INSTITUTE OF TECHNOLOGY NIRMA UNIVERSITY

TRAFFIC LIGHT CONTROLLER

A report of special assignment for

FPGA-SD (2EC202)

Semester IV

Electronics and Communication Engineering

Submitted to:

DR. AKASH MECWAN

Prepared by:

PRAGHYA CHATURVEDI

(22BEC014)

Table of Contents:

ABSTRACT

Traffic Light Controller is implemented using Verilog HDL. The Traffic Light Controller manages traffic flow at intersections, ensuring safe and efficient movement of vehicles. The Verilog code employs finite state machines (FSMs) for sequencing traffic lights, configurable timing parameters, and synchronization mechanisms for smooth transitions. The design is optimized for resource utilization and validated through simulation and FPGA implementation, demonstrating its effectiveness in real-world traffic management.

KEYWORD

Traffic Light Controller, Verilog HDL, Finite State Machines (FSMs), Timing Parameters, Simulation, FPGA Implementation.

STATE OF THE ART TECHNOLOGY AVAILABLE

- **Adaptive Traffic Control Systems:** The present system is connected with real-time information from the sensors, cameras, etc. It dynamically adjust traffic signal timing to match the current situation on roadways.

- **Connected Vehicle Technology:** Vehicles-to-infrastructure (V2I) communication systems that are embedded into traffic lights streamline traffic control and make it more dynamic. Intersection traffic lights, connected with the vehicles, by obtaining data around vehicle speed, location, and vehicle density, can create further coordinated movement of the traffic.

- **Smart Intersection Management:** Advanced intersection management systems leverage technologies like radar, LiDAR, and computer vision to detect vehicles, pedestrians, and cyclists approaching intersections. By accurately identifying different road users, these systems can optimize signal timings, provide priority to road users, and prevent accidents.

LIMITATIONS OR DRAWBACKS of CURRENTLY AVAILABLE TECHNOLOGY

While current traffic light controller (TLC) technologies offer significant advancements, they still face several limitations and drawbacks:

- **Infrastructure Requirements:** Sophisticated TLC systems may require a great deal of network infrastructure- encompassing sensors, cameras, communication networks and backend servers for both data processing and analysis. Establishing and upkeeping such facilities can be a hurdle, particularly when aiming for an instant connection of areas with obsolete or defective infrastructure.

- **Privacy Concerns:** The operator systems that gather data from the vehicles, people and other traffic participants up are pose some questions of privacy. This might also comprise of apprehension related to gathering of storing and hacking into the personal data, while surveillance and monitoring might be the part of concern too.

- **Dependency on Data Accuracy:** The efficiency of adaptive TLC systems relies crucially on the fidelity and credibility of data provision, e.g., traffic sensors, cameras and vehicle to infrastructure interactions. The main problem is the presence of the inaccurate data or the outdated data, which can result in the making of the suboptimal traffic management decisions and possible safety hazards.

- **Cybersecurity Risks:** The TLC network that is linked to the internet can be used as an instrument by cybercriminals to attack the system by way of hacking, data breaches, and interference. Therefore, guarding these systems is the only way to prevent them from IT attacks and continuous cautiousness is required.

PROPOSED SOLUTIONS

Many of the problems can be resolved by using Verilog code of Traffic light controller design, here are some specification mentioned by using it I have designed it.

1) **Parameterize the Verilog code** to allow for flexibility and customization of timing parameters, signal sequences, and synchronization mechanisms.

2) **Define parameters** for timing intervals, transition thresholds, and configurable options to accommodate different traffic scenarios, intersection geometries, and operational requirements.

3) **Simulation and Verification:** Conduct thorough simulation and verification of the Verilog code using industry-standard simulation tools and methodologies. Develop comprehensive test benches to validate the functionality and timing correctness. Perform formal verification and code coverage analysis to ensure the correctness and completeness of the Verilog code.

   Considering all the benefits of Verilog based Traffic Light Controller system we can encounter problems for example in case cybersecurity risk and privacy concern, now there is no chance of that since here we don't relay on data which is sensed by any sensor so no chance of data loss. Also dependency on data and expensive resources are also not require and it can be used in any part of area.

STATE DIAGRAM

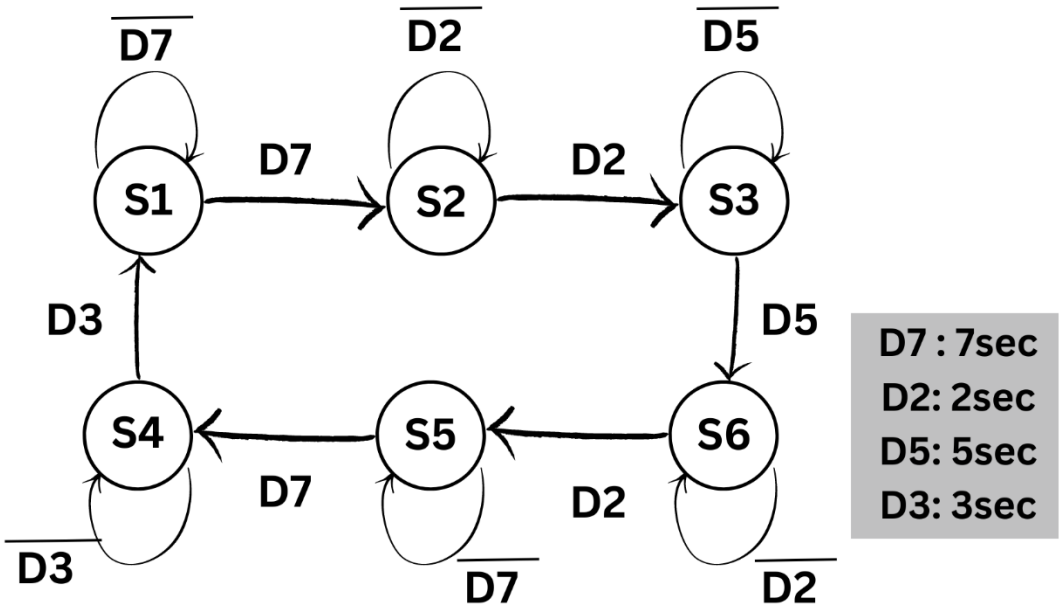

Fig. 1: State Diagram of TLC

STATE TABLE

Present State ABC	Input	Next State	State	mainRoad RYG	mainRoad2 RYG	mainRoadTurn RYG	sideTurn RYG
000	-	001	1	000	000	000	000
001 001	D7’ D7	001 010	0 1	001	001	100	100
010 010	D6’ D6	010 011	0 1	001	010	100	100
011 011	D5’ D5	011 100	0 1	001	100	001	100
100 100	D4’ D4	100 101	0 1	010	100	010	100
101 101	D3’ D3	101 110	0 1	100	100	100	001
110 110	D2’ D2	110 0011	0 1	100	100	100	010
111 111	-	000	0 1	000	000	000	000

Table 1: State Table of TLC

FLOW CHART OF THE CODE

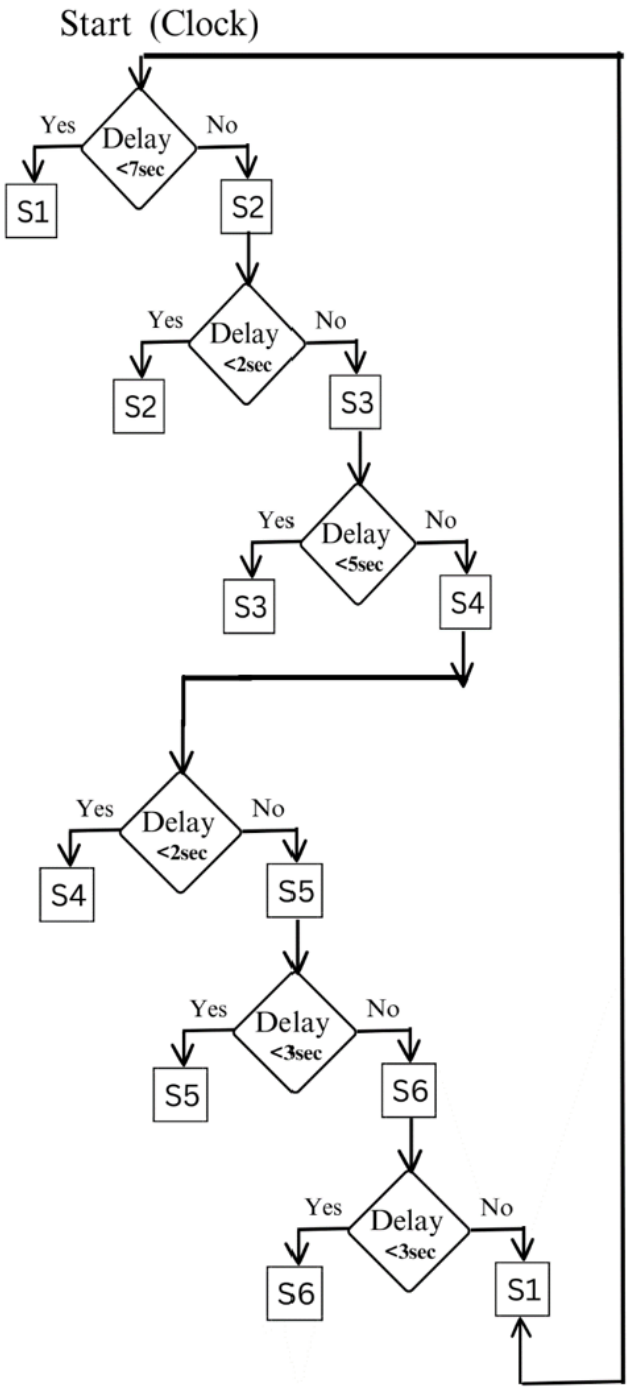

Fig.2: Flowchart of TLC

EXPLANATION OF THE MODEL AND VARIABLES USED

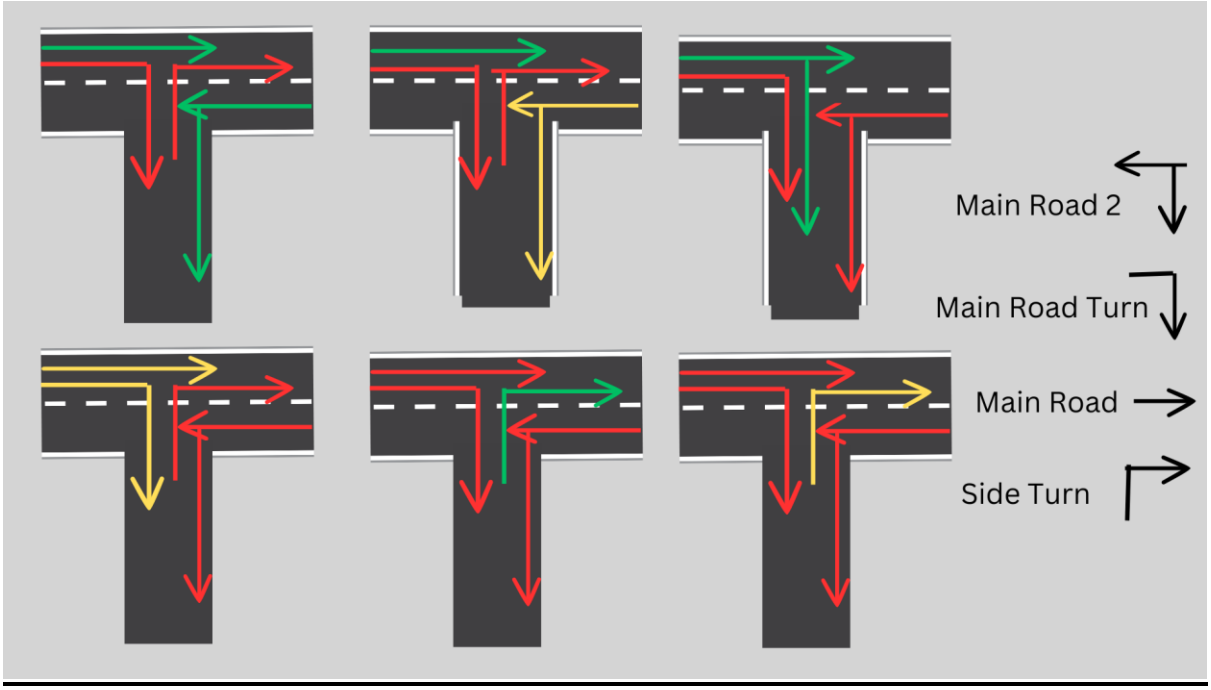

Fig.3: Design

RTL

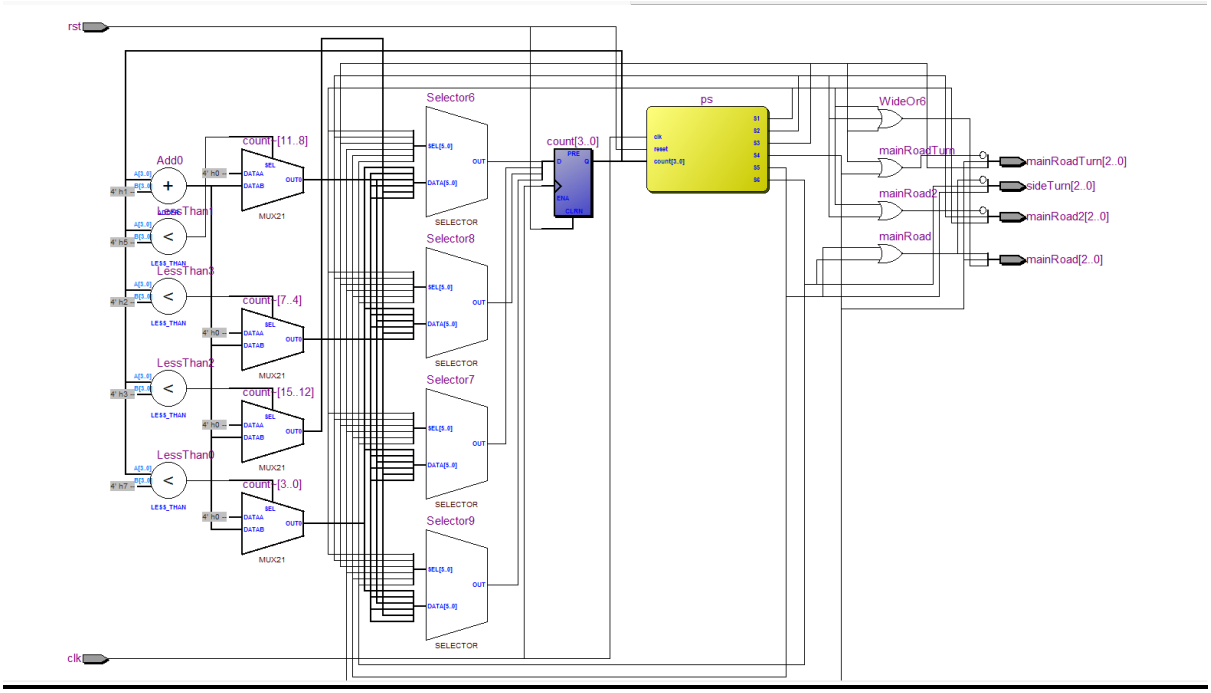

Fig.4: RTL of TLC code

TTL

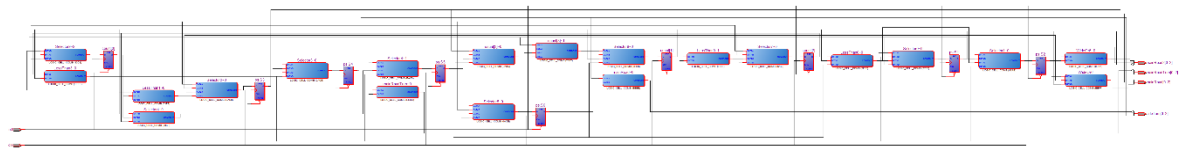

Fig.5: TTL of TLC code

STATE DIAGRAM OF OUTPUT

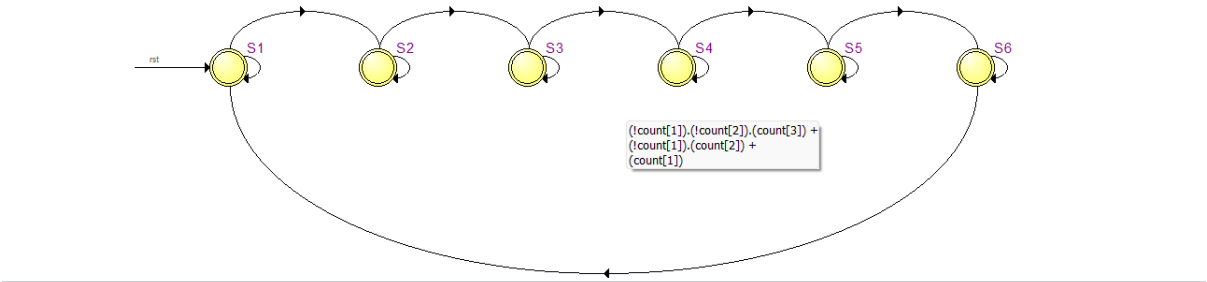

Fig.6: State diagram of RTL

SIMULATION

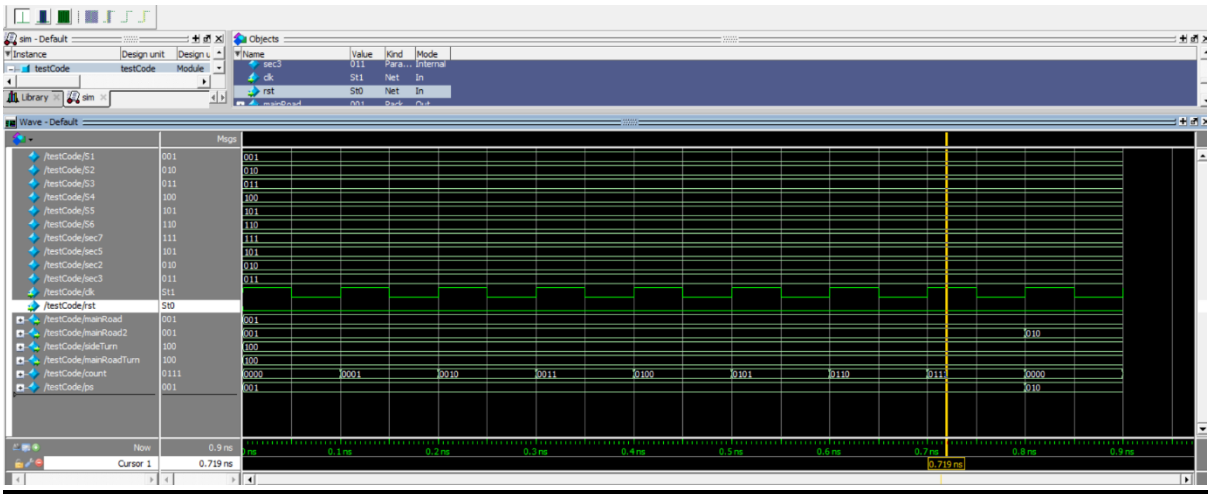

Fig.7: Simulation of code

CONCLUSION

The development of a traffic light controller (TLC) using Verilog code presents an opportunity to address the challenges and limitations of current traffic management systems. By adopting a comprehensive approach encompassing modular design, parameterization, synchronization mechanisms, simulation and verification. Through the design, we can create a flexible and scalable Verilog codebase that facilitates easy integration of new features, customization of timing parameters, and adaptation to diverse traffic scenarios. Using Moore machine, it became easier to implement our design, also it has less power consumption, cost effective, easy maintenance than compare other traffic light controller system.

The fundamental idea of this simple project is to control the traffic. It can be used to avoid the vehicles collisions and traffic. This project is just one-way traffic controller, although it can be further modified as per our need. By this the traffic intensity is also sensed and according to this the time is allotted for traffic to pass. By following the proposed technology, we can elevate the state-of-the-art in traffic light controller design, paving the way for smarter, safer, and more sustainable urban environments.

REFERRENCES

- https://youtu.be/Yt7no6rwCVk?si=FEnBXJUfNTe4pZum
- https://www.studocu.com/in/document/visvesvaraya-technological-university/verilog/traffic-light-controller-using-verilog/31081846

APPENDIX

```
module tfc(clk, rst, mainRoad, mainRoad2, sideTurn, mainRoadTurn);
input clk,rst;            //input
output reg [2:0]mainRoad;  //output
output reg [2:0]mainRoad2;
output reg [2:0]sideTurn;
output reg [2:0]mainRoadTurn;



parameter  S1=3'b001, S2=3'b010, S3 =3'b011, S4=3'b100, S5=3'b101,S6=3'b110; //declaring states
reg [3:0]count=3'b000;
reg[2:0] ps;
parameter  sec7=3'b111,sec5=3'b101,sec2=3'b010,sec3=3'b011;  //Delays



always@(posedge clk or posedge rst) //Assigning the flow of delay
begin
if(rst==1)            //If reset is provide
begin
ps<=S1;
count<=0;
end

else
case(ps)                  //Appling case of each state
S1: if(count<sec7)         //State 1
begin
ps<=S1;
count<=count+1;
```

```
end                    //Go to 2nd State
else
begin
ps<=S2;
count<=0;
end


S2: if(count<sec2)         //2nd State
begin
ps<=S2;
count<=count+1;
end
else                   //Go to 3rd state
begin
ps<=S3;
count<=0;
end

S3: if(count<sec5)       //3rd State
begin
ps<=S3;
count<=count+1;
end
else                 //Go to 4th State
begin
ps<=S4;
count<=0;
end
```

```
S4:if(count<sec2)          //4th State
begin
ps<=S4;
count<=count+1;
end
else                       //Got to 5th state
begin
ps<=S5;
count<=0;
end

S5:if(count<sec3)          //5th state
begin
ps<=S5;
count<=count+1;
end
else                       //Go to 6th State
begin
ps<=S6;
count<=0;
end

S6:if(count<sec2)          //6th State
begin
ps<=S6;
count<=count+1;
end
else                       //Again go to 1st State (loop continues)
begin
ps<=S1;
```

```
count<=0;
end

default: ps<=S1;      //Default case will go to State 1
endcase
end


//Moore machine depends on present state
always@(ps)
begin
case(ps)
//Assigning outputs to every State

S1:                         //State 1
begin
  mainRoad<=3'b001;            //Green color
  mainRoad2<=3'b001;           //Green color
  mainRoadTurn<=3'b100;      //Red color
  sideTurn<=3'b100;           //Red color
end

S2:                         //State 2
begin
  mainRoad<=3'b001;            //Green color
  mainRoad2<=3'b010;           //Yellow color
  mainRoadTurn<=3'b100;       //Red color
  sideTurn<=3'b100;            //Red color
end
```

```
S3:                          //State 3
begin
  mainRoad<=3'b001;                  //Red color
  mainRoad2<=3'b100;                 //Red color
  mainRoadTurn<=3'b001;              //Green color
  sideTurn<=3'b100;                  //Red color
end

S4:                          //State 4
begin
 mainRoad<=3'b010;           //Yellow color
 mainRoad2<=3'b100;          //Red color
 mainRoadTurn<=3'b010;       //Yellow color
 sideTurn<=3'b100;           //Red color
end

S5:                          //State 5
begin
 mainRoad<=3'b100;           //Red color
 mainRoad2<=3'b100;          //Red color
 mainRoadTurn<=3'b100;       //Red color
 sideTurn<=3'b001;           //Green color
end

S6:                          //State 6
begin
 mainRoad<=3'b100;           //Red color
 mainRoad2<=3'b100;          //Red color
 mainRoadTurn<=3'b100;              //Red color
 sideTurn<=3'b010;                  //Yellow color
```

```
end

default:                    //Default case
begin
 mainRoad<=3'b000;          //None of the LED will blink in all these cases
 mainRoad2<=3'b000;
 mainRoadTurn<=3'b000;
 sideTurn<=3'b000;
end
endcase

end
endmodule
```