PROJECT WORK

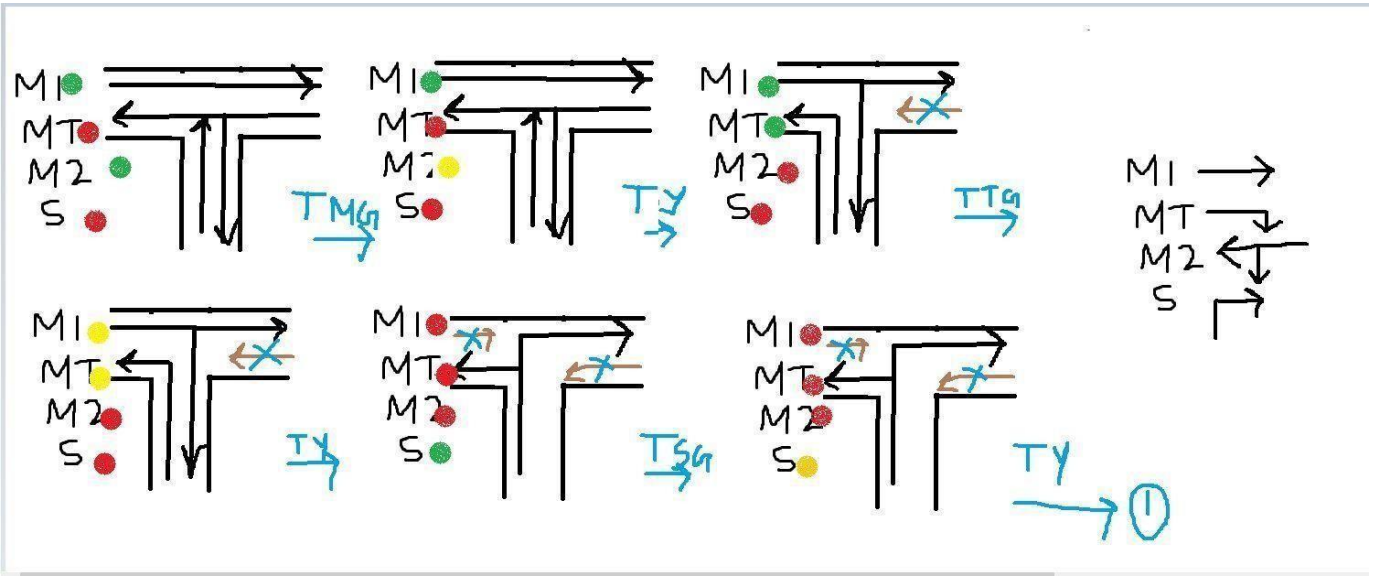
"Verilog Implementation of Digital Circuit Designs on FPGA using Vivado"

Topic: Traffic Light Controller Design using Verilog

PROBLEM STATEMENT

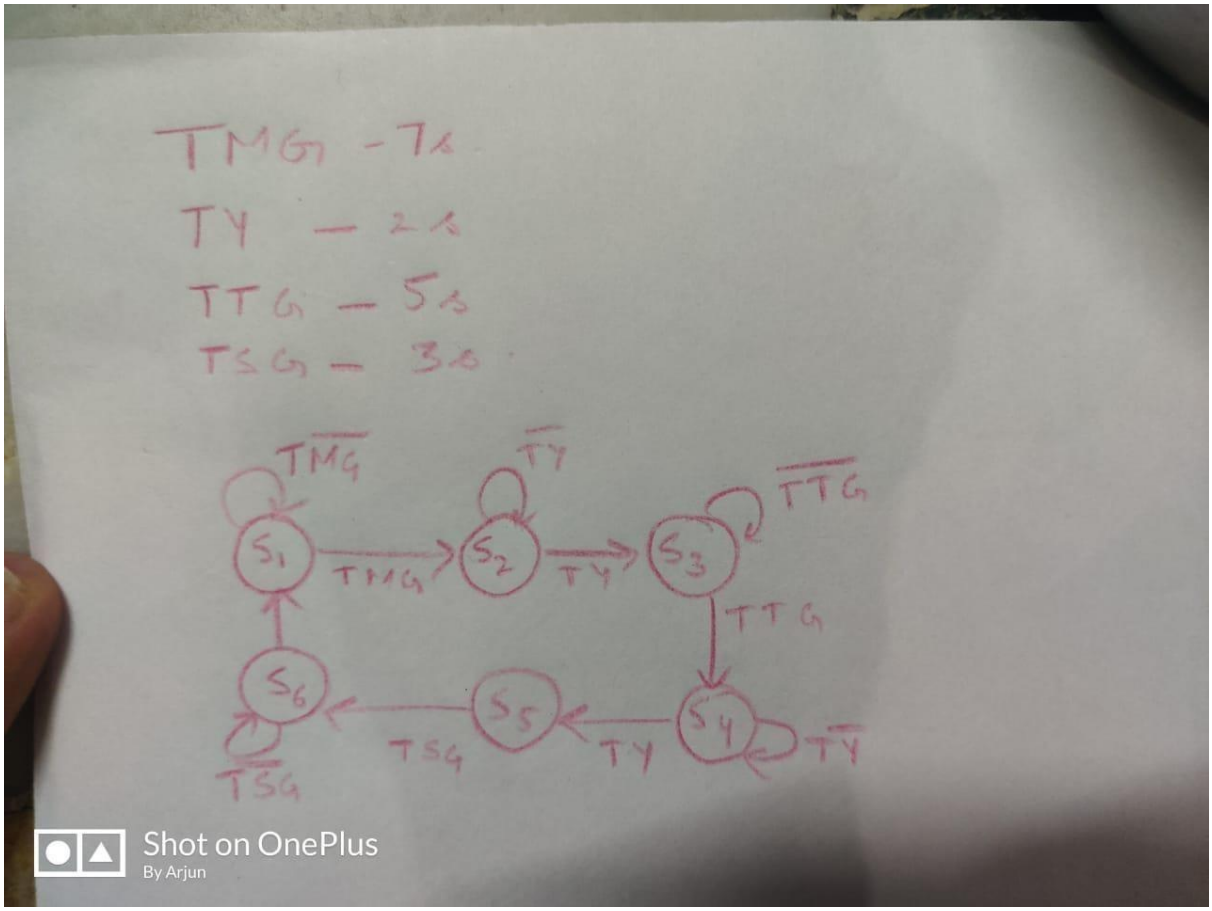
The aim of the project is to design a traffic controller for a T-intersection.

Let's understand the problem statement through the image given below.



The six cases present here eventually turn to the six states .

This is the state diagram:



From the state diagram we for the state table:

State Table

Present state A B C	Input	NS $A^+B^+C^+$		M1 R Y G	M2 R Y G	T R Y G	S R Y G
001	$\overline{TMG}$	001	}	001	001	100	100
001	TMG	010					
010	$\overline{TY}$	010	}	001	010	100	100
	TY	011					
011	$\overline{TTG}$	011	}	001	100	001	100
	TTG	100					
100	$\overline{TY}$	100	}	010	100	010	100
	TY	101					
101	$\overline{TSG}$	101	}	100	100	100	001
	TSG	110					
110	$\overline{TY}$	110	}	100	100	100	010
	TY	001					
111	-	000		000	000	000	000

VERILOG CODE

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Module Name: Traffic_Light_Controller // Project Name:

//////////////////////////////////////////////////////////////////////////////////



module Traffic_Light_Controller(



   input clk,rst,     output reg
[2:0]light_M1,     output reg
[2:0]light_S,     output reg
[2:0]light_MT,     output reg
[2:0]light_M2
   );

   parameter  S1=0, S2=1, S3 =2, S4=3, S5=4,S6=5;     reg
[3:0]count;     reg[2:0] ps;      parameter
sec7=7,sec5=5,sec2=2,sec3=3;



   always@(posedge clk or posedge rst)
      begin
if(rst==1)        begin
ps<=S1;
```

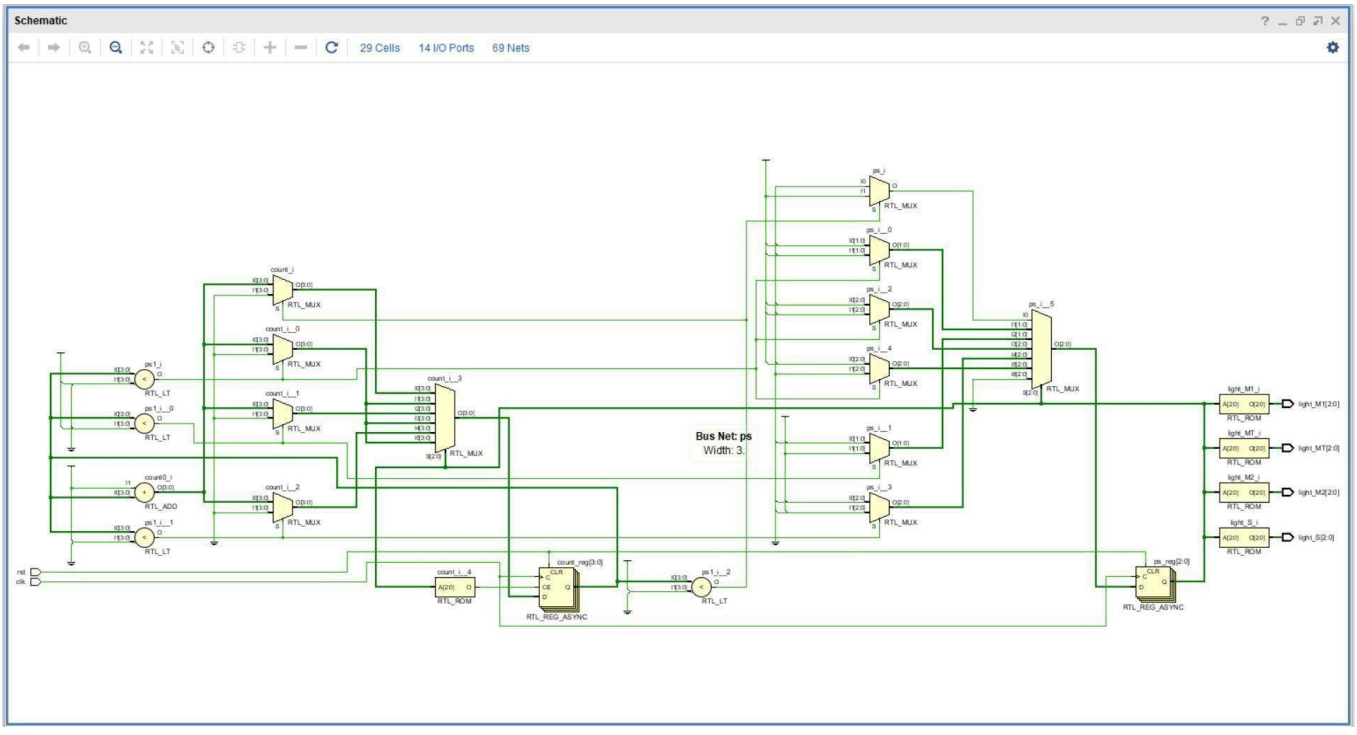
```
count<=0;        end
else


case(ps)              S1:
if(count<sec7)
begin                 ps<=S1;
count<=count+1;
end                   else
begin

                  ps<=S2;
count<=0;                 end
          S2: if(count<sec2)
begin                 ps<=S2;
count<=count+1;                 end

               else
begin                 ps<=S3;
count<=0;                 end
          S3: if(count<sec5)
begin                 ps<=S3;
count<=count+1;                 end
               else
begin                 ps<=S4;
count<=0;                 end
          S4:if(count<sec2)
begin                 ps<=S4;
count<=count+1;                 end
```

```
                else
begin                   ps<=S5;
count<=0;                       end
            S5:if(count<sec3)
begin                   ps<=S5;
count<=count+1;                     end
                else
begin                   ps<=S6;
count<=0;                       end

            S6:if(count<sec2)
begin                   ps<=S6;
count<=count+1;                     end
                else
begin
ps<=S1;
count<=0;
end             default:
ps<=S1;
endcase         end

always@(ps)
begin               case(ps)
                                S1:
begin                   light_M1<=3'b001;
light_M2<=3'b001;
light_MT<=3'b100;
light_S<=3'b100;                end
S2:                 begin
```

```
light_M1<=3'b001;
light_M2<=3'b010;
light_MT<=3'b100;
light_S<=3'b100;                end
S3:             begin
light_M1<=3'b001;
light_M2<=3'b100;
light_MT<=3'b001;
light_S<=3'b100;                end
S4:             begin
light_M1<=3'b010;
light_M2<=3'b100;
light_MT<=3'b010;
light_S<=3'b100;                end
S5:             begin
light_M1<=3'b100;
light_M2<=3'b100;
light_MT<=3'b100;
light_S<=3'b001;                end
S6:             begin
light_M1<=3'b100;
light_M2<=3'b100;
light_MT<=3'b100;
light_S<=3'b100;                end
default:                begin
light_M1<=3'b000;
light_M2<=3'b000;
light_MT<=3'b000;
light_S<=3'b010;                end
endcase         end
```

endmodule

RTL-SCHEMATIC

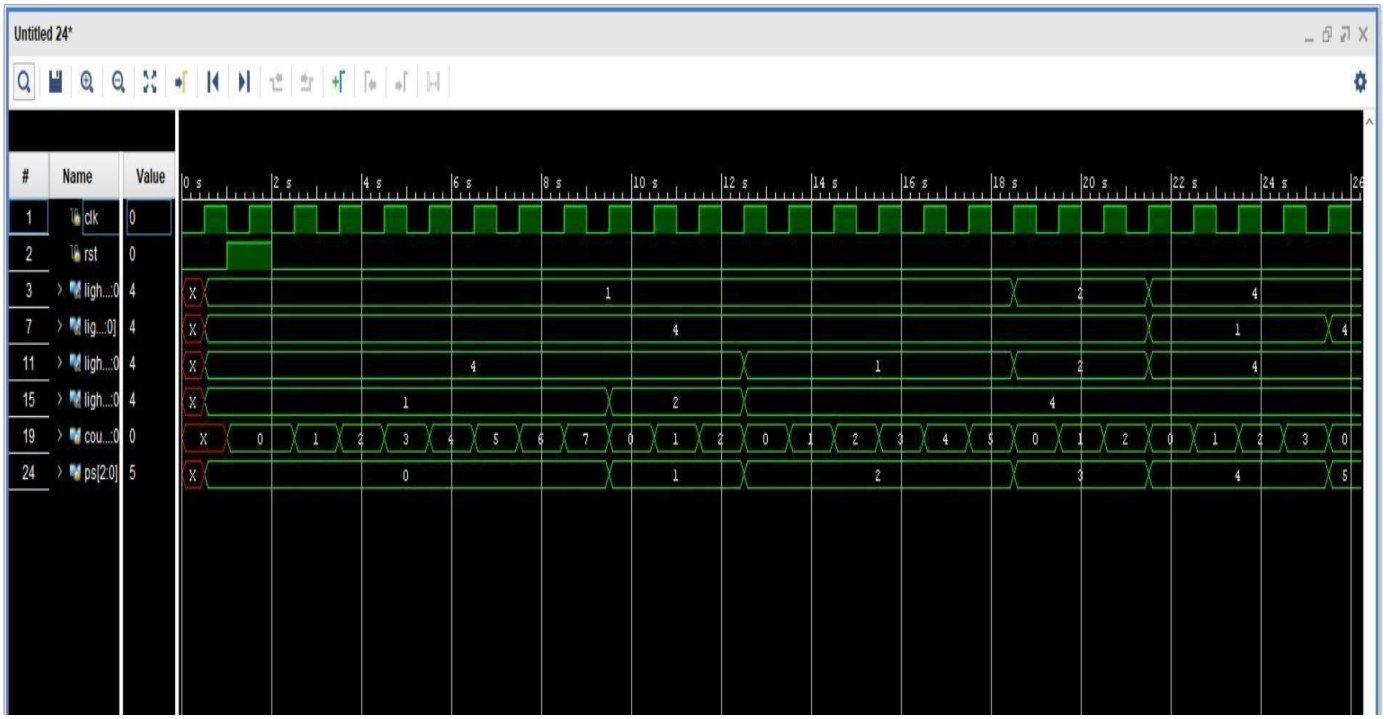


TESTBENCH

`timescale 1ns / 1ps

```
module Traffic_Light_Controller_TB;

reg clk,rst; wire [2:0]light_M1; wire

[2:0]light_S; wire [2:0]light_MT; wire

[2:0]light_M2;

Traffic_Light_Controller dut(.clk(clk) , .rst(rst) , .light_M1(light_M1) , .light_S(light_S)
,.light_M2(light_M2),.light_MT(light_MT)   );

 initial begin     clk=1'b0;      forever

#(1000000000/2) clk=~clk; end  initial begin      rst=0;

    #1000000000;     rst=1;

    #1000000000;     rst=0;

    #(1000000000*200);

    $finish;     end

endmodule
```

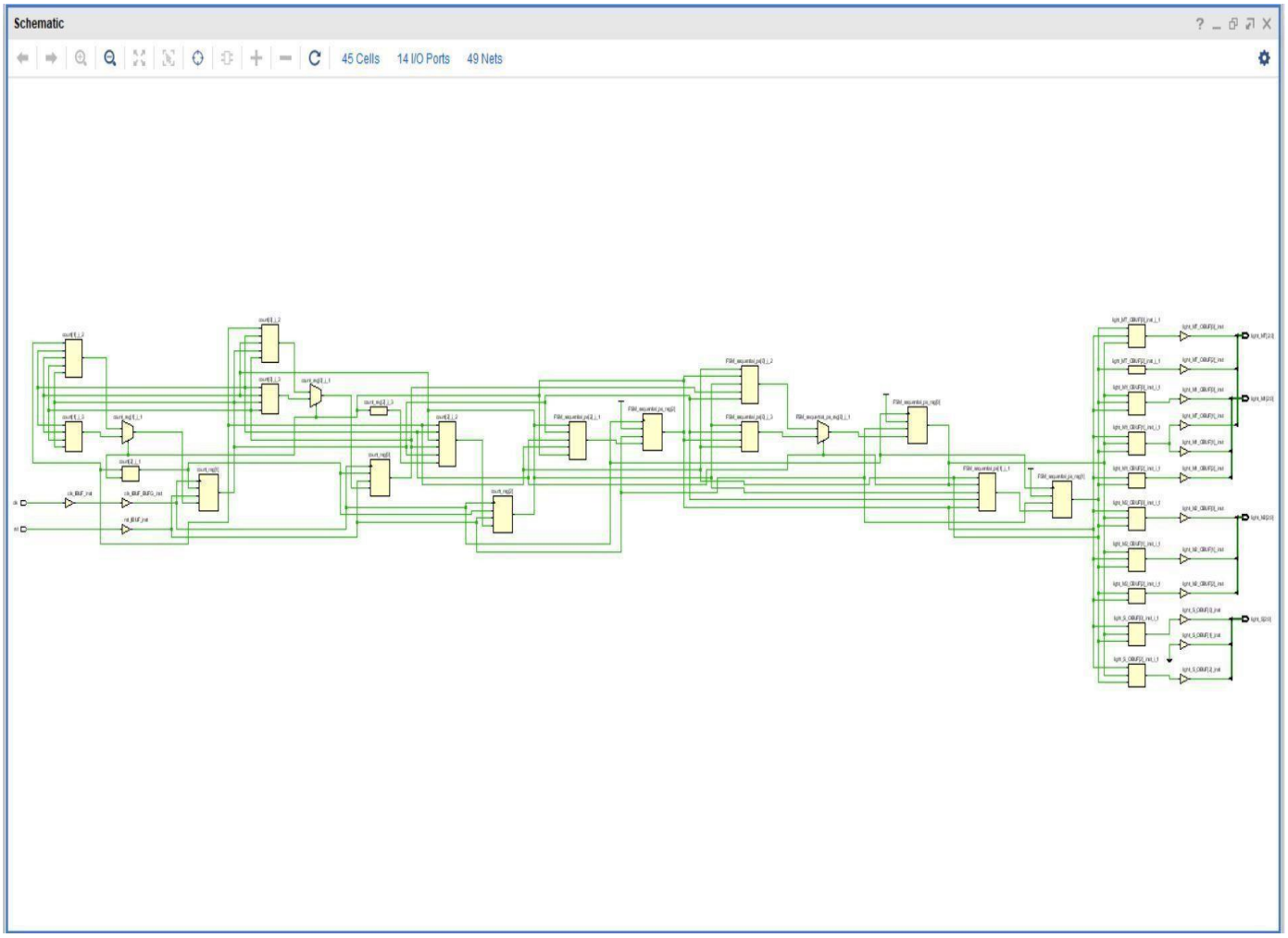
SIMULATED WAVEFORM



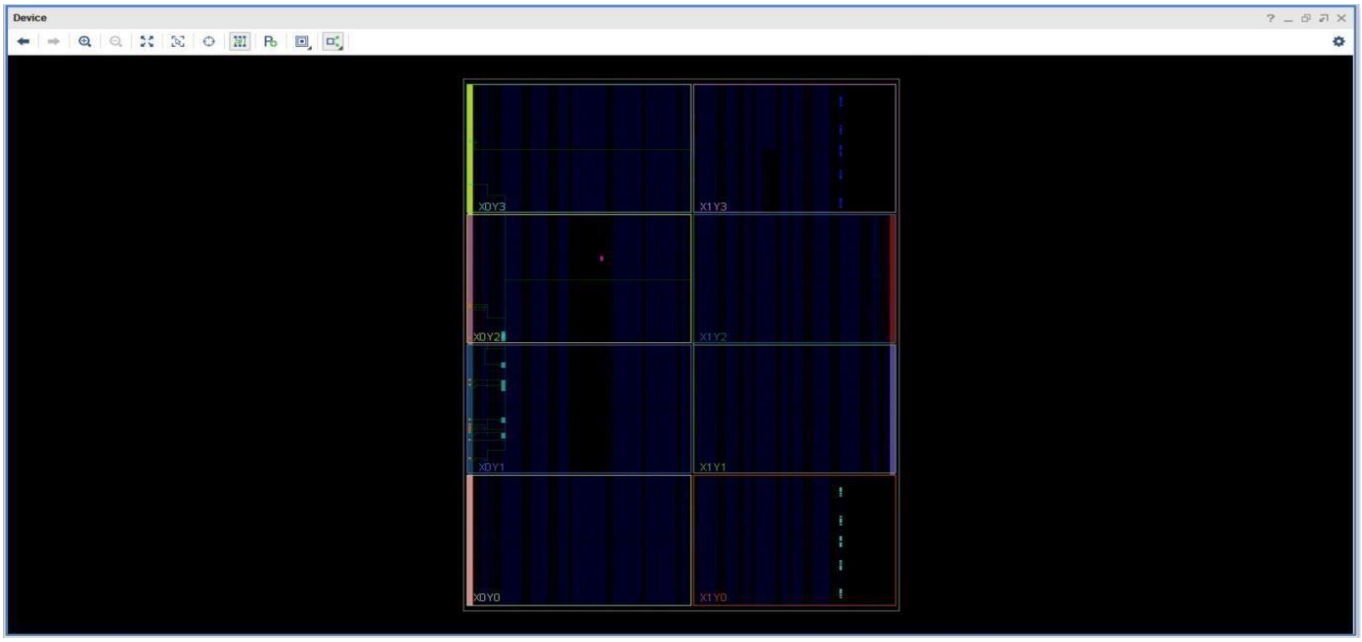
Upon analysing the waveform we can clearly see that the FSM works perfectly.

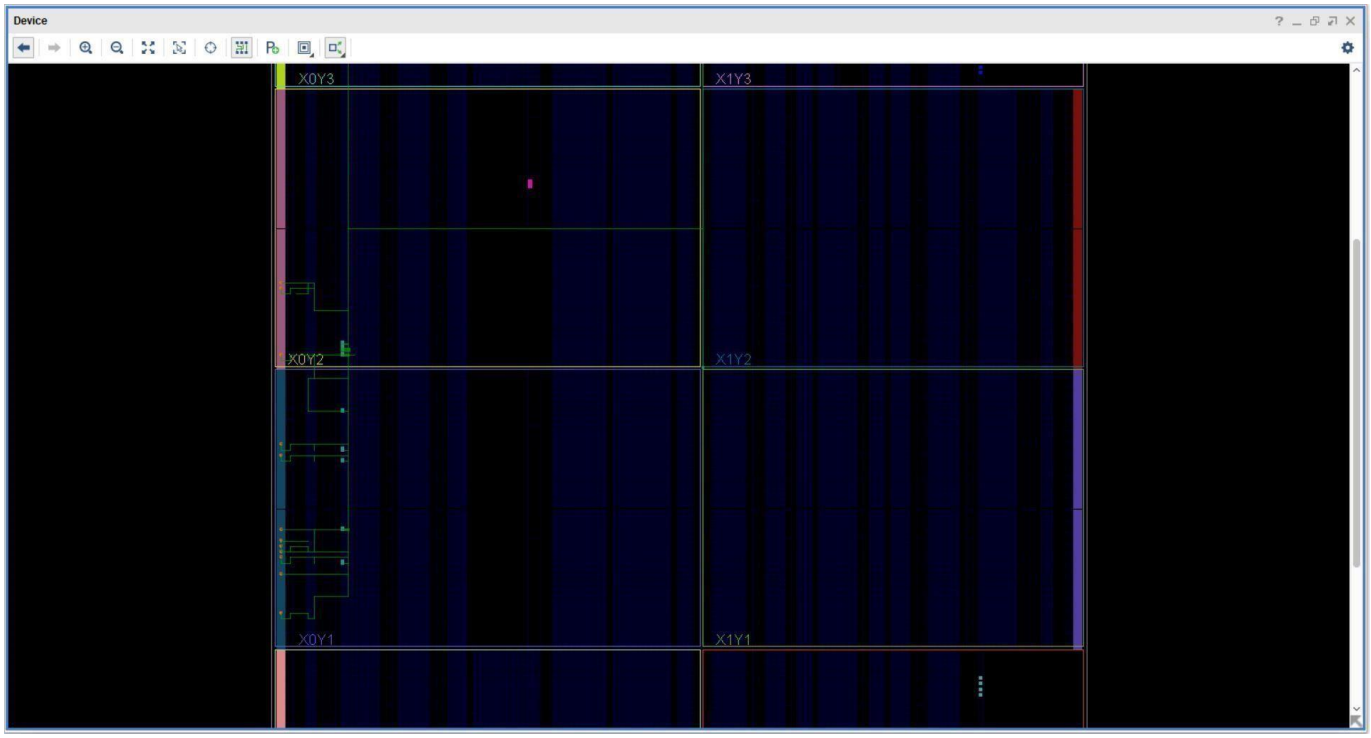
The ports are assigned from the ucf file .

SCHEMATIC AFTER SYNTHESIS



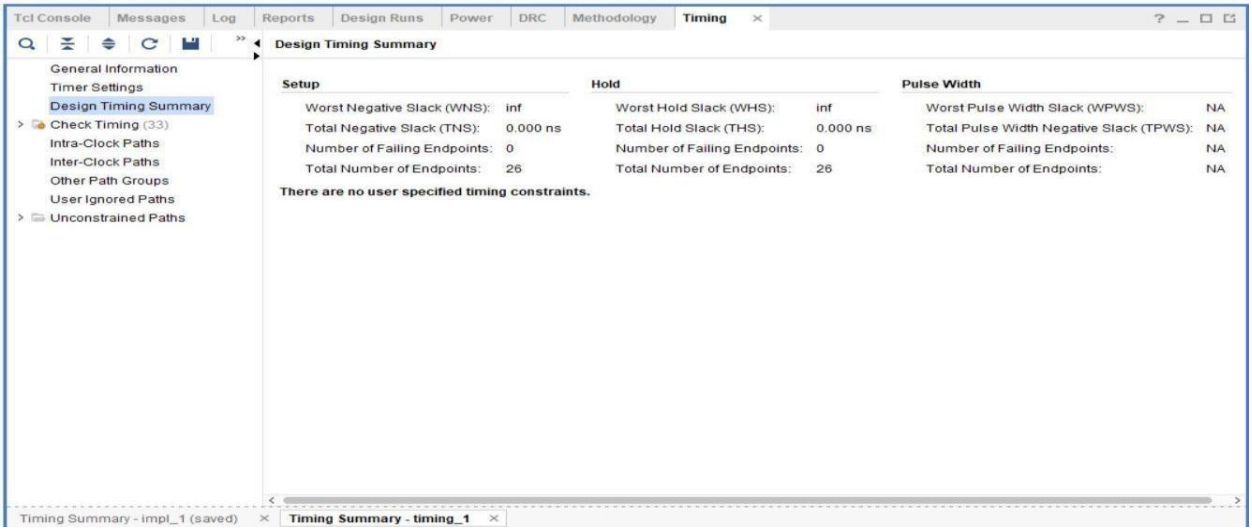
DEVICE LAYOUT AFTER IMPLEMENTATION

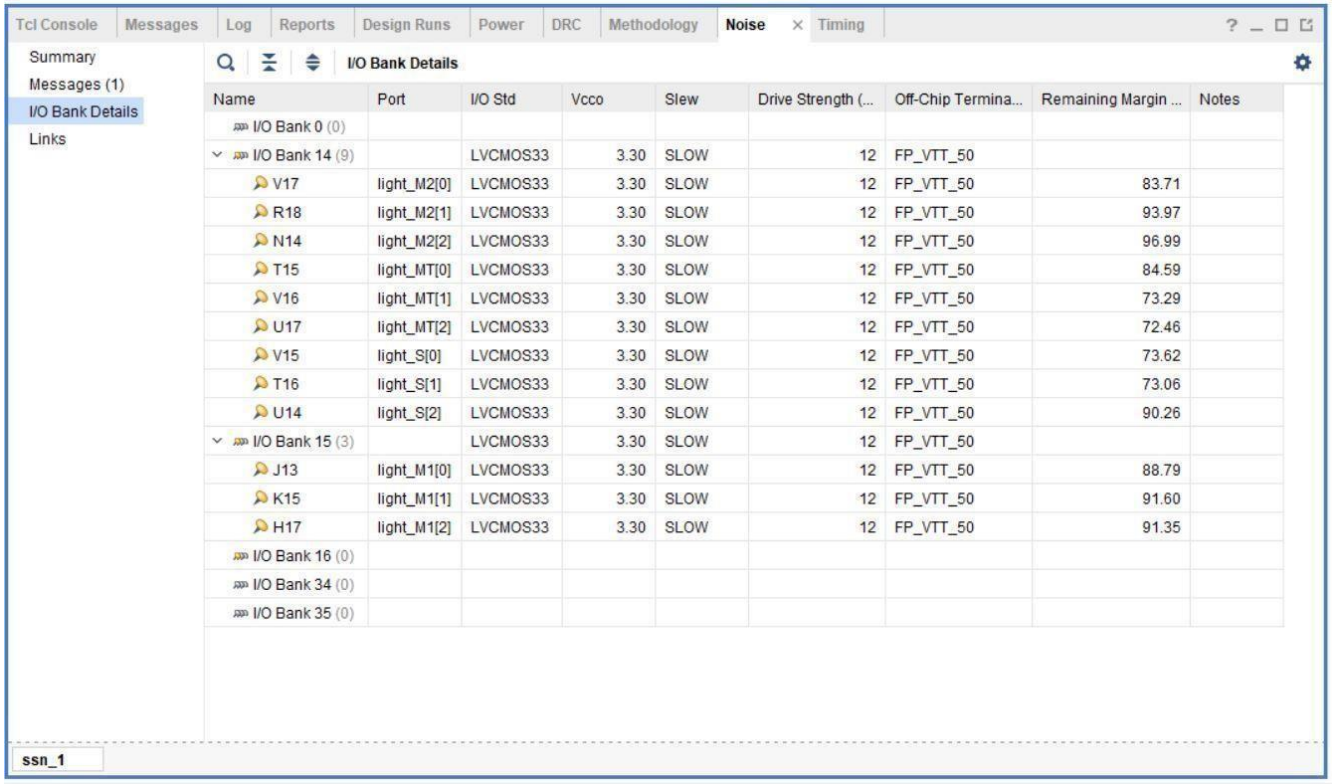




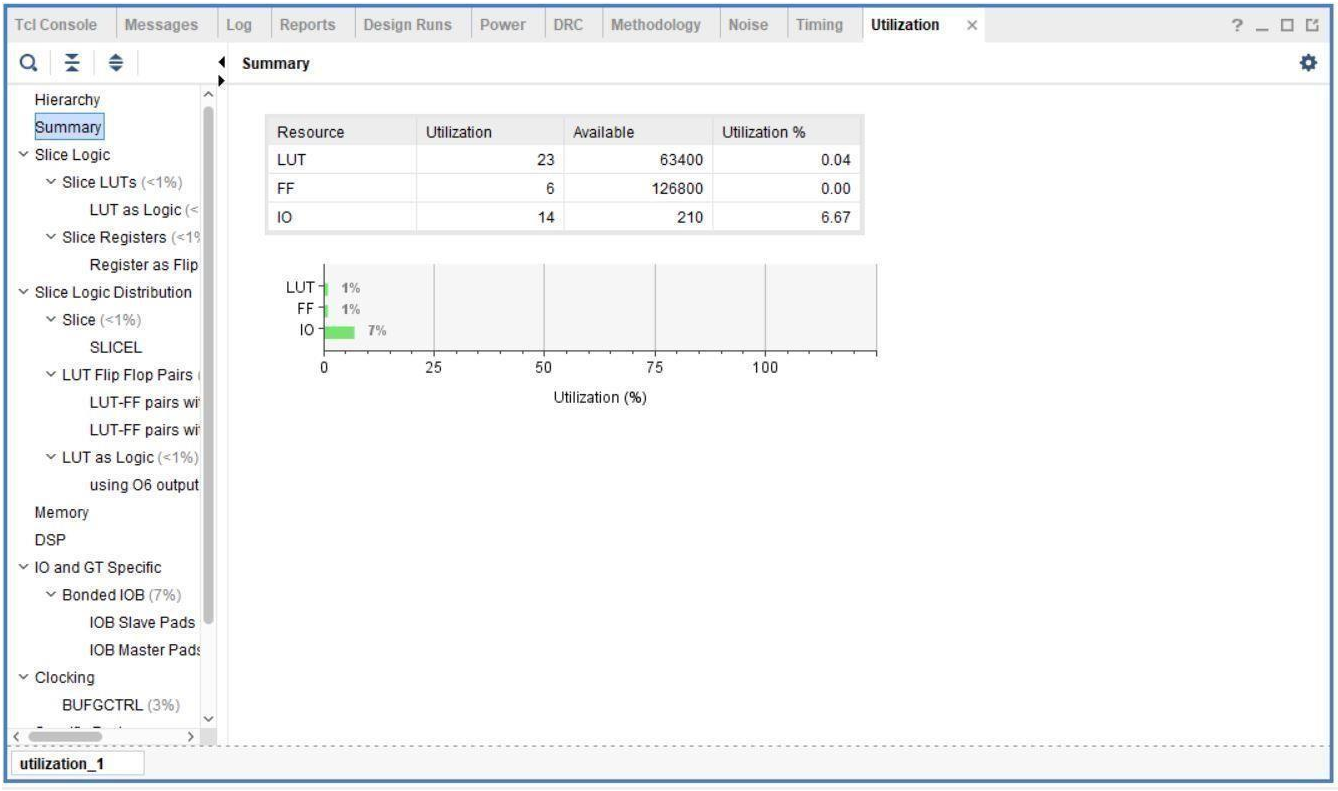
REPORTS AFTER SYNTHESIS

TIMING REPORT

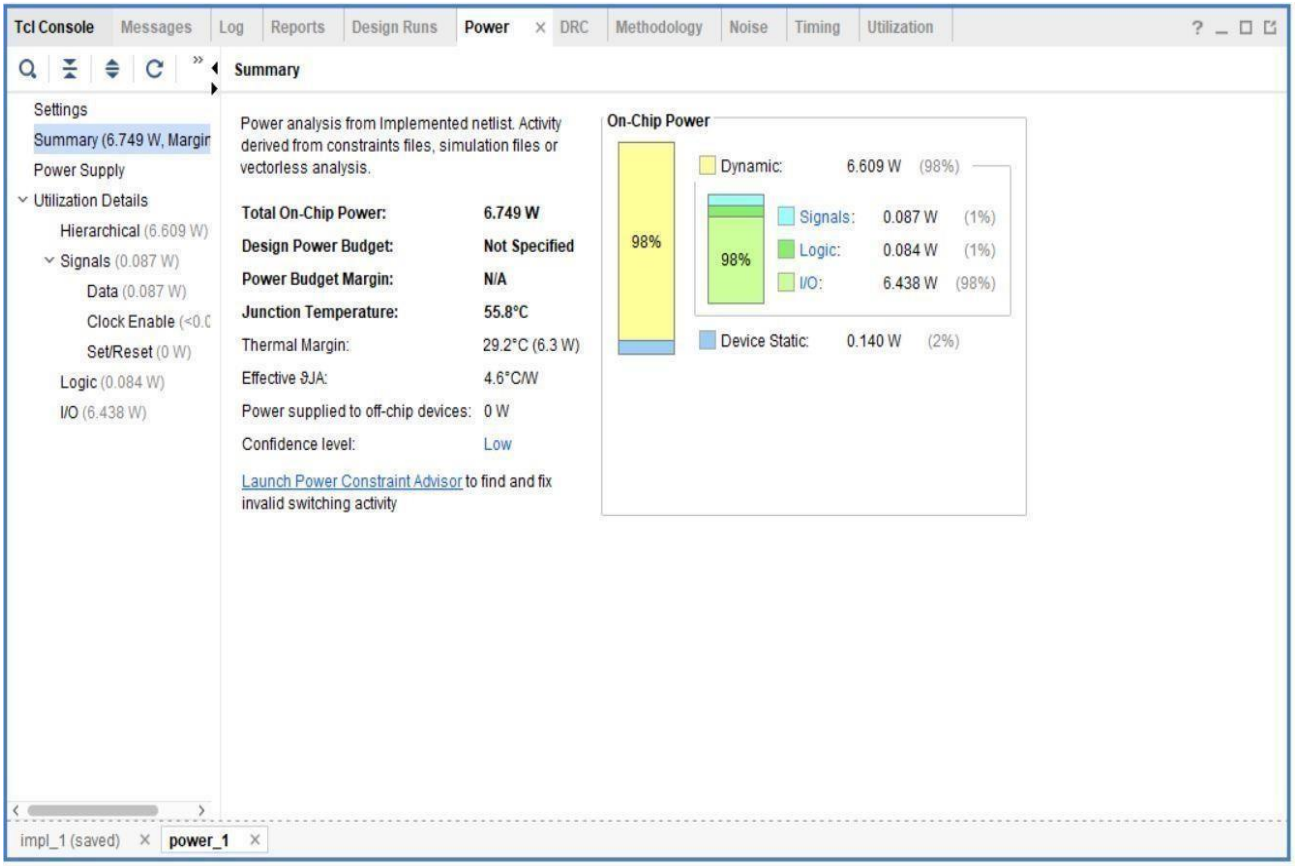




UTILIZATION REPORT



POWER REPORT



Github Link : https://github.com/Jidnyesh-Chaudhari-2004/Traffic-LightController-System-Verilog.git