4 lane automotive Traffic Light Controller (TLC) system design and verification using verilog

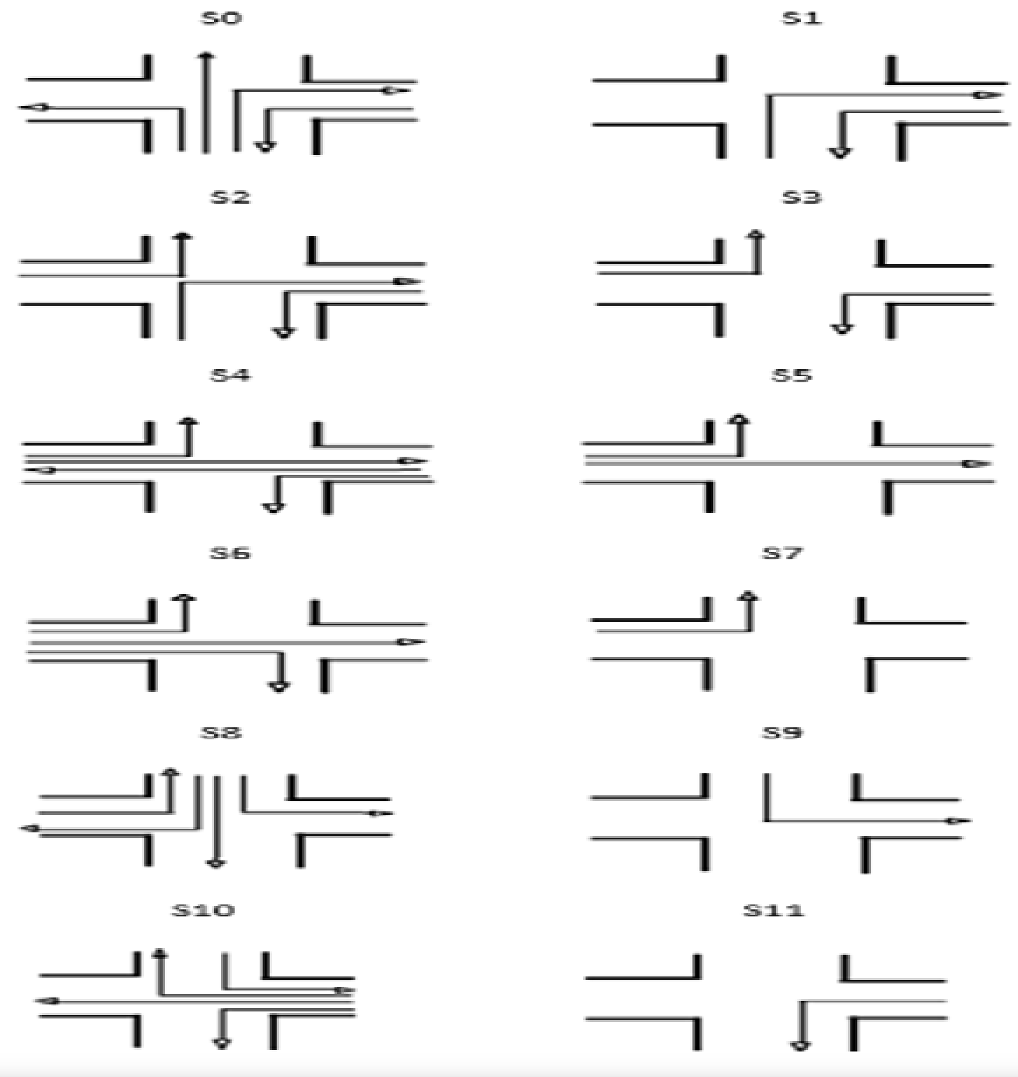
Name - Arpit Paul
Dept. - Electronics & Communication department
Year - 4th year
College - National Institute of Technology, Sikkim

Introduction

A traffic light controller system is a vital component of modern urban transportation infrastructure designed to manage and regulate the flow of vehicular and pedestrian traffic at intersections and road junctions. Its primary purpose is to enhance safety, minimize congestion, and optimize traffic flow by controlling the timing and sequencing of traffic signals.

Problem Statement

We have to design the TLC system for 4 lane junction, and for this problem the situation is as follows,



State Diagram

The state diagram for the above situation is as follows,

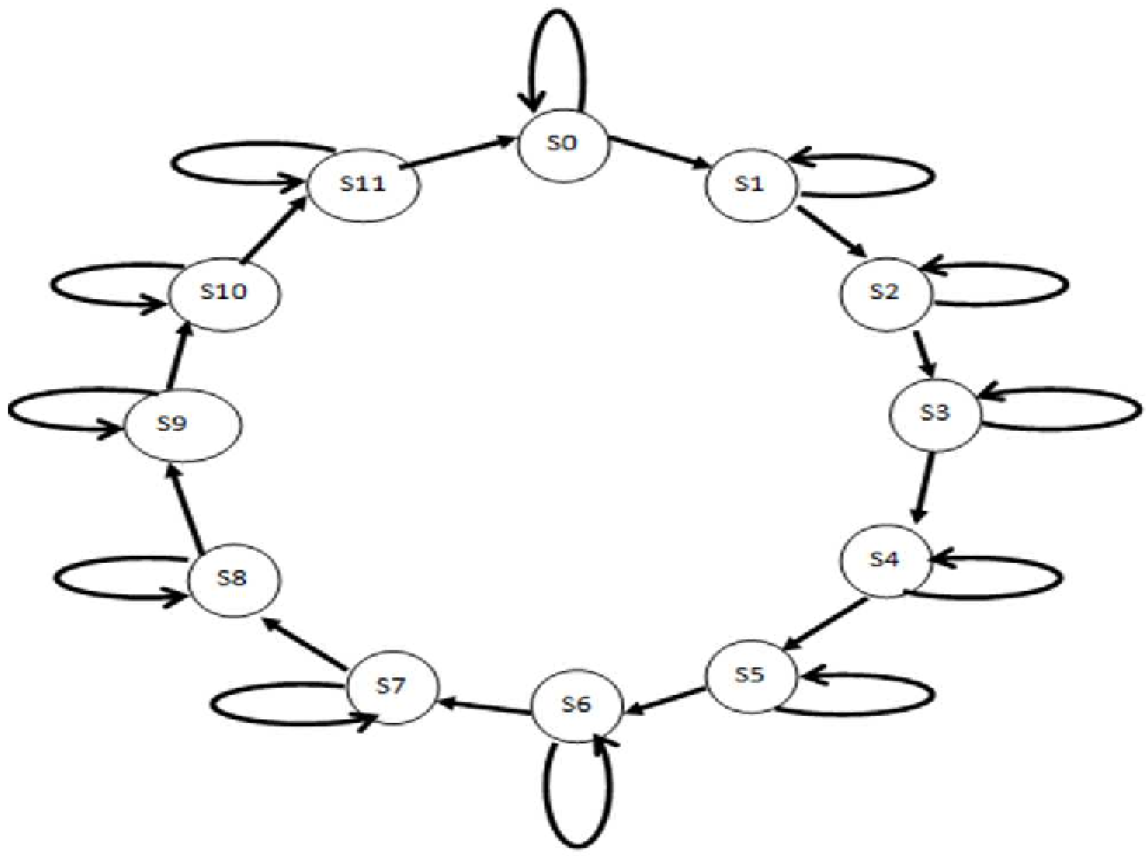
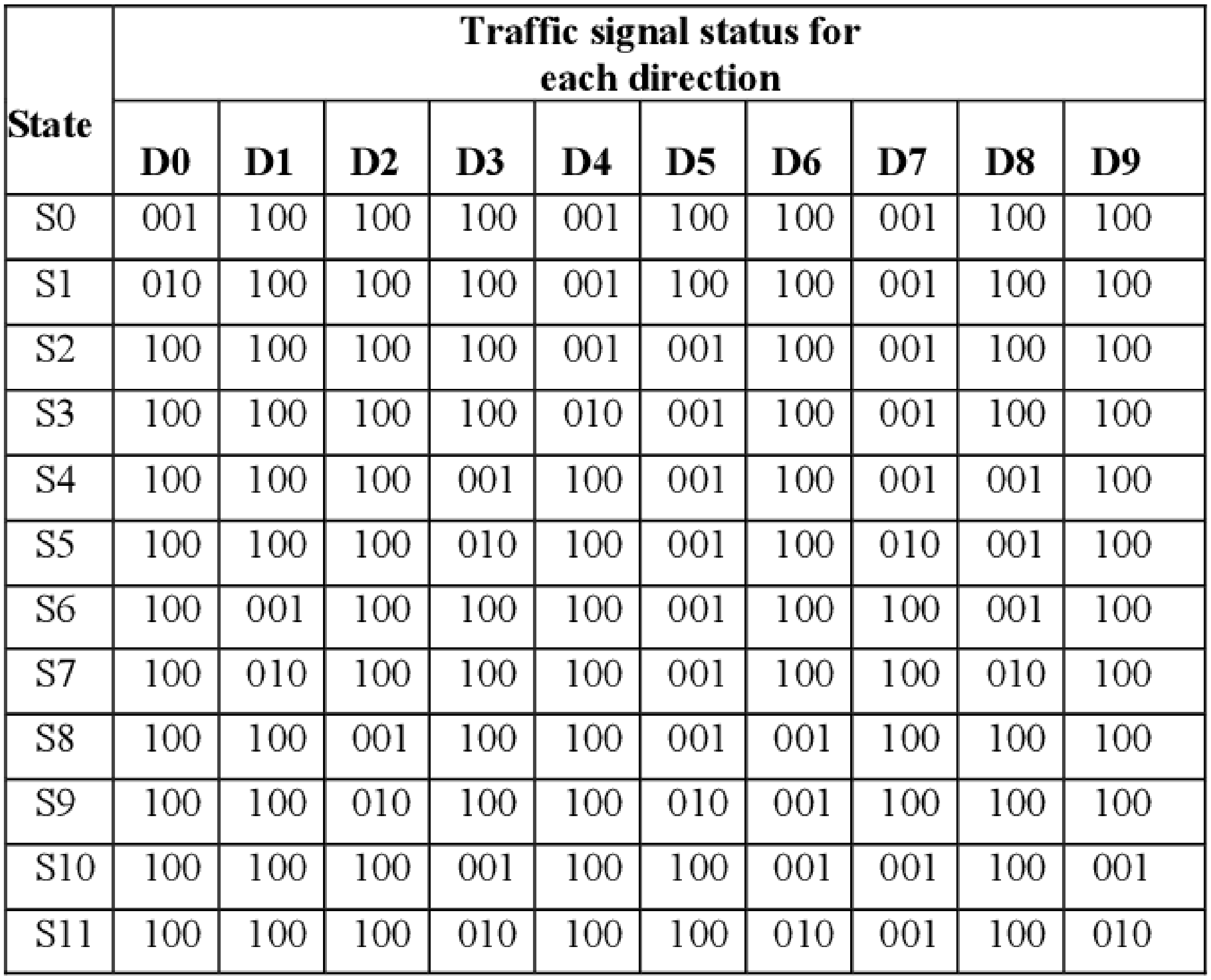


Fig.1 - State Diagram of TLC for 4 lane system

State diagram table

Traffic light indication table based on the state diagram is as follows,

D (Red, Yellow, Green)



State	Traffic signal status for each direction									
	D0	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
S0	001	100	100	100	001	100	100	001	100	100
S1	010	100	100	100	001	100	100	001	100	100
S2	100	100	100	100	001	001	100	001	100	100
S3	100	100	100	100	010	001	100	001	100	100
S4	100	100	100	001	100	001	100	001	001	100
S5	100	100	100	010	100	001	100	010	001	100
S6	100	001	100	100	100	001	100	100	001	100
S7	100	010	100	100	100	001	100	100	010	100
S8	100	100	001	100	100	001	001	100	100	100
S9	100	100	010	100	100	010	001	100	100	100
S10	100	100	100	001	100	100	001	001	100	001
S11	100	100	100	010	100	100	010	001	100	010

Fig. 2 - Characteristics table for state diagram

Verilog code (Design)

```
timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Mr. Arpit Paul
//
// Create Date: 25.09.2023 14:29:40
// Design Name:
// Module Name: TLC_design
// Project Name: 4 lane Traffic Light Controller
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module TLC_design( input rst, clk,
    output reg [2:0] tr_light0,
    output reg [2:0] tr_light1,
    output reg [2:0] tr_light2,
    output reg [2:0] tr_light3,
    output reg [2:0] tr_light4,
    output reg [2:0] tr_light5,
     output reg [2:0] tr_light6,
     output reg [2:0] tr_light7,
     output reg [2:0] tr_light8,
     output reg [2:0] tr_light9


    );


    parameter  s0=0, s1=1, s2=2, s3=3, s4=4, s5=5, s6=6, s7=7, s8=8, s9=9, s10=10,
s11=11;
    parameter t1=6, t2=3, t3=2, t4=5, t5=3, t6=3, t7=2, t8=2, t9=2,
t10=2,t11=3,t12=3;                   /////2-process methodology........
    reg [3:0] ps;
    reg[2:0] counter;

    always @(posedge clk or posedge rst)
    begin

      if(rst==1)
      begin
        ps <= s0;
        counter<=0;
      end
      else
                                            ////////////Reset Logic and Next State Decoder.........
          case(ps)
             s0: begin
                  if(counter<t1)
                    begin
                      ps <= s0;
                      counter <= counter+1;
                    end
                    else
                      begin
                        ps <= s1;
                        counter<=0;
                      end
                    end
              s1: begin
                      if(counter<t2)
                        begin
                         ps <= s1;
                         counter <= counter+1;
                        end

                        else
                          begin
                            ps <= s2;
                            counter<=0;
                          end
                 end

                s2: begin
                         if(counter<t3)
                           begin
                            ps <= s2;
                            counter <= counter+1;
                           end

                           else
                             begin
                               ps <= s3;
                               counter<=0;
                             end
```

```
 end

s3: begin
         if(counter<t4)
           begin
            ps <= s3;
            counter <= counter+1;
           end

           else
             begin
               ps <= s4;
               counter<=0;
             end
    end
 s4: begin
            if(counter<t5)
              begin
               ps <= s4;
               counter <= counter+1;
              end

              else
                begin
                  ps <= s5;
                  counter<=0;
                end
       end

  s5: begin
               if(counter<t6)
                 begin
                  ps <= s5;
                counter <= counter+1;
               end

               else
                 begin
                   ps <= s6;
                   counter<=0;
                 end
        end
s6: begin
              if(counter<t7)
                  begin
                   ps <= s6;
                   counter <= counter+1;
                  end

                  else
                    begin
                      ps <= s7;
                      counter<=0;
                    end
     end

 s7: begin
             if(counter<t8)
               begin
                ps <= s7;
                counter <= counter+1;
               end

               else
                 begin
                   ps <= s8;
                 counter<=0;
               end
      end
s8: begin
             if(counter<t9)
               begin
                ps <= s8;
                counter <= counter+1;
               end

               else
                 begin
                   ps <= s9;
                   counter<=0;
                 end
      end
 s9: begin
             if(counter<t10)
               begin
                ps <= s9;
                counter <= counter+1;
               end

               else
                 begin
                   ps <= s10;
                   counter<=0;
                 end
     end
  s10: begin
            if(counter<t11)
              begin
               ps <= s10;
```

```
                     counter <= counter+1;
                    end

                    else
                      begin
                        ps <= s11;
                        counter<=0;
                      end
             end
      s11: begin
                     if(counter<t12)
                       begin
                        ps <= s11;
                        counter <= counter+1;
                       end

                       else
                         begin
                           ps <= s0;
                           counter<=0;
                         end
           end

     default: ps <= s0;
   endcase



 end

always @ (ps)
begin
```

```
case(ps)

    s0:begin
      tr_light0 <= 3'b001;
      tr_light1 <= 3'b100;
      tr_light2 <= 3'b100;
      tr_light3 <= 3'b100;
      tr_light4 <= 3'b001;
      tr_light5 <= 3'b100;
      tr_light6 <= 3'b100;
      tr_light7 <= 3'b001;
      tr_light8 <= 3'b100;
      tr_light9 <= 3'b100;
     end
    s1:begin
         tr_light0 <= 3'b010;
         tr_light1 <= 3'b100;
         tr_light2 <= 3'b100;
         tr_light3 <= 3'b100;                              /////////Output Decoder................
         tr_light4 <= 3'b001;
         tr_light5 <= 3'b100;
         tr_light6 <= 3'b100;
         tr_light7 <= 3'b001;
         tr_light8 <= 3'b100;
         tr_light9 <= 3'b100;
        end
        s2:begin
            tr_light0 <= 3'b100;
            tr_light1 <= 3'b100;
            tr_light2 <= 3'b100;
            tr_light3 <= 3'b100;
            tr_light4 <= 3'b001;
            tr_light5 <= 3'b001;
```

```
 tr_light6 <= 3'b100;
 tr_light7 <= 3'b001;
 tr_light8 <= 3'b100;
 tr_light9 <= 3'b100;
end
s3:begin
    tr_light0 <= 3'b100;
    tr_light1 <= 3'b100;
    tr_light2 <= 3'b100;
    tr_light3 <= 3'b100;
    tr_light4 <= 3'b010;
    tr_light5 <= 3'b001;
    tr_light6 <= 3'b100;
    tr_light7 <= 3'b001;
    tr_light8 <= 3'b100;
    tr_light9 <= 3'b100;
   end
 s4:begin
       tr_light0 <= 3'b100;
       tr_light1 <= 3'b100;
       tr_light2 <= 3'b100;
       tr_light3 <= 3'b001;
       tr_light4 <= 3'b100;
       tr_light5 <= 3'b001;
       tr_light6 <= 3'b100;
       tr_light7 <= 3'b001;
       tr_light8 <= 3'b001;
       tr_light9 <= 3'b100;
      end
   s5:begin
          tr_light0 <= 3'b100;
          tr_light1 <= 3'b100;
          tr_light2 <= 3'b100;
      tr_light3 <= 3'b010;
      tr_light4 <= 3'b100;
      tr_light5 <= 3'b001;
      tr_light6 <= 3'b100;
      tr_light7 <= 3'b010;
      tr_light8 <= 3'b001;
      tr_light9 <= 3'b100;
     end
s6:begin
        tr_light0 <= 3'b100;
        tr_light1 <= 3'b001;
        tr_light2 <= 3'b100;
        tr_light3 <= 3'b100;
        tr_light4 <= 3'b100;
        tr_light5 <= 3'b001;
        tr_light6 <= 3'b100;
        tr_light7 <= 3'b100;
        tr_light8 <= 3'b001;
        tr_light9 <= 3'b100;
       end
     s7:begin
           tr_light0 <= 3'b100;
           tr_light1 <= 3'b010;
           tr_light2 <= 3'b100;
           tr_light3 <= 3'b100;
           tr_light4 <= 3'b100;
           tr_light5 <= 3'b001;
           tr_light6 <= 3'b100;
           tr_light7 <= 3'b100;
           tr_light8 <= 3'b010;
           tr_light9 <= 3'b100;
          end
        s8:begin
```

```
    tr_light0 <= 3'b100;
    tr_light1 <= 3'b100;
    tr_light2 <= 3'b001;
    tr_light3 <= 3'b100;
    tr_light4 <= 3'b100;
    tr_light5 <= 3'b001;
    tr_light6 <= 3'b001;
    tr_light7 <= 3'b100;
    tr_light8 <= 3'b100;
    tr_light9 <= 3'b100;
   end
s9:begin
       tr_light0 <= 3'b100;
       tr_light1 <= 3'b100;
       tr_light2 <= 3'b010;
       tr_light3 <= 3'b100;
       tr_light4 <= 3'b100;
       tr_light5 <= 3'b010;
       tr_light6 <= 3'b001;
       tr_light7 <= 3'b100;
       tr_light8 <= 3'b100;
       tr_light9 <= 3'b100;
      end
  s10:begin
          tr_light0 <= 3'b100;
          tr_light1 <= 3'b100;
          tr_light2 <= 3'b100;
          tr_light3 <= 3'b001;
          tr_light4 <= 3'b100;
          tr_light5 <= 3'b100;
          tr_light6 <= 3'b001;
          tr_light7 <= 3'b001;
          tr_light8 <= 3'b100;
```

```
                                          tr_light9 <= 3'b001;
                                         end
                                     s11:begin
                                            tr_light0 <= 3'b100;
                                            tr_light1 <= 3'b100;
                                            tr_light2 <= 3'b100;
                                            tr_light3 <= 3'b010;
                                            tr_light4 <= 3'b100;
                                            tr_light5 <= 3'b100;
                                            tr_light6 <= 3'b010;
                                            tr_light7 <= 3'b001;
                                            tr_light8 <= 3'b100;
                                            tr_light9 <= 3'b010;
                                        end
                              default: begin
                                   tr_light0 <= 3'b000;
                                   tr_light1 <= 3'b000;
                                   tr_light2 <= 3'b000;
                                   tr_light3 <= 3'b000;
                                   tr_light4 <= 3'b000;
                                   tr_light5 <= 3'b000;
                                   tr_light6 <= 3'b000;
                                   tr_light7 <= 3'b000;
                                   tr_light8 <= 3'b000;
                                   tr_light9 <= 3'b000;
                                  end
   endcase


   end
endmodule
```

Verilog Code (Testbench)

```
timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 25.09.2023 16:22:52
// Design Name:
// Module Name: tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module tb();

        reg clk, rst;
        wire [2:0] tr_light0;
        wire [2:0] tr_light1;
        wire [2:0] tr_light2;
        wire [2:0] tr_light3;
        wire [2:0] tr_light4;
```

```
      wire [2:0] tr_light5;
      wire [2:0] tr_light6;
     wire [2:0] tr_light7;
      wire [2:0] tr_light8;
      wire [2:0] tr_light9;


  TLC_design dut(.rst(rst), .clk(clk), .tr_light0(tr_light0), .tr_light1(tr_light1),
.tr_light2(tr_light2) ,.tr_light3(tr_light3), .tr_light4(tr_light4), .tr_light5(tr_light5),
.tr_light6(tr_light6), .tr_light7(tr_light7), .tr_light8(tr_light8), .tr_light9(tr_light9));

    initial begin

     clk = 1'b0;

      forever #10 clk = ~clk;

    end

     initial begin

      rst=1'b0;
      #10;
      rst=1'b1;
      #20;
      rst=1'b0;

      #(20*100);
      $finish;


    end
   endmodule
```

Block Diagram and RTL Design

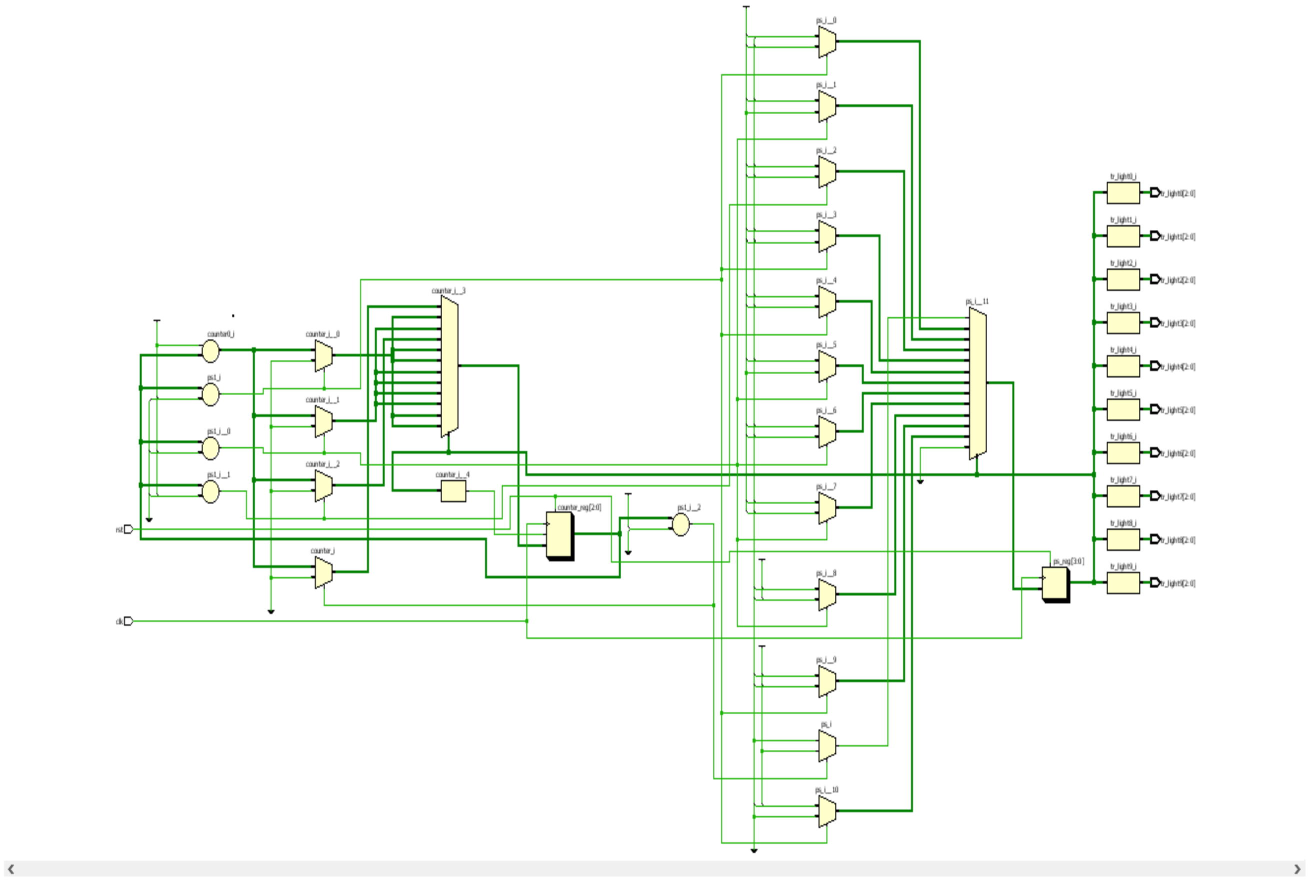


Fig. 3 - RTL Design for 4 lane TLC

Timing Diagram for verification

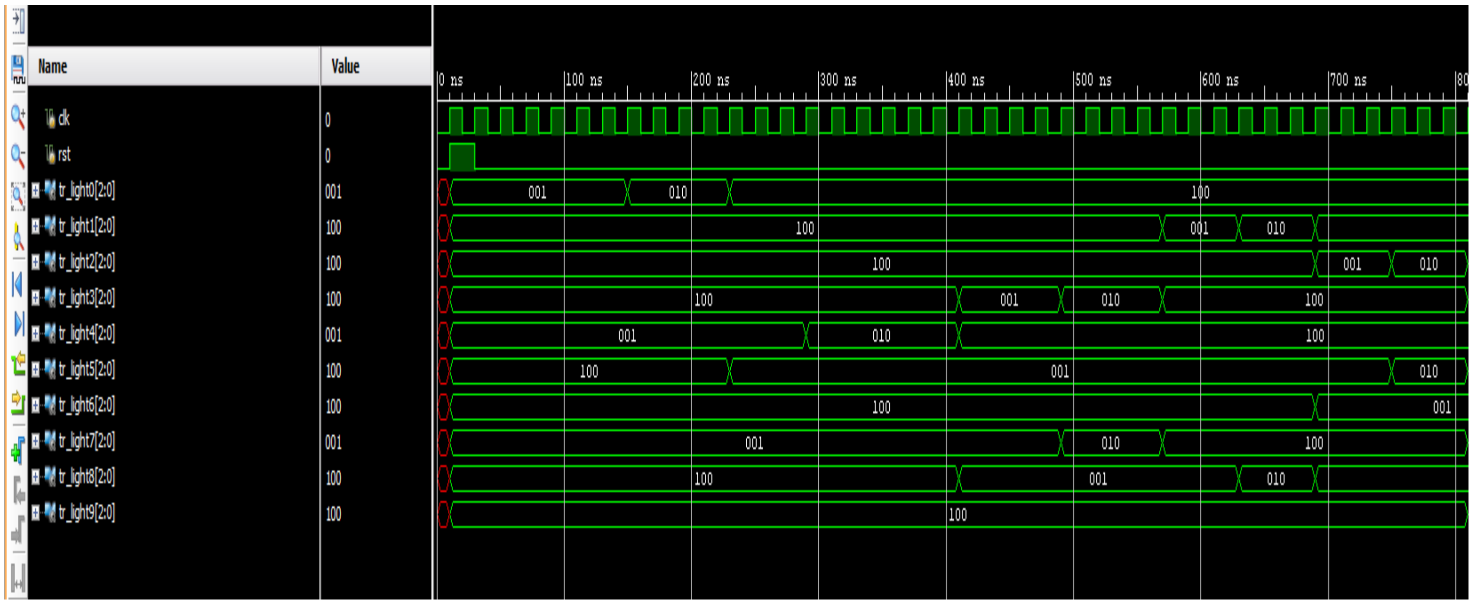


Fig. 4 - Timing diagram for verification

Synthesized Design

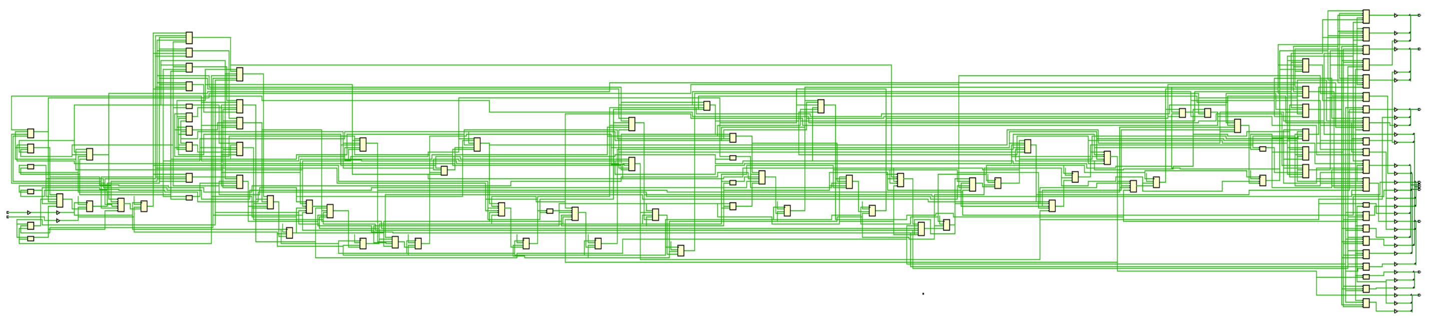
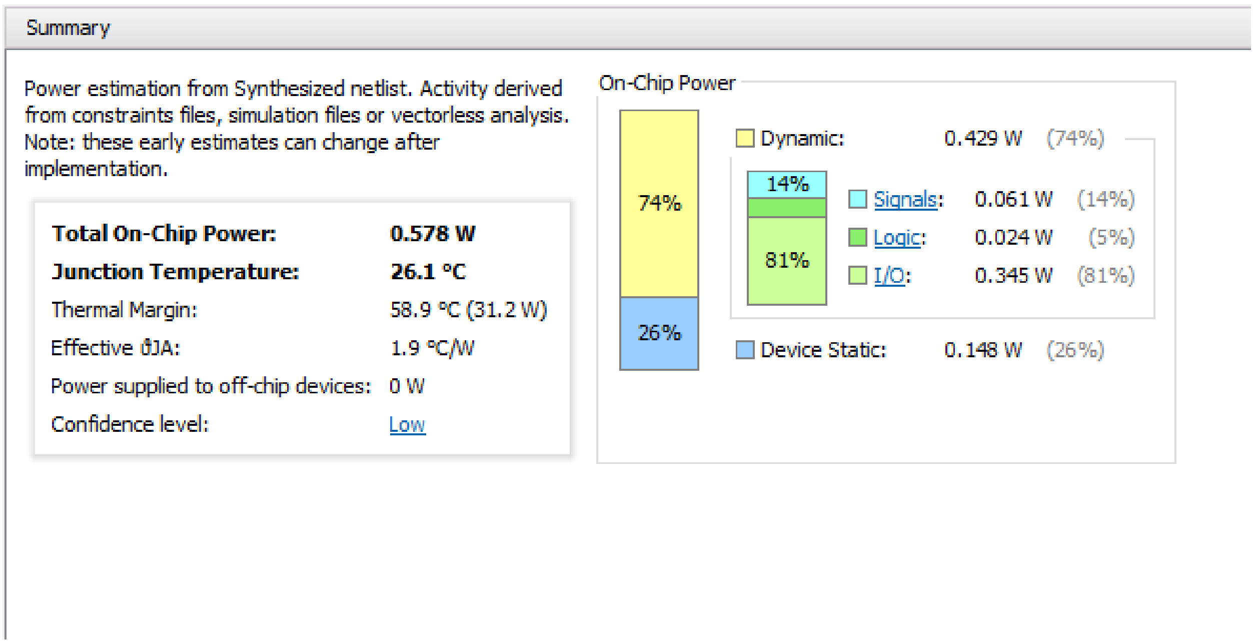


Fig. 5 - Synthesized block diagram

Power utilization of TLC


Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.578 W**
Junction Temperature:	**26.1 °C**
Thermal Margin:	58.9 °C (31.2 W)
Effective ϑJA:	1.9 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

On-Chip Power

Dynamic:	0.429 W	(74%)
Signals:	0.061 W	(14%)
Logic:	0.024 W	(5%)
I/O:	0.345 W	(81%)
Device Static:	0.148 W	(26%)

Fig. 6 - Power utilization table

Memory Utilization of TLC

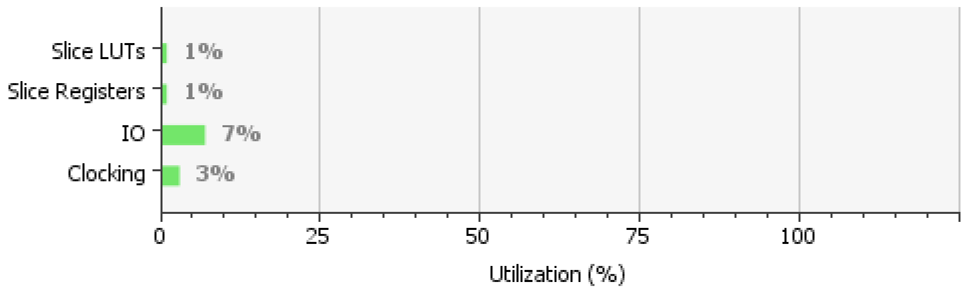


Fig. 7 - Memory utilization

***************End of project***************