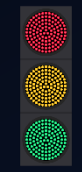
4-way Traffic Light Controller



PRESENTATION

Nikhil Raj
(B19CSE059)

Visual Overview at the design

- 4 traffic lights are used, one on every lane i.e. north, west, south, east.
- To avoid collisions and complexity, the approach of ‘One green at a time’ is used.

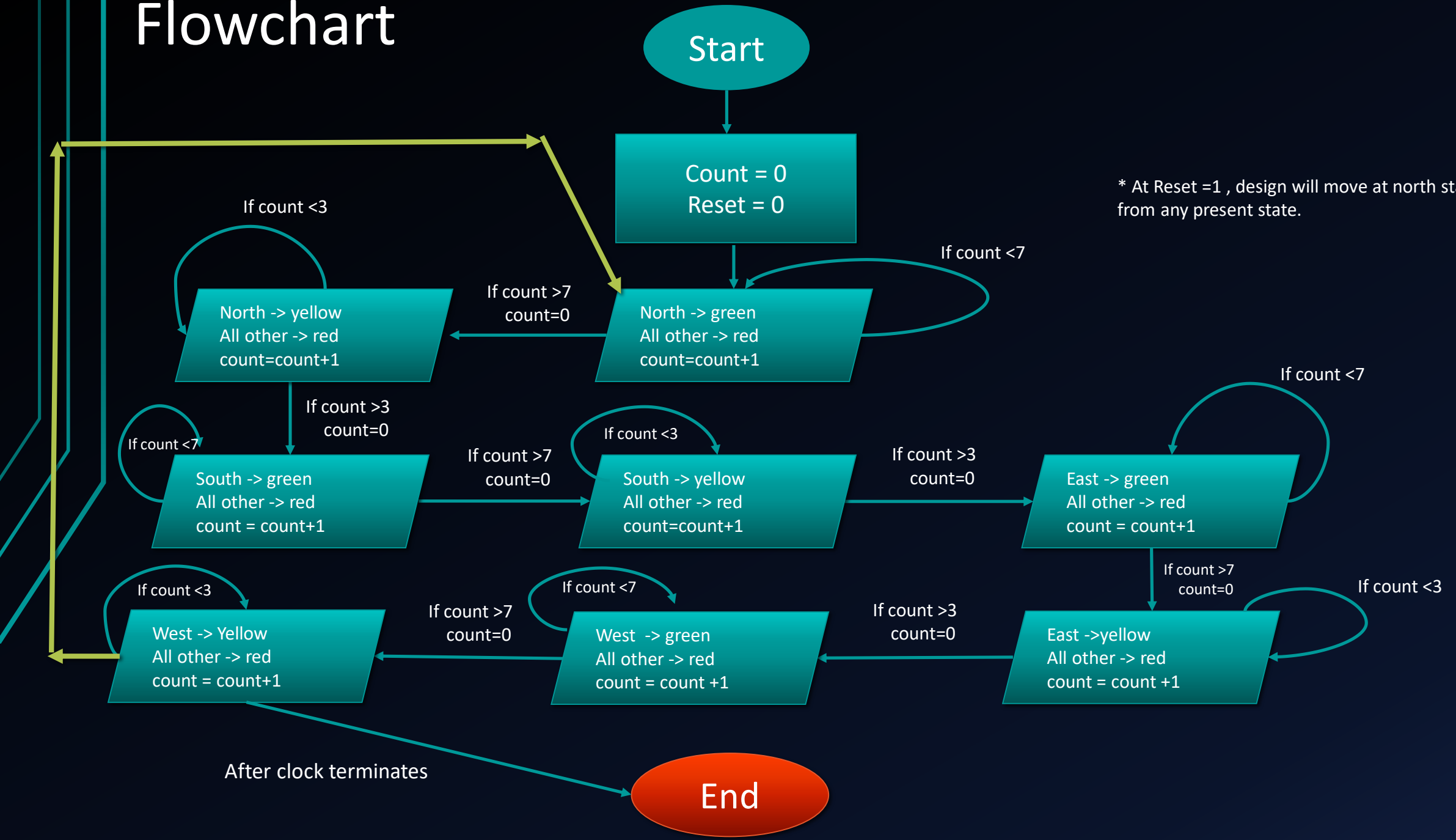
Technical Overview of the design along with working principle

- Basically there are total 8 states in which the program will enter.
- Firstly north state where green light is turned on and rest all other lights be red, then north-yellow state where yellow light is on (in north) and all red and then south state all the way to the last state of west-yellow and from west-yellow to north again (will be displayed in flowchart as well).
- Two inputs are there which are clock and reset and four outputs will be shown which are the four lights in all four directions.
- For lights, 001 is named as Green, 010 as yellow and 100 as red.

Technical Overview of the design along with working principle

- Asynchronous reset button is added which on high signal sends the current state to north state.
- For 8 clock cycles, the design will remain green and then for four clock cycles it’ll remain yellow.
- The sequence of states is : north -> north-yellow -> south -> south yellow -> east -> east-yellow -> west -> west-yellow.

Flowchart


Start
Count = 0
Reset = 0
* At Reset =1 , design will move at north state from any present state.
If count <3
If count <7
North -> yellow
All other -> red
count=count+1
If count >7
count=0
North -> green
All other -> red
count=count+1
If count >3
count=0
If count <7
If count <3
South -> green
All other -> red
count = count+1
If count >7
count=0
South -> yellow
All other -> red
count=count+1
If count >3
count=0
If count <7
East -> green
All other -> red
count = count+1
If count >7
count=0
If count <3
If count <3
West -> Yellow
All other -> red
count = count+1
If count >7
count=0
If count <7
West -> green
All other -> red
count = count +1
If count >3
count=0
East ->yellow
All other -> red
count = count +1
After clock terminates
End

Verilog Code

Initialization

```
module traffic_light(
                        north_light,south_light,east_light,w
est_light,clock,reset
   );
                        output reg [2:0]
north_light,south_light,east_light,west_light;
  input clock;
  input reset;
  parameter [2:0] north=3'b000;
  parameter [2:0] north_yellow=3'b001;
  parameter [2:0] south=3'b010;
  parameter [2:0] south_yellow=3'b011;
  parameter [2:0] east=3'b100;
  parameter [2:0] east_yellow=3'b101;
  parameter [2:0] west=3'b110;
  parameter [2:0] west_yellow=3'b111;
  reg [2:0] pre_state;
  reg [2:0] count;
```

Initial Check

```
always @(posedge clock, posedge reset)
    begin
      if (reset)
        begin            //async reset
          count =3'b000;
          pre_state=north;
        end
```

Verilog Code

Else cases:

```
begin
case (pre_state)
```

```
north :
        begin
          if (count==3'b111)
            begin
            count=3'b000;
            pre_state=north_yellow;
            end

        else
            begin
            count=count+3'b001;
            pre_state=north;
            end
        end
```

```
north_yellow :
        begin
          if (count==3'b011)
            begin
            count=3'b000;
            pre_state=south;
            end
          else
            begin
            count=count+3'b001;
            pre_state=north_yellow;
          end
        end
```

```
south :
        begin
          if (count==3'b111)
            begin
            count=3'b0;
            pre_state=south_yellow;
            end
          else
            begin
            count=count+3'b001;
            pre_state=south;
          end
        end
```

```
south_yellow :
        begin
          if (count==3'b011)
            begin
            count=3'b0;
            pre_state=east;
            end
          else
            begin
            count=count+3'b001;
            pre_state=south_yellow;
            end
        end
```

Verilog Code

Else cases:

```
east :
        begin
          if (count==3'b111)
            begin
            count=3'b0;
            pre_state=east_yellow;
            end
          else
            begin
            count=count+3'b001;
            pre_state=east;
            end
          end
```

```
east_yellow :
        begin
          if (count==3'b011)
            begin
            count=3'b0;
            pre_state=west;
            end
          else
            begin
            count=count+3'b001;
            pre_state=east_yellow;
            end
          end
```

```
west :
        begin
          if (count==3'b111)
            begin
            pre_state=west_yellow;
            count=3'b0;
            end
          else
            begin
            count=count+3'b001;
            pre_state=west;
            end
          end
```

```
endcase
        end
  end
```

```
west_yellow :
        begin
          if (count==3'b011)
            begin
            pre_state=north;
            count=3'b0;
            end
          else
            begin
            count=count+3'b001;
            pre_state=west_yellow;
            end
          end
```

Verilog Code For lights

```
always @(pre_state)
    begin
        case (pre_state)
```

- 001 -> green
- 010 -> yellow
- 100 -> red

```
north :
        begin
            north_light = 3'b001;
            south_light = 3'b100;
            east_light = 3'b100;
            west_light = 3'b100;
        end
```

```
north_yellow :
        begin
            north_light = 3'b010;
            south_light = 3'b100;
            east_light = 3'b100;
            west_light = 3'b100;
        end
```

```
south :
        begin
            north_light = 3'b100;
            south_light = 3'b001;
            east_light = 3'b100;
            west_light = 3'b100;
        end
```

```
south_yellow :
        begin
            north_light = 3'b100;
            south_light = 3'b010;
            east_light = 3'b100;
            west_light = 3'b100;
        end
```

Verilog Code For lights

```
west :
        begin
          north_light = 3'b100;
          south_light = 3'b100;
          east_light = 3'b100;
          west_light = 3'b001;
        end
```

```
west_yellow :
        begin
          north_light = 3'b100;
          south_light = 3'b100;
          east_light = 3'b100;
          west_light = 3'b010;
        end
```

```
east :
        begin
          north_light = 3'b100;
          south_light = 3'b100;
          east_light = 3'b001;
          west_light = 3'b100;
        end
```

```
east_yellow :
        begin
          north_light = 3'b100;
          south_light = 3'b100;
          east_light = 3'b010;
          west_light = 3'b100;
        end
```

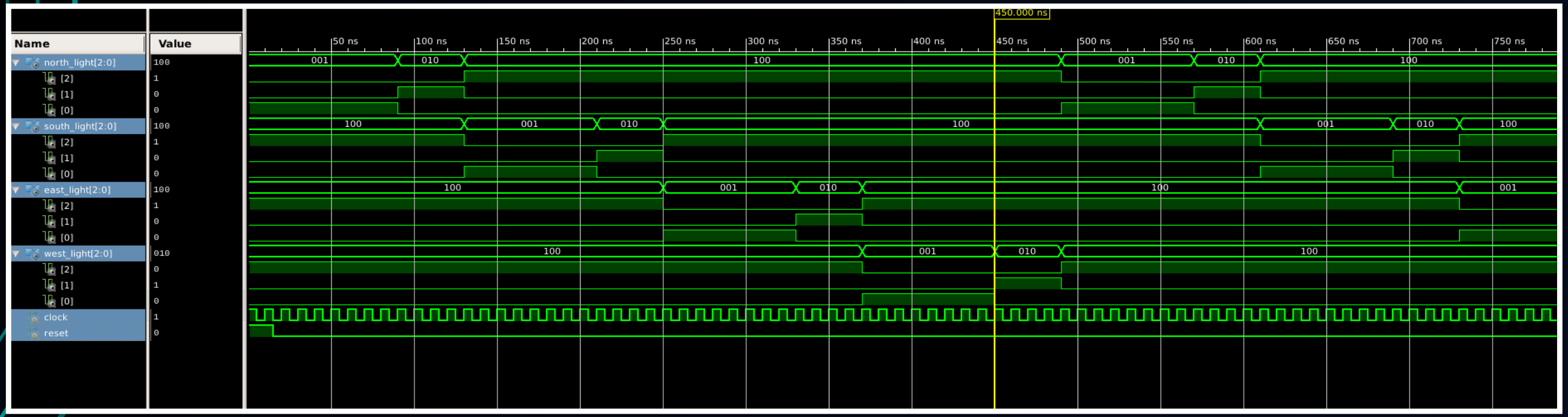
```
endcase

        end
endmodule
```

Test Bench

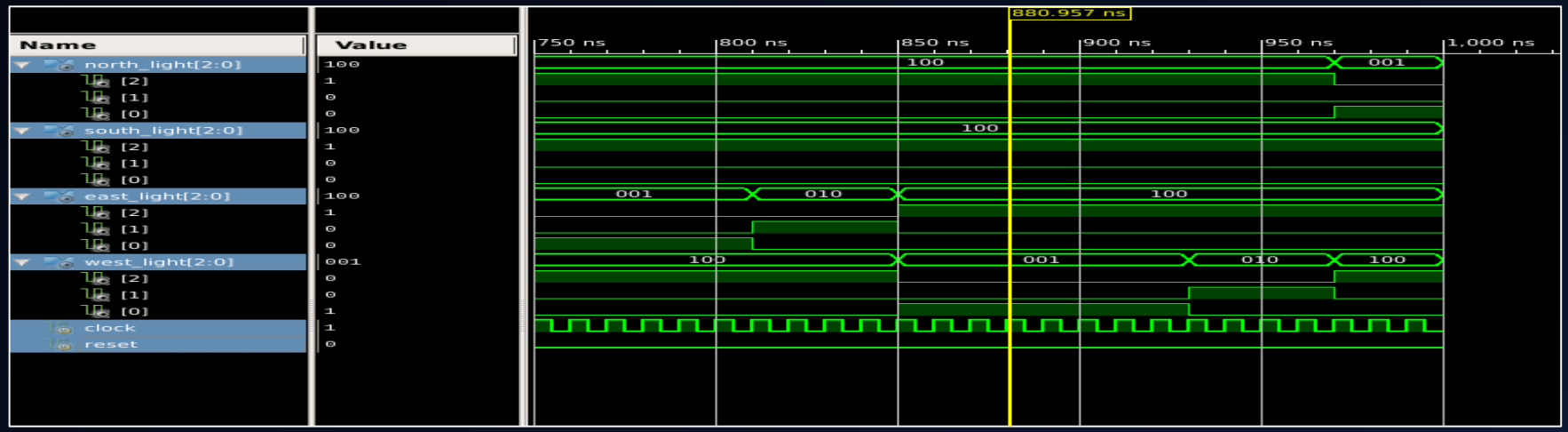
```
module testbench(
   );
                    reg clock,reset;
                    wire [2:0] north_light,south_light,east_light,west_light;

                   traffic_light T (north_light,south_light,east_light,west_light,clock,reset);

                   initial
                   begin
                   clock=1'b1;
                   forever #5 clock=~clock;
                   end

                   initial
                                    begin
                                    reset=1'b1;
                                    #15 reset=1'b0;
                                    end

                                    initial #1000;

endmodule
```

Simulation

0 ns to 750 ns



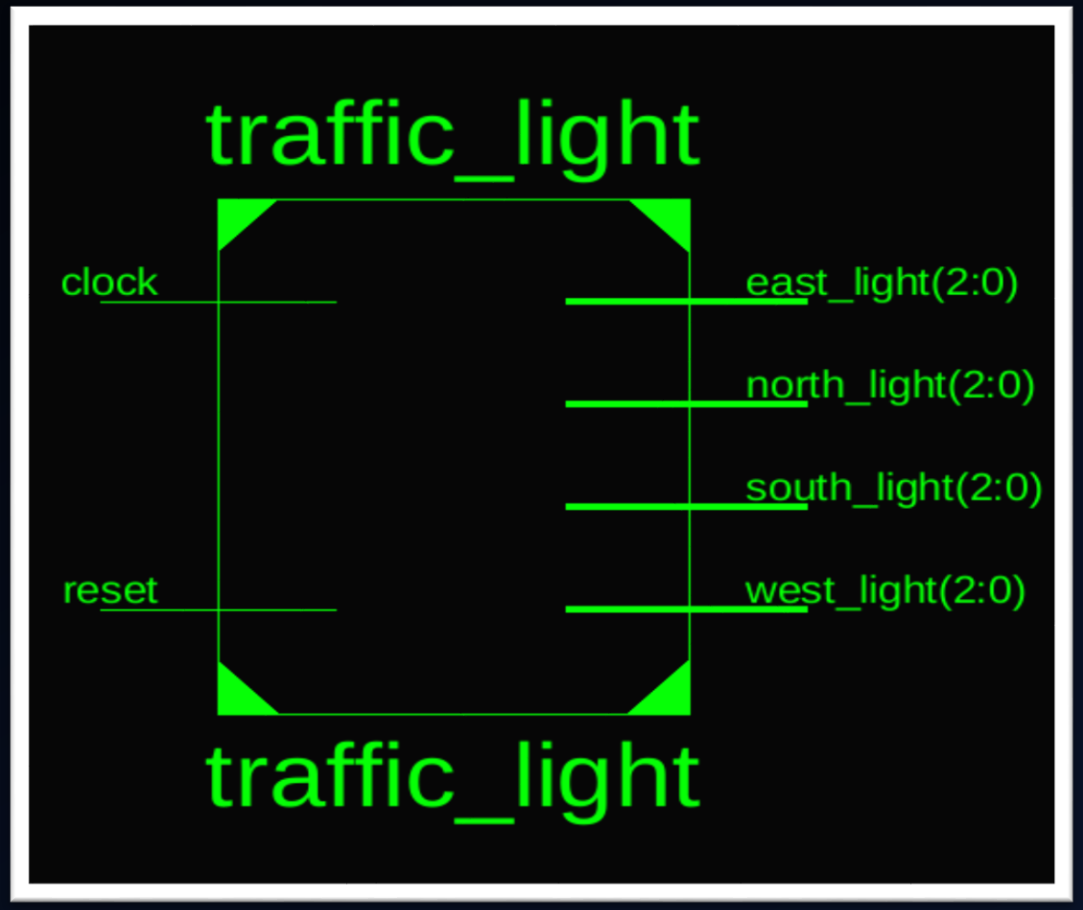
750 ns to 1000 ns



Explanation of Test Bench

- In the beginning, clock is initialized with 1 and set to flip after every 5 ns interval.
- Then for 15 ns, reset was set to 1, i.e. the current state should remain at north (green -> 001) for 15 ns which can be verified from the simulation.
- At last after 15 ns, reset was turned to 0 and from there, the traffic lights went functional which turned green for 8 clock cycle and then yellow for 4 clock cycles for every direction while the rest of them remained red.

Pin Structure of Overall Design



Comment on Hardware (with reference from RTL Schematic generated from the code)

- Everywhere where conditions are implemented in the code, it is basically handled by series of MUXs.
- All the states are handled by RTL ROMs and when to load which state is implemented again by 8:1 MUX (as there were total 8 states).
- After getting output from the mux, a D flip-flop is used to store it for the clock to come and send it to the light module to display the colour.
- The light module is also made up of 4 RTL ROMs for storing the code for lights to output it.
- Count is done by a adder module (could be a half-adder).

Conclusion

- A practical design of ‘one green at a time’ for traffic light was implemented successfully.
- On highways this design may appear a bit sluggish but in cities, this will work perfectly fine (due to relatively shrined roads and high crowd intensity).
- The working correctness is verified from the simulation.
- Learnt to solve a real life problem by utilizing the concepts and learning of the Digital Design course.

Thank You