tl==1)
next_state=s7;
end
s7:
begin
if(ts==0)
next_state=s7;
else if(ts==1)
next_state=s0;
end
endcase
end
endmodule
```

Verilog Outputs:

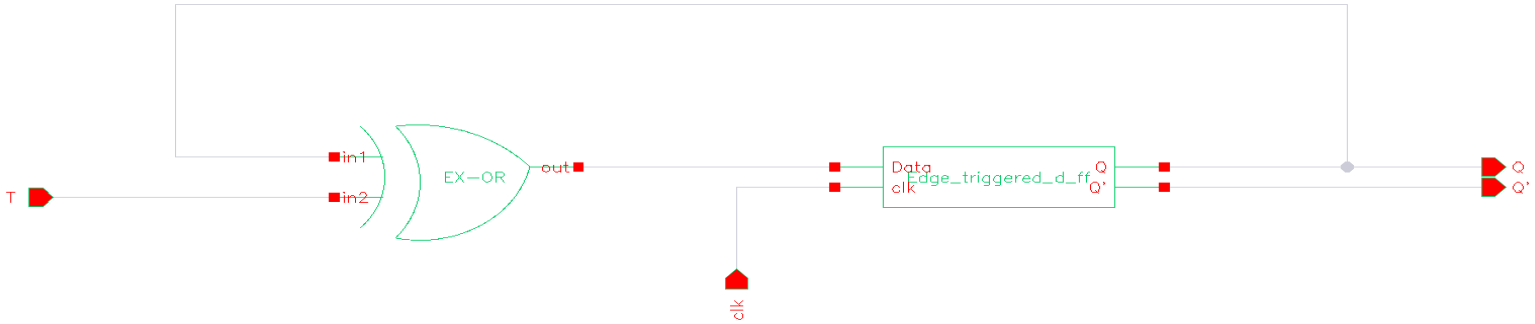
When no car at C street:



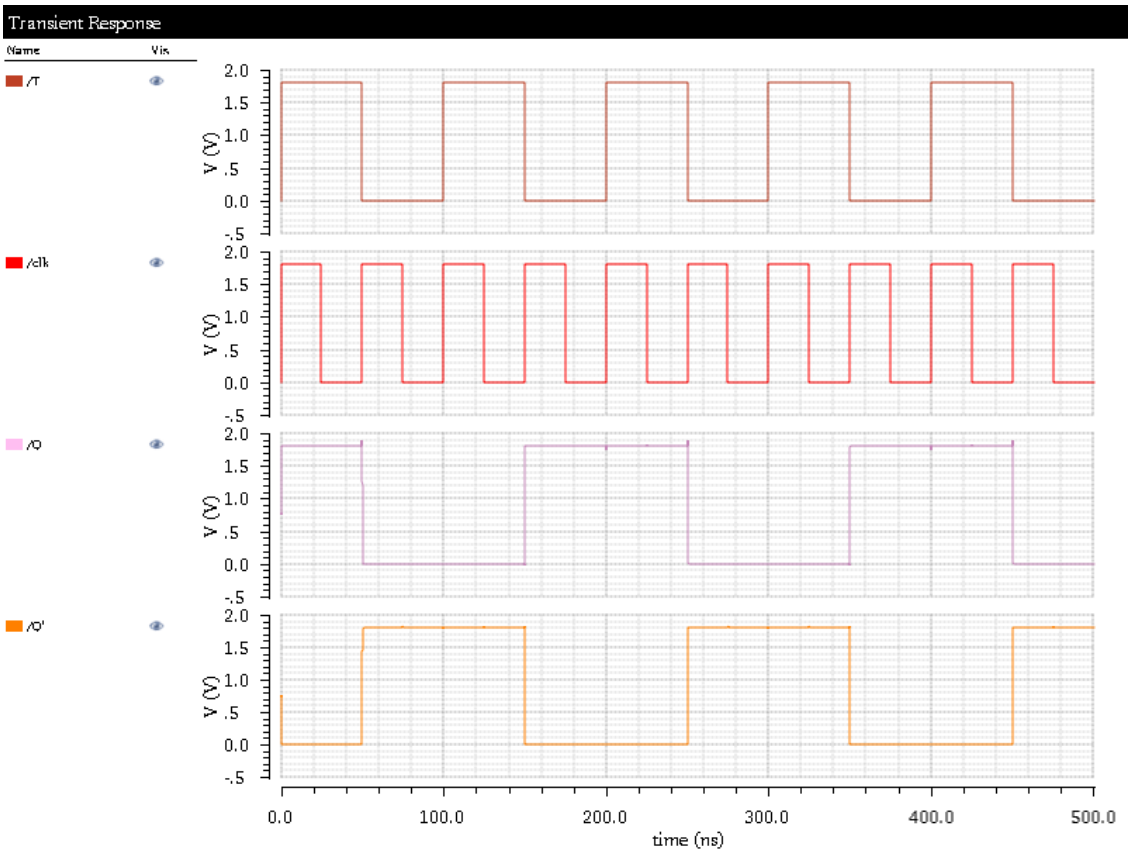
When Car on C street:



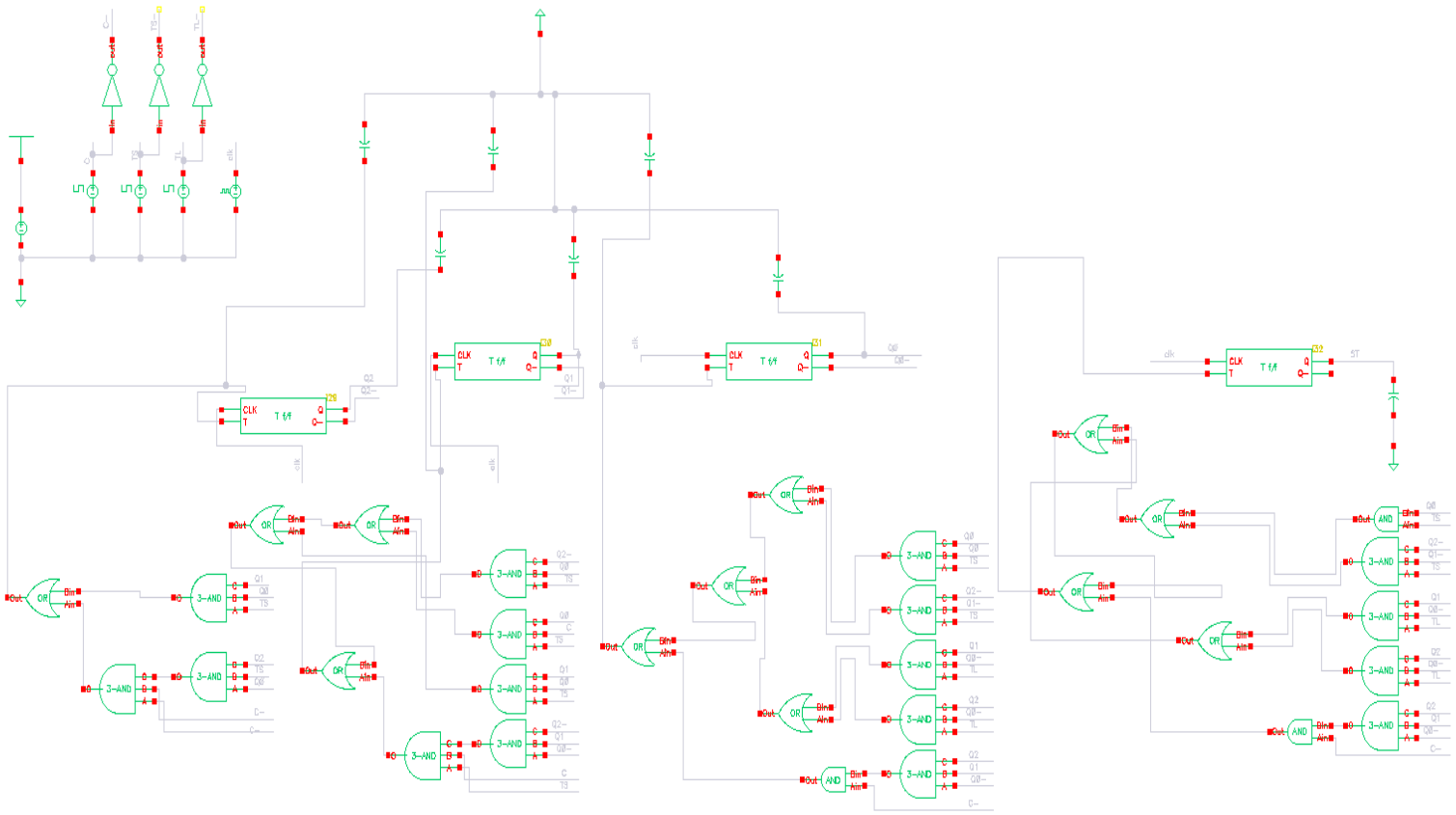
T filp flop Schematic:



T flip flop output:

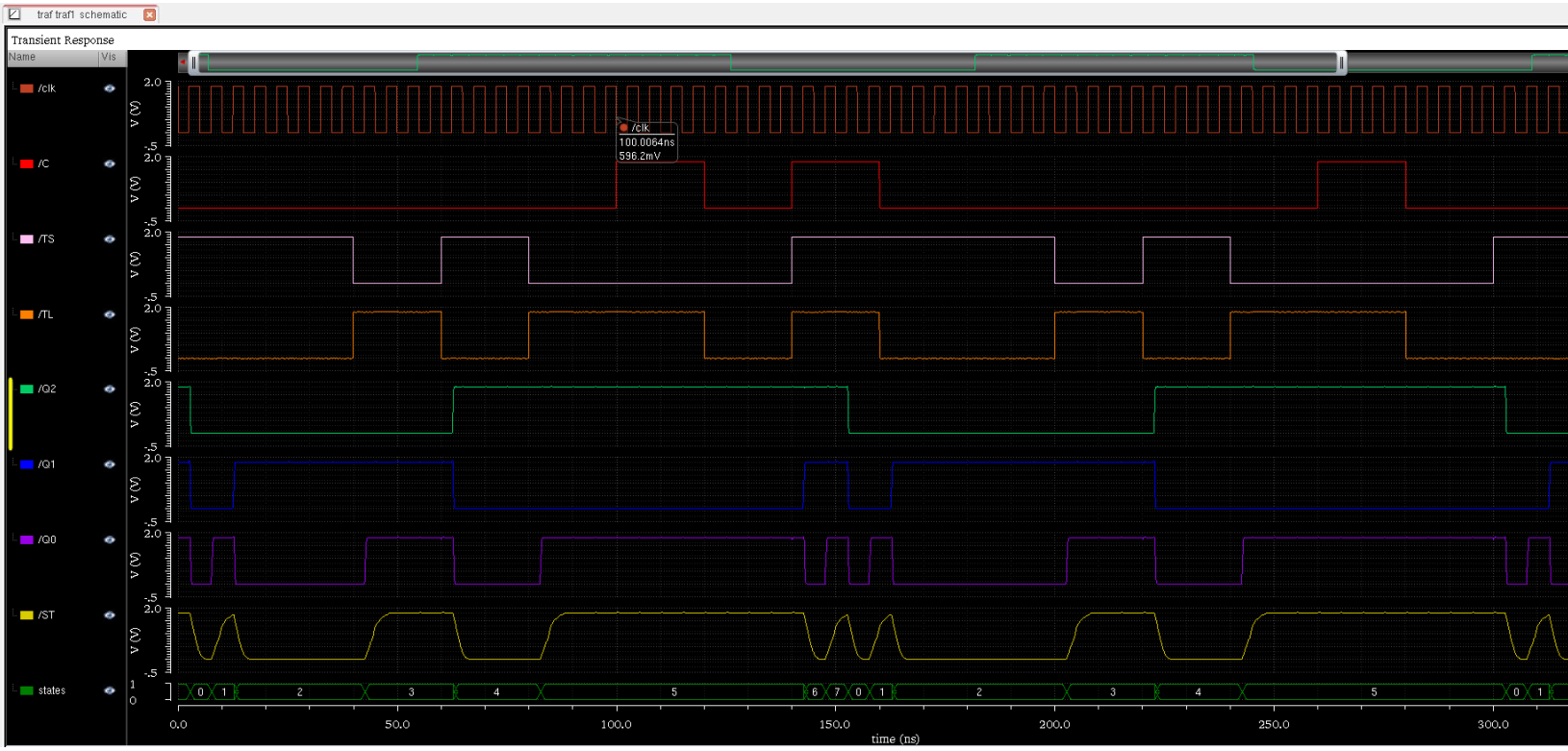


Traffic signal schematic:

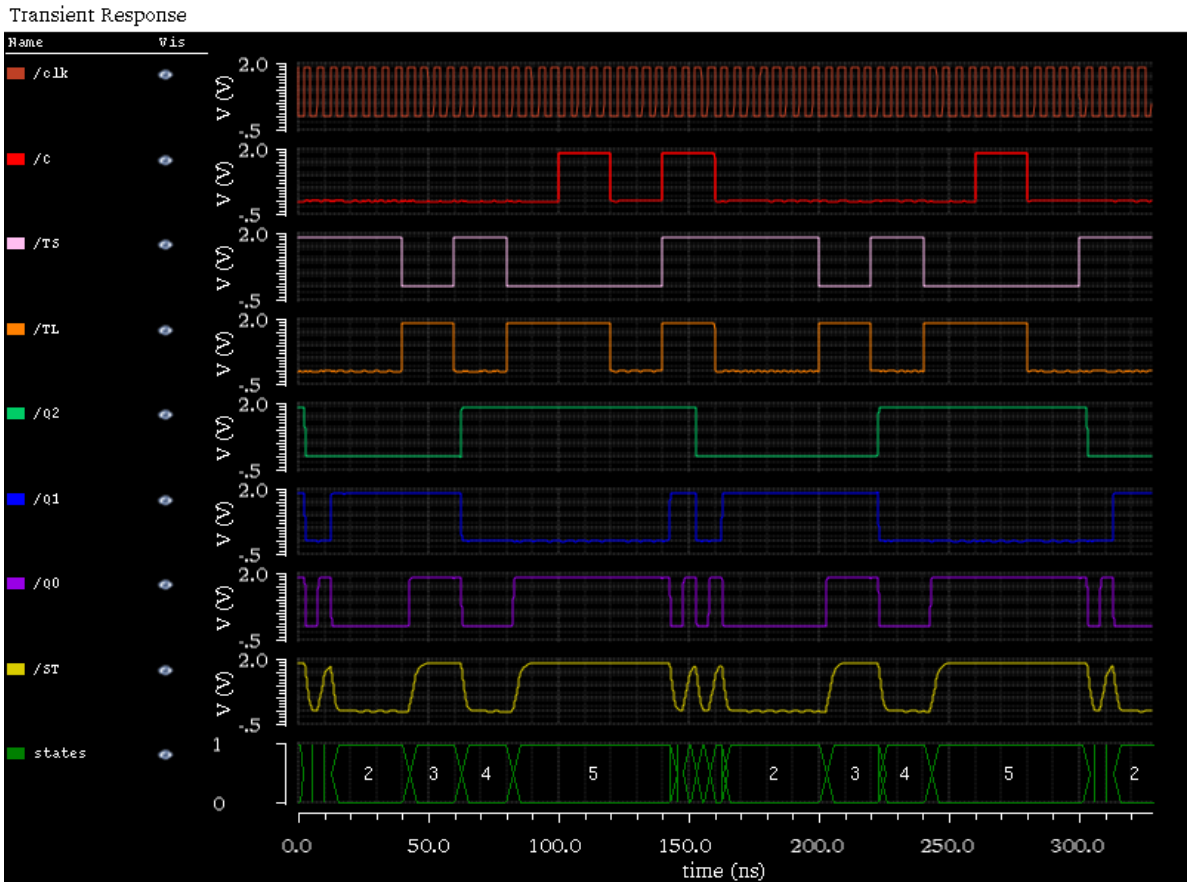


Traffic Signal Outputs:

When car on C street:



When no car on C street:



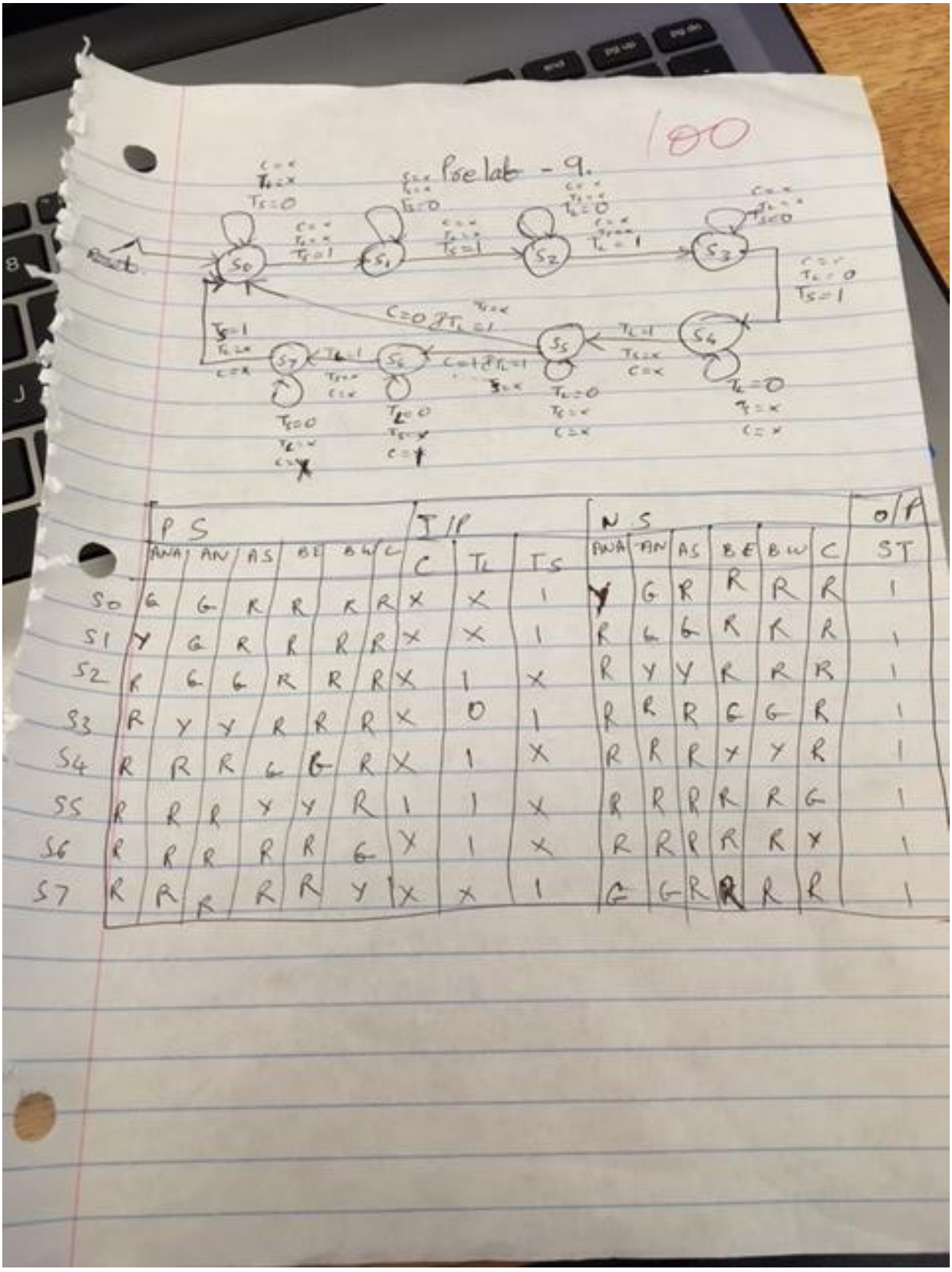
Conclusion: In this lab, we learnt how to implement a complex FSM and simulate it in Verilog and Cadence.

Questions:

1. Explain a "race condition" in the context of a Finite State Machine (FSM).
Ans: Race Condition in FSM is when more than one state transition occurs in a single clock cycle.

2. Explain the advantages of state reduction, state assignment selection and type of flip-flop selection in a FSM.
Ans: State reduction or state optimization helps to reduce number of states in a state transition diagram. State assignment technique in an FSM like One hot encoding, grey code encoding helps reduce power and cost of the entire circuit and both these leads to easy debugging of circuits. Also, Flip flop selection techniques helps to reduce the number of gates requires and also the power requirement of the circuit.

Pre-lab



	P S						I/P			N S						O/P
	NWA	AN	AS	BE	BW	C	C	TL	TS	NWA	AN	AS	BE	BW	C	ST
S0	G	G	R	R	R	R	X	X	1	Y	G	R	R	R	R	1
S1	Y	G	R	R	R	R	X	X	1	R	G	G	R	R	R	1
S2	R	G	G	R	R	R	X	1	X	R	Y	Y	R	R	R	1
S3	R	Y	Y	R	R	R	X	0	1	R	R	R	G	G	R	1
S4	R	R	R	G	G	R	X	1	X	R	R	R	Y	Y	R	1
S5	R	R	R	Y	Y	R	1	1	X	R	R	R	R	R	G	1
S6	R	R	R	R	R	G	X	1	X	R	R	R	R	R	Y	1
S7	R	R	R	R	R	Y	X	X	1	G	G	R	R	R	R	1