Lab Report 8

Objective: To create a subway station controller using sequential logic design.

Function: This is the continuation of Lab 3. In Lab 3, the direction signal D is given as a primary input signal. Now, let's design a finite state machine to generate signal D. Let's assume that the traffic pattern for the track is such when two consecutive left-to-right trains passed by, the next train must go from right-to-left. After that, there will be two more left-to-right trains, etc. To detect the train direction through the station, we set up two light beams just above the rail track and place two photocells, P1 and P2, some distance apart. Assume the train can fit inside of the two beams. When the beam shines on a photocell, it produces a 0, and when the beam is blocked, it produces a 1. The train may stop at the middle of the photocells. If this happens, no change is made to signal D. If the train stopping at the middle and then back to the direction it came from, then this train does not count as a passing train. Design a logic circuit to generate D.

Inputs:

P1, P2: Two photocell outputs.

Reset: Global reset.

Outputs:

D: The direction control signal.

Steps:

1. Made truth table and finite sate machine. (Refer Prelab)

2. Implemented the code in Verilog.

3. Implemented the circuit in Cadence.

Verilog Code:

```
module subway(clk,reset,p1,p2,d,curr,next);

input clk,reset,p1,p2;

output d,curr,next;

reg d;

localparam      s0=3'd0,

                s1=3'd1,

                s2=3'd2,

                s3=3'd3,

                s4=3'd4,
```

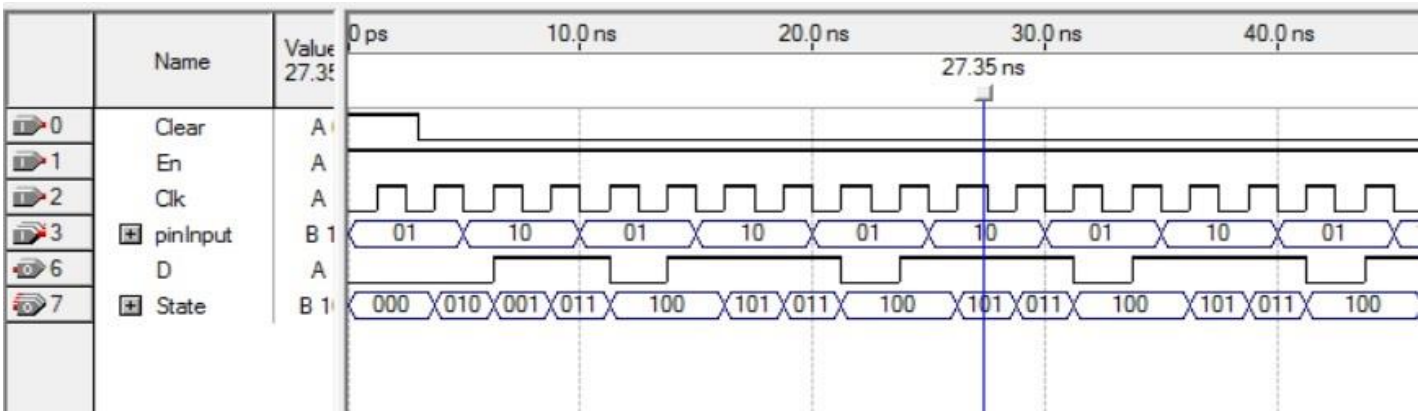
```
                s5=3'd5,
                s6=3'd6,
                s7=3'd7;
reg [2:0] curr;
reg [2:0] next;
always@ (posedge clk)
if(reset)
curr<=s0;
else
curr<=next;
always@ (*)
begin
next=curr;
case(curr)
        s0:
        begin
        if (p1 & !p2)
        begin
        next=s1;
        d=1;
        end
        else if (!p1 & p2)
        begin
        next=s5;
        d=0;
        end
        end
        s1:
        begin
```

```
if (!p1 & p2)
begin
next=s2;
d=1;
end
end
s2:
begin
if (p1 & !p2)
begin
next=s3;
d=1;
end
end
s3:
begin
if (!p1 & p2)
begin
next=s4;
d=1;
end
end
s4:
begin
if (!p1 & p2)
begin
next=s5;
d=1;
end
```

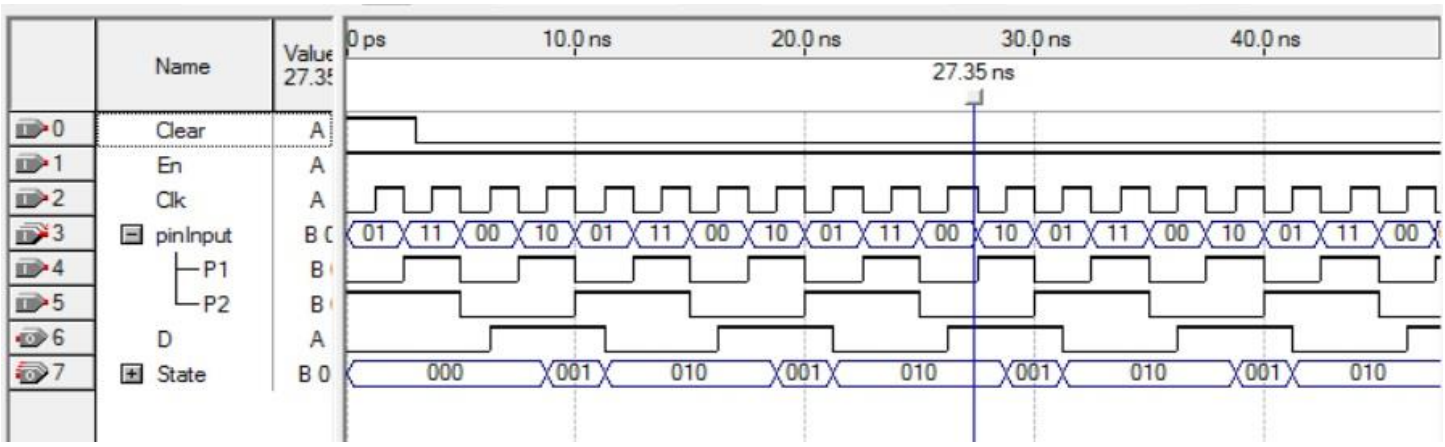
```
        end
        s5:
        begin
        if (p1 & !p2)
        begin
        next=s6;
        d=0;
        end
        end
        s6:
        begin
        if (p1 & !p2)
        begin
        next=s1;
        d=0;
        end
        end
        end case
end
end module
```

Verilog Output:

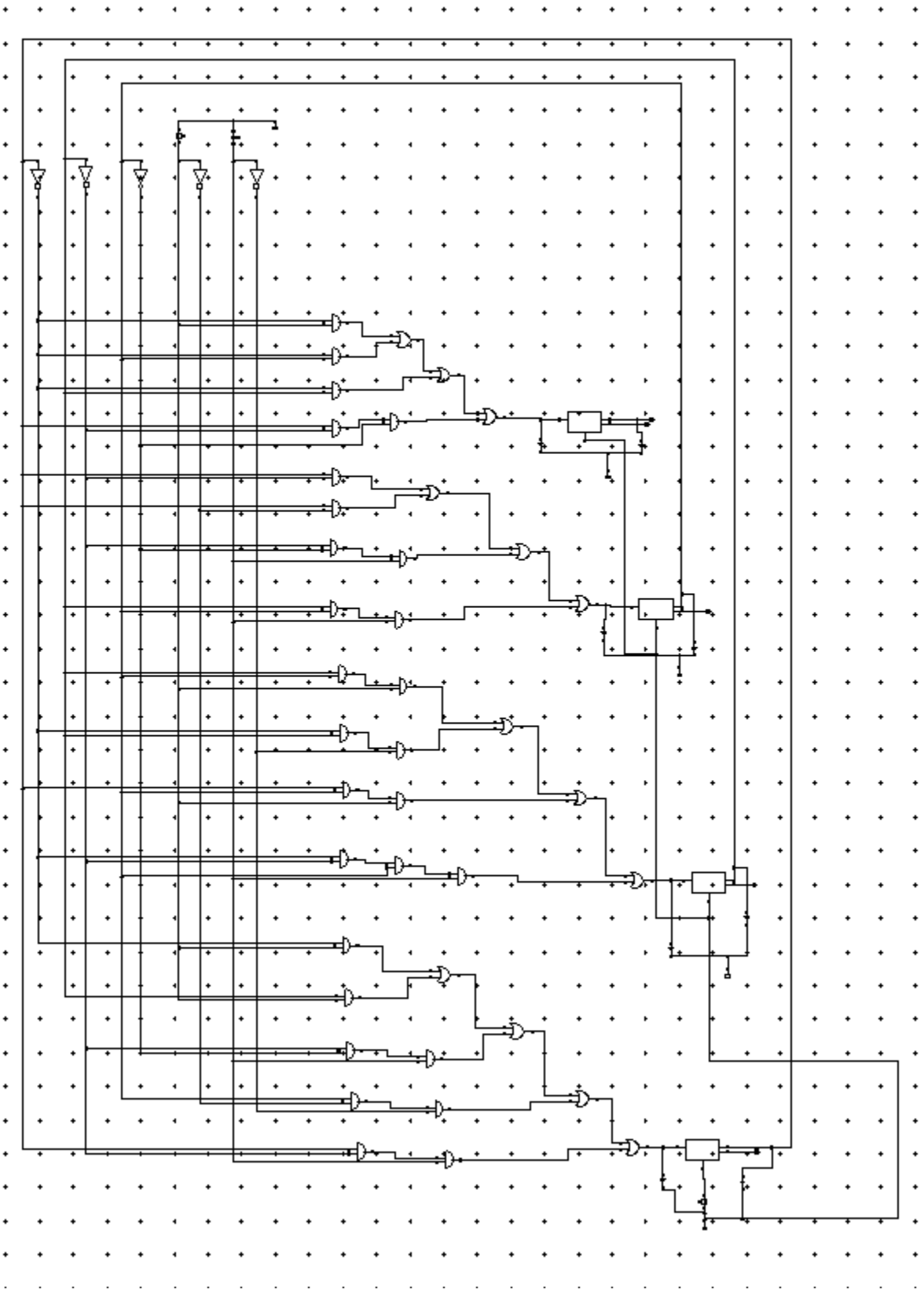
When the train comes to the station and goes back again.



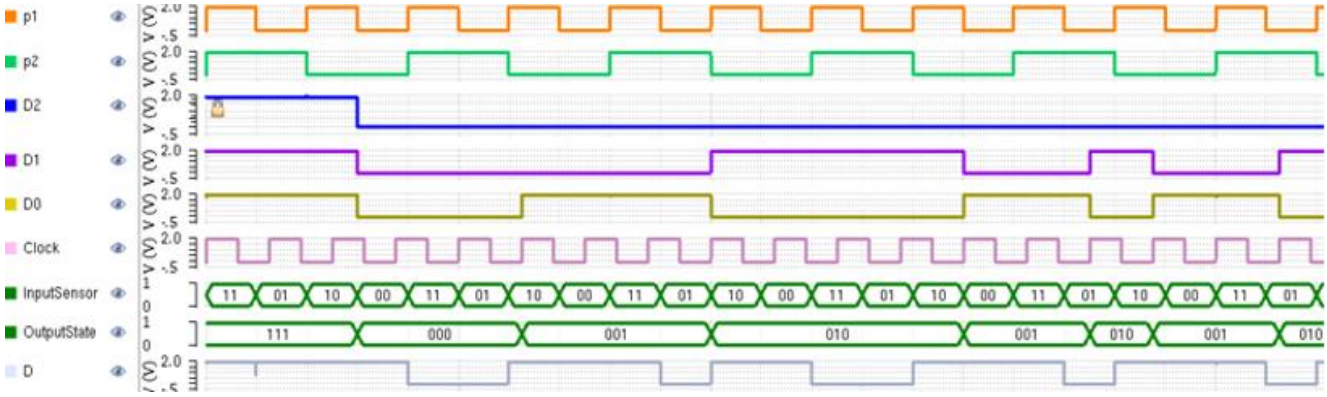
After two left bound trains, there is a right bound train.



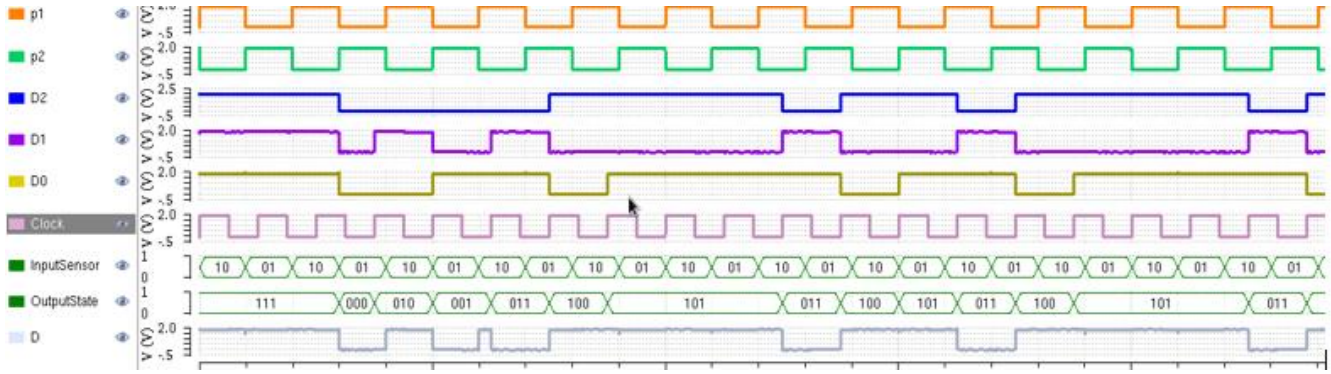
Cadence Schematic:



Cadence Output:



When the train comes to the station and goes back again.



After two left bound trains, there is a right bound train.

Conclusion:

This Lab thought us how to implement practical applications through FSM. It shows that the output of Verilog gives us ideal outputs, but cadence output nearly reflects the real world scenario as shown in the figure, where due to a small glitch the output enters state 3 for a small time.

Questions:

The output of Moore machine depends on the current state whereas the output of Mealy machine depends on current as well as the current inputs. The output of Moore machine changes on clock edge whereas the output of Mealy machine changes as soon as the there is a change in the input. It does not wait for the clock edge. Hence, Mealy machine reacts faster than the Moore machine. The number of states in Moore machine is more than the Mealy machine.