**Traffic Light Controller Using FPGA**

**A Project Report Submitted**

**In the partial fulfilment the award of degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**ELECTRONICS AND COMMUNICATION ENGINEERING BY**

**CH. ROSHITHA CHARAN SAI - (211801130011)**

**P. GANESH (211801130018)**

**G. CHANDANA (211801130002)**

**K.SAI POORNIMA DEVI (211801130012)**

**CH.PRANEETH (211801131002)**

**Under the esteemed guidance of Mr.K. JOGINAIDU,**

**Assistant professor**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION**

**ENGINEERING**

**CENTURION UNIVERSITY OF TECHNOLOGY &**

**MANAGEMENT**

**Vizianagaram, Andhra Pradesh**

**2022-2023**



**CENTURION UNIVERSITY OF TECHNOLOGY &**

**MANAGEMENT**

**Vizianagaram, Andhra Pradesh**

# BONAFIDE CERTIFICATE

This is to certify that the project work entitled “**Traffic Light Controller Using**

**FPGA**” is a fulfillment of project work done by students, **P. Ganesh (211801130018), CH. Roshitha Charan Sai (211801130011), G. Chandana (211801130002) ,K. Sai Poornima (211801130012) , CH. Praneeth (211801131002)** for the award the degree of BACHELOR OF TECHNOLOGY in ELECTRONICS AND

COMMUNICATION ENGINEERING, during academic year 2023-2024.

**INTERNAL GUIDE HEAD OF THE DEPARTMENT**

**MR. K.JOGINAIDU**, Assistant Professor **MR. K. JOGINAIDU**, Assistant

Deptartment Of ECE Professor Deptartment Of ECE

# ACKNOWLEDGEMENT

I am immensely thankful to Assistant Professor K.Joginaidu, of the Department of Electronics and Communication Engineering at SoET, Vizianagaram Campus. K.Joginaidu sir led me through the complexities of this project effortlessly, displaying unparalleled generosity and guidance.

I wish to express my profound and sincere gratitude to Associate Professor, K. Ramya mam, Department of Electronics and Communication Engineering, SoET, Vizianagaram Campus, who guided me into the intricacies of this project nonchalantly with matchless magnanimity.

I thank Prof. K.J, Head of the Dept. of Department of Computer Science and Engineering, SoET, Vizianagaram Campus for extending their support during Course of this investigation.

I thank Dr. P. A. Sunny Dayal, Dean of SoET, Vizianagaram Campus for their invaluable guidance, insightful feedback, and continuous support throughout the course of this project. Your expertise and mentorship have been invaluable.

I thank Dr. P. Pallavi, Registrar, CUTM, Vizianagaram Campus for their assistance and cooperation in facilitating the necessary resources and administrative support essential for the successful execution of this project.

I thank P. Prasanta Kumar Mohanty, Vice Chancellor, CUTM, Vizianagaram Campus for fostering an environment that encourages academic excellence and innovation. Your vision has been a constant source of inspiration.

I also express my deepest appreciation to my parents for their unconditional love, encouragement, and belief in my abilities. Their unwavering support has been the cornerstone of my achievements.

I am sincerely grateful to each one of you for your contributions, guidance, and unwavering support, without which this project would not have been possible

## DECLARATION

We hereby declare that the project entitled “ Traffic Lignt Controller Using FPGA " submitted for the fulfilment of the award of the degree of B. TECH (ECE) at CENTURION UNIVERSITY OF TECHNOLOGY AND MANAGEMENT (A.P) is our original work and has not been submitted in any part or full for the award of any other degree or diploma at any other university or institute. We certify that all the sources of information and material used in the preparation of this project have been duly acknowledged.

**CH. Roshitha Charan Sai (211801130011),**

**P. Ganesh (211801130018),**

**G. Chandana (211801130002),**

**K. Poornima (211801130012),**

**CH. Praneeth (211801131002)**

**LIST OF ACRONYMS**

**1. ASIC - Application-Specific Integrated Circuit**

**2. RTL - Register Transfer Level**

**3. FPGA - Field-Programmable Gate Array**

**4. GDSII - Graphic Data System II (a file format used in IC design)**

**5. DRC - Design Rule Check**

**6. LVS - Layout versus Schematic**

**7. STA - Static Timing Analysis**

**8. EDA - Electronic Design Automation**

**9. SoC - System on Chip**

**10. IP - Intellectual Property (blocks/modules used in chip design)**

**11. HDL- Hardware descriptive language**

**LIST OF FIGURES page no Fig 1.2: Image of FPGA 1**

**Fig 1.3: FPGA Design flow 2**

**Fig 1.3.1: Design entry 3**

**Figure 1.4 :Synthesis 4**

**Fig 1.4.1:Translate 5**

**Fig 1.4.2: Map 6**

**Fig 1.4.3: Place and Route 7**

**Fig 1.5: Design Flow 8**

**Fig 1.5.3: Timing Stimulation 9**

**Fig 1.6:Static time analysis step-1 11**

**Fig 1.6:Static time analysis step-2 11**

**Fig 2.1:Design flow of Traffic Controller 12**

**Fig 2.2:Traffic controller System 13**

**Fig 2.3: Requirements 15**

**Fig 2.4: Process In Software 16**

**Fig 2.5: Traffic Light Control 17**

**Fig 2.6: FSM Machines 18**

**Fig 2.6.1:Moore &Mealy Machines 18**

**Fig 3.2: Debug Structure 26**

**Fig 3.3:Debug Design Flow 27**

**Fig 3.3:RTL Schematic 27**

**Fig 4.1 RTL Schematic of Traffic Light Controller 29**

**Fig 4.2:RTL Schematic 30**

**Fig 4.3 Clock Source -1 31**

**Fig 4.3.1 : Clock Source-2 31**

**Fig 4.3.1 : Clock Source-2 32**

**Fig 4.4: Wave Forms 33**

**Fig 4.5: Schematic 34**

**Fig 4.6: Equations 34**

**Fig 4.7:Truth Table 35**

**Fig 4.8 :Input and Output Pins 35**

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**CERTIFICATE** i

**DECLARATION ii**

**ACKNOWLEDGEMENT iii**

**LIST OF ACRONYMS iv**

# LIST OF FIGURES

**ABSTRACT**

1. ***CHAPTER – 1 INTRODUCTION 1-11***
   1. “Traffic controller” 1
   2. “Introduction of FPGA” 2
   3. “FPGA Design Flow” 3
      1. “Design Entry” 3
   4. “Synthesis” 4

1.4.1.“translate” 5

1.4.2 “Map” 6

1.4.3.“Place and Route” 7

1.4.4.Device programming 7

* 1. “Design verification” 7
     1. Behavioural Simulation 7
     2. “Functional Simulation” 8
     3. “Timing Simulation” 8
  2. “Static timing Analysis” 9

1. ***CHAPTER – 2 FORMULATION OF PROJECT / IDENTIFICATION OF THE PROBLEM***

***12-18***

* 1. “Design of Traffic Controller” 12
  2. “Explanation of traffic controller System” 13
  3. “Define Requirements” 14
  4. “Choose FPGA board and development environment” 15
  5. “Understand the traffic controller logic” 16
  6. “Develop finite state machine” 17

1. ***CHAPTER 3 PROJECT WORK PART-I***  ***19-28***
   1. “Implementation” 19
   2. “Source code for Testbench” 22
   3. “Document and Optimize” 26
2. ***CHAPTER – 4 PROJECT WORK PART 29-35***

* 1. “Simulation Results” 29
  2. “RTL Schematic” 30
  3. “Technology Schematic” 30
  4. “Wave Forms” 33
  5. “Schematic diagram with inner gates” 34
  6. “Equations” 34
  7. “Truth table” 35
  8. Input and Output Pins 35

1. ***CHAPTER – 5 SUMMARY, CONCLUSIONS & SCOPE FOR FURTHER STUDY*** 
   1. “Summary” 36
   2. “Future Scope” 36
   3. “Conclusion” 37
   4. “References” 37

**FPGA BASED TRAFFIC CONTROLLING SYSTEM**

**ABSTRACT:**

The traffic in road crossings /junctions is controlled by switching ON/OFF

Red, Green & Amber lights in a particular sequence. The Traffic Light Controller is designed to generate a sequence of digital data called switching sequences that can be used to control the traffic lights of a typical four roads junction in a fixed sequence. It is also proposed to implement the day mode and night mode operations. It plays more and more important role in modern management and control of urban traffic to reduce the accident and traffic jam in road. It is a sequential machine to be analyzed and programmed through a multistep process. The device that involves an analysis of existing sequential machines in traffic lights controllers, timing and synchronization and introduction of operation and flashing light synthesis sequence. The methods that are used in this project are design the circuit, write a coding, simulation, synthesis and implement in hardware. In this project, XILINX Software was chosen to design a schematic using schematic edit, writes a coding using Verilog HDL (Hardware Description Language) text editor.

Traffic light controller is a set of rules and instructions that drivers, pilots, train engineer, and ship captains rely on to avoid collisions and other hazards. Traffic control systems include signs, lights and other devices that communicate specific directions, warnings, or requirements. Traffic light controller (TLC) has been implemented using verilog HDL. It has many advantages over other with reference to the speed, number of input/output ports and performance which are all very important in design. This paper concerns with an design implementation of an advanced traffic light controller system that was built as a term project of a VLSI design subject using VHDL. The system has been successfully tested and implemented in hardware using Xilinx Spartan 3 FPGA. The system has many advantages over the other exciting Traffic Light Controller. The VHDL code is being used in order to implement the design and the simulation is being tested using the Isim Simulator.

Keywords: Xilinx, Traffic controlling System, Verilog, VHDL , FPGA, Timing , collisions.

**CHAPTER -1**

### INTRODUCTION

**1.1 TRAFFIC CONTROLLER**

Traffic congestion is a severe problem in many modern cities around the world. Traffic congestion has been causing many critical problems and challenges in the major and most populated cities. To travel to different places within the city is becoming more difficult for the travelers in traffic. Due to these congestion problems, people lose time, miss opportunities, and get frustrated. Traffic congestion directly impacts the companies.

Due to traffic congestions there is a loss in productivity from workers, trade opportunities are lost, delivery gets delayed, and thereby the costs goes on increasing. To solve these congestion problems, we have to build new facilities & infrastructure but at the same time make it smart. The only disadvantage of making new roads on facilities is that it makes the surroundings more congested. So for that reason we need to change the system rather than making new infrastructure twice Therefore many countries are working to manage their existing transportation systems to improve mobility, safety and traffic flows in order to reduce the demand of vehicle use. Therefore, many researches about traffic light system have been done in order to overcome some complicated traffic phenomenon but existent research had been limited about present traffic system in well-travelled traffic scenarios.

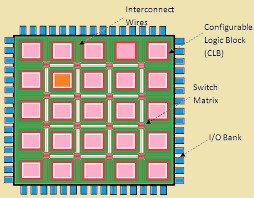
The time of allocation is fixed from east to west or opposite way and from north to south way in crossroads. Field Programmable Gate Arrays (FPGAs) are extensively used in rapid proto typing and verification of a conceptual design and also used in electronic systems when the mask-production of a custom IC becomes prohibitively expensive due to the small quantity. Many system designs that used to be built in custom silicon VLSI are now implemented in Field Programmable Gate Arrays. This is because of the high cost of building a mask production of a custom VLSI especially for small quantity.

FPGA based traffic controller for five intersecting main roads with minimum resource usage. It is interesting to note that the FPGA can have up to 64 logic states (i.e. 64 individual output ports), and his design requires only 25 logic states leaving enough room for other operations.

**1.2 Introduction to FPGA:**

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC).

FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. It is used in broad range of applications.



**Fig 1.2: Image of FPGA**

FPGA is the abbreviation of Field Programmable Gate Array. This denotes an integrated circuit which is programmed in the field, i.e. by the system manufacturer. FPGAs can be characterized by the following items:

* + High production cost

* + Low design density

* + Programmable fabric adds significant overhead

* + No NRE and Re-Spin cost

* + Low development effort

* + Low dead-time

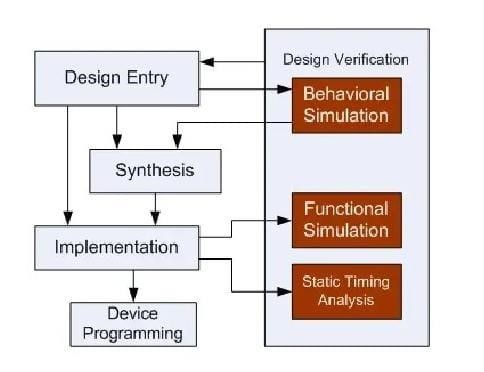
* + simplified timing

* + Relaxed verification

* + Physical design is “hands-off

The circuit description can be done using HDLs, followed by the functional simulation and synthesis. The design flow is followed till the timing simulation and then the generated file is downloaded into the target device (FPGA).

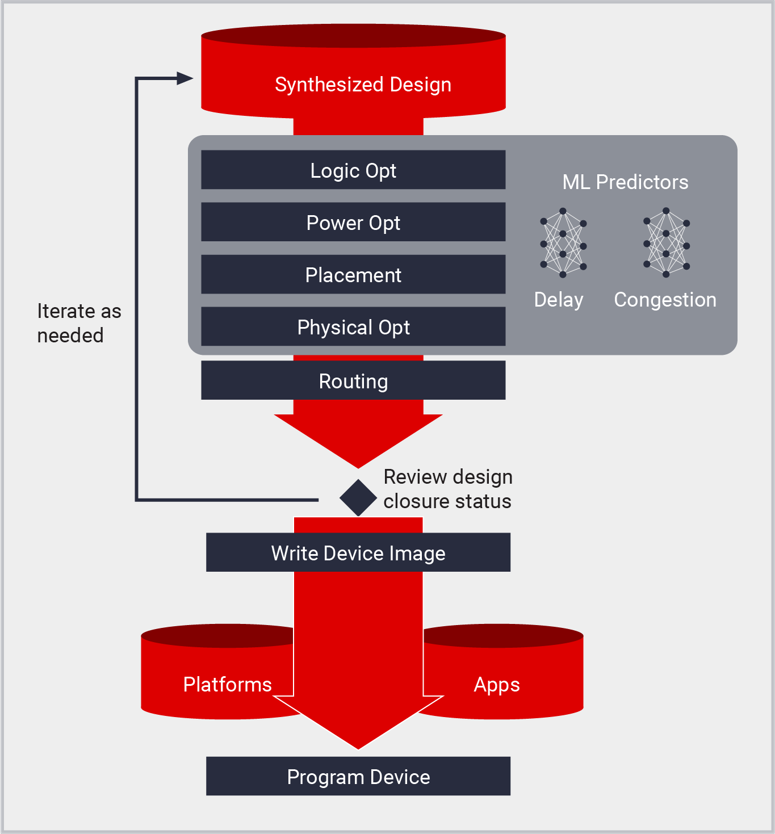
**1.3. FPGA design flow**



**Fig 1.3: FPGA Design flow**

**1.3.1 Design Entry:**

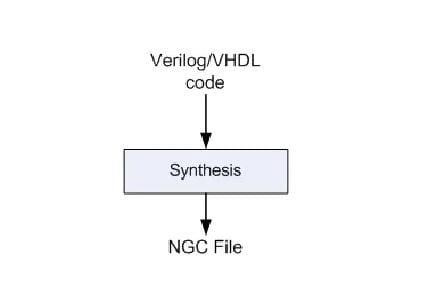
There are different techniques for design entry. Schematic based, Hardware Description Language and combination of both etc. . Selection of a method depends on the design and designer. If the designer wants to deal more with Hardware, then Schematic entry is the better choice. When the design is complex or the designer thinks the design in an algorithmic way then HDL is the better choice. Language based entry is faster but lag in density. HDLs represent a level of abstraction that can isolate the designers from thedetails of the hardware implementation. Schematic based entry gives designers much more visibility into the hardware. It is the better choice forthose who are hardware oriented. Another method but rarely used is state-machines. It is the better choice for the designers who think the design as aseries of states. But the tools for state machine entry are limited. In this documentation we are going to deal with the HDL based design entry.



**Fig 1.3.1: Design entry**

**1.4 Synthesis**

The process which translates VHDL or Verilog code into a device netlist format i.e a complete circuit with logical elements (gates, flip flops, etc…) for the design. If the design contains more than one sub designs, ex. To implement a processor, we need a CPU as one design element and RAM as another and so on, then the synthesis process generates netlist for each design element Synthesis process will check code syntax and analyze the hierarchy of the design which ensures that the design is optimized for the design architecture, the designer has selected. The resulting netlist(s) is saved to an NGC (Native Generic Circuit) file (for Xilinx® Synthesis Technology (XST)).



**Fig 1.4.: Synthesis**

This process consists a sequence of three steps

1. Translate
2. Map
3. Place and Route

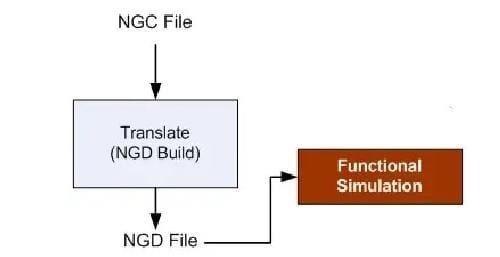
**1.4.1 Translate**

process combines all the input netlists and constraints to a logicdesign file. This information is saved as a NGD (Native Generic Database)file. This can be done using NGD Build program.

Here, defining constraints is nothing but, assigning the ports in the design to the physical elements(ex. pins, switches, buttons etc) of the targeted device and specifying timerequirements of the design.

This information is stored in a file named UCF(User

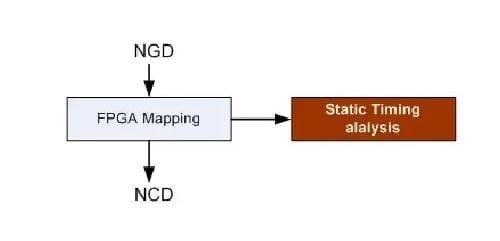
Constraints File). Tools used to create or modify the UCF are PACE, Constraint Editor et



**Fig 1.4 .1:Translate**

* + 1. **MAP**

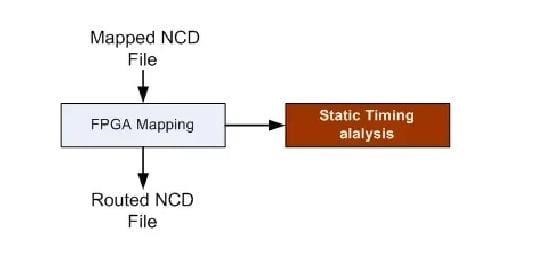
process divides the whole circuit with logical elements into sub blocks such that they can be fit into the FPGA logic blocks. That means map process fits the logic defined by the NGD file into the targeted FPGA elements (Combinational Logic Blocks (CLB), Input Output Blocks (IOB)) and generates an NCD (Native Circuit Description) file which physically represents the design mapped to the components of FPGA. MAP program is used for this purpose**.**



**Fig 1.4.2 : Map**

* + 1. **Place and Route**

PAR program is used for this process. The place and route process places the sub blocks from the map process into logic blocks according to the constraints and connects the logic blocks. Ex. if a sub block is placed in a logic block which is very near to IO pin, then it may save the time but it may affect some other constraint. So trade off between all the constraints is taken account by the place and route process. The PAR tool takes the mapped NCD file as input and produces a completely routed NCD file as output. Output NCD file consists the routing information.



**Fig 1.4.3: Place and route**

**1.4.4 Device Programming:**

Now the design must be loaded on the FPGA. But the design must be converted to a format so that the FPGA can accept it. BITGEN program deals with the conversion. The routed NCD file is then given to the BITGEN program to generate a bit stream (a.BIT file) which can be used to configure the target FPGA device. This can be done using a cable. Selection of cable depends on the design.

* 1. **Design verification:**

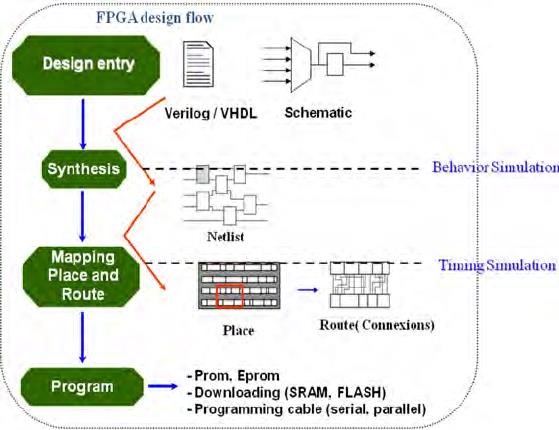
Verification can be done at different stages

**1.5.1 Behavioural Simulation**

(RTL Simulation) This is first of all simulation steps; those are encountered throughout the hierarchy of the design flow. This simulation is performed before synthesis process to verify RTL(behavioural) code and to confirm that the design is functioning as intended.

Behavioural simulation can be performed on either VHDL or Verilog designs.

In this process, signals and variables are observed, procedures and functions are traced and breakpoints are set. This is a very fast simulation and so allows the designer to change the HDL code if the required functionality is not met with in a short time period. Since the design is not yet synthesized.



**Fig 1.5: Design Flow**

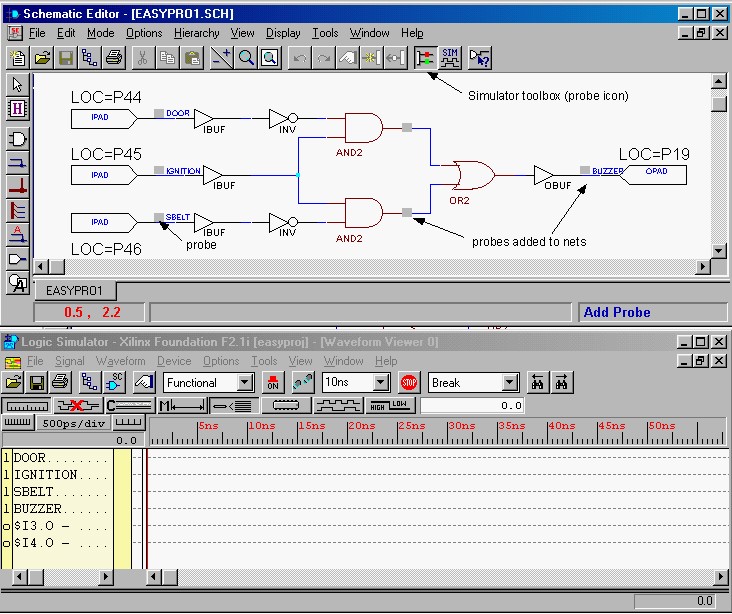
**1.5.2.Functional simulation:**

(Post Translate Simulation) Functional simulation gives information about the logic operation of the circuit. Designer can verify the functionality of the design using this process after the Translate process. If the functionality is not as expected, then the designer has to made changes in the code and again follow the design flow steps.

**1.5.3.Timing Simulation :**

Overview of the Traffic Lights simulation. The Traffic Lights model models the processing for sequencing the Red, Amber and Green lights of a Traffic Lights system. The Traffic Lights system has a Wait button which allows pedestrians to cross the road.

The simulation model is capable of optimizing signal light timing at a single junction as well as an actual road network with multiple junctions. It also provides signal light timing for certain time periods according to traffic demand.



**Fig 1.5.3: Timing Stimulation**

**1.6. Static timing Analysis:**

This can be done after MAP or PAR processes Post MAP timing report lists signal path delays of the design derived from the design logic. Post Place and Route timing report incorporates timing delay information to provide a comprehensive timing summary of the design. Static Timing Analysis (STA) is one of the techniques to verify design in terms of timing.

This kind of analysis doesn’t depend on any data or logic inputs, applied at the input pins. The input to an STA tool is the routed netlist, clock definitions (or clock frequency) and external environment definitions

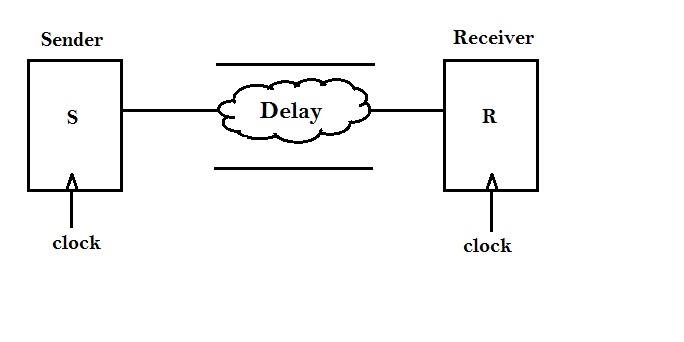
The STA will validate whether the design could operate at the rated clock frequency, without any timing violations. Some of the basic timing violations are **setup violation** and **hold violation.**

Static timing analysis (STA) is a method of validating the timing performance of a design by checking all possible paths for timing violations. STA breaks a design down into timing paths, calculates the signal propagation delay along each path, and checks for violations of timing constraints inside the design and at the input/output interface.

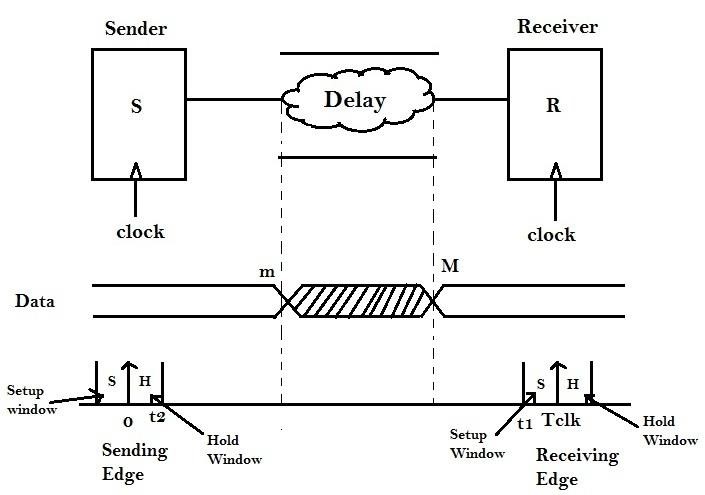
* **Start point.** The start of a timing path where data is launched by a clock edge or where the data must be available at a specific time. Every startpoint must be either an input port or a register clock pin.
* **Combinational logic network.** Elements that have no memory or internal state. Combinational logic can contain AND, OR, XOR, and inverter elements, but cannot contain flip-flops, latches, registers, or RAM.
* **Endpoint.** The end of a timing path where data is captured by a clock edge or where the data must be available at a specific time. Every endpoint must be either a register data input pin or an output port.

STA tool may let you specify the following types of exceptions:

* **False path.** A path that is never sensitized due to the logic configuration, expected data sequence, or operating mode.
* **Multicycle path.** A path designed to take more than one clock cycle from launch to capture.
* **Minimum or maximum delay path.** A path that must meet a delay constraint that you explicitly specify as a time value.



**Fig 1.6:Static time analysis step-1**



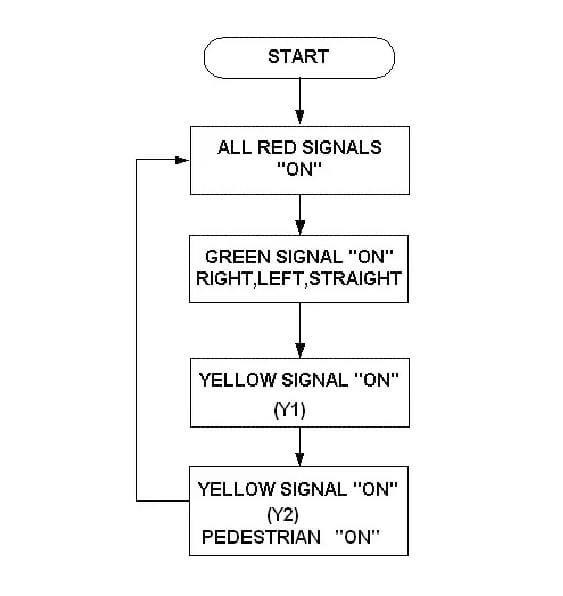
**Fig 1.6:Static time analysis step-2**

### CHAPTER-2

**2.1. DESIGN OF TRAFFIC CONTROLLER**

Traffic Light Controller can be designed by starting with some arbitrary assumptions. At first the North traffic will be allowed to move and then traffic in the East, South and West direction will be allowed to move in sequence. The advantage of writing Traffic Light Controller program is that in a program, modifications as per requirements can be done easily i.e.

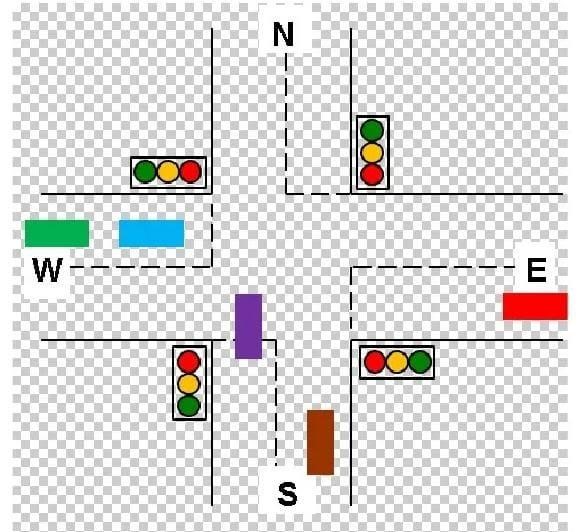
suppose the traffic on main road should be allowed for more time and for side roads the traffic should be allowed for less time; then the clock is divided in such a way that for main road the clock period will be more and for side roads the clock period will be less, this is because the main road traffic is heavy when compared to the side road traffic. In general TLC System will be having three lights (red, green and yellow) in each direction where red light stands for traffic to be stopped, green light stands for traffic to be allowed and yellow light stands for traffic is going to be stopped in few seconds.



**Fig 2.1:Design flow of Traffic Controller**

**2.2. EXPLANATION OF TRAFFIC CONTROLLING SYSTEM**

In this structure, there are four traffic signals, represented by R1, R2, R3and R4 to be controlled. All the four signals have same priority as they all are main roads.



**Fig 2.2:Traffic controller System**

First of all the signal controller is in the reset mode where in the signal of road (R1) is green whereas all the other roads R2, R3 and R4 are red. This state we have assigned as S0. Later the controller sends the control to state S1 where the R1 is yellow whereas all the other signals are still red only. In this state the controller checks whether the sensor at road R2 which is X2 is low or not. If the sensor gives a low signalling that there is no traffic on that road, then that signal on road R2 is skipped transferring control to the state S4 where signal on road R3 is turned whereas rest of the signals are showing red.

On the hand if the traffic is present on the road R2 then the control is sent to state S2 which switches on the signal on road R2 to green and rest of the signals are red only when the control is with state S2 after showing the green signal the signal light changes from green to yellow for signal on the road R2 while all the other signals continue to be in red light mode only which is the operation of state S3. Again

when the controller is in state S3 it checks for the response of sensor X3 on road R3. If the output of sensor is low the control of the system will be transferred to state S6 skipping the working of the signal on roadR3 otherwise the control is given to corresponding next state S4

When in S4 the traffic signal of road R3 turns green on the other hand the signals of roads R1, R2 and R4 remain red itself. The control is then transferred to state S5.

When the control is with state S5 it checks for the output of the sensor X4on the road R4. Depending on the output of X4 the further state change takes place accordingly. If low then the control is transferred to state S0skipping the operation of the signal on road R4 otherwise the control is with the S6. When the controller is in state S5 there is change of signal on roadR3 from green to yellow.

When the control is with state S6 the signal of road R4 turns green where as all the signal turn or remain in red signal only. The control is then shifted to state S0.

In state S7 the signal of road R4 turns from green to yellow. Simultaneously the sensor on the first road R1 which is X1 is checked for its output. If the signal is low then the control is shifted directly to state S2 otherwise the control is shifted to default state S0. These states are not mandatory. The number of states, the order of the lights and the delay can be specified by the user. This is one of the most advantages in this project.

**2.3. Define Requirements:**

Clearly outline the requirements for your traffic light controller. This includes the number of traffic lights, the sequence of light transitions, and any additional features (e.g., pedestrian signals, sensors).

**Process:**

The information of the intersection is collected by the camera, ultrasonic and infrared sensor and transmitted to the FPGA for processing. The data of the camera will be stored in the SDRAM to be called in the first place. After comprehensive processing and judgment of the above road information, the system will be implemented according to the pre-given traffic light scheme.

1. **Camera:**

Always take the current picture of the road and send its data to SDRAM by the way of SPI.

1. **Ultrasonic sensor:**

By sending an ultrasonic signal and receiving a signal from the reflected signal to obtain the presence of an object on the current path. **c) Infrared sensor:**

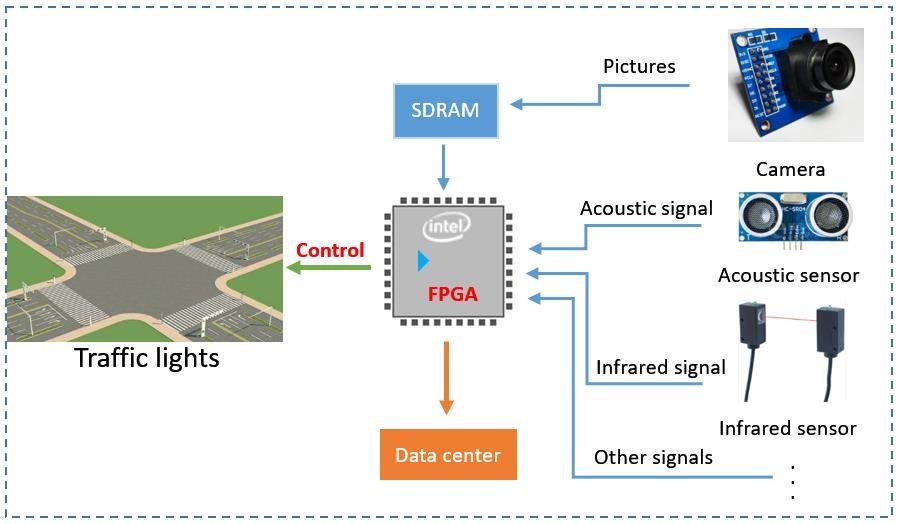
The sending port and the receiving port are separated by the two sides of the road, and the infrared signal is checked to determine whether there is an object in the current road.

1. **SDRAM:**

The image data captured by the temporary camera so that the FPGA can be called at any time during processing.

1. **The data center**

Display current road conditions and traffic light schemes through displays and other devices.

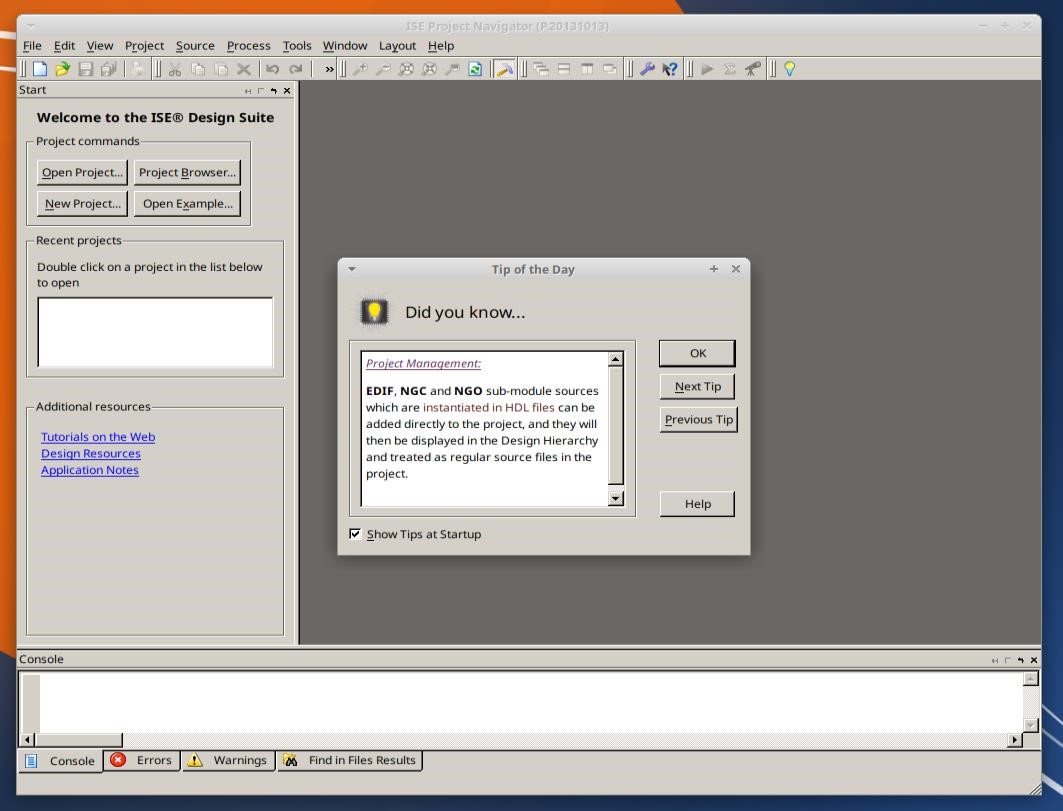


**Fig 2.3: Requirements**

* 1. **Choose FPGA Board and Development Environment:**

Select an FPGA board that suits your project requirements. Popular FPGA vendors include Xilinx and Intel (formerly Altera). Choose a development environment compatible with your FPGA board, such as Xilinx Vivado or Intel Quartus.

The first thing you might like to check is that you can actually launch ISE. Normally you'd need to configure paths and mess around, but xilt can do it for you:

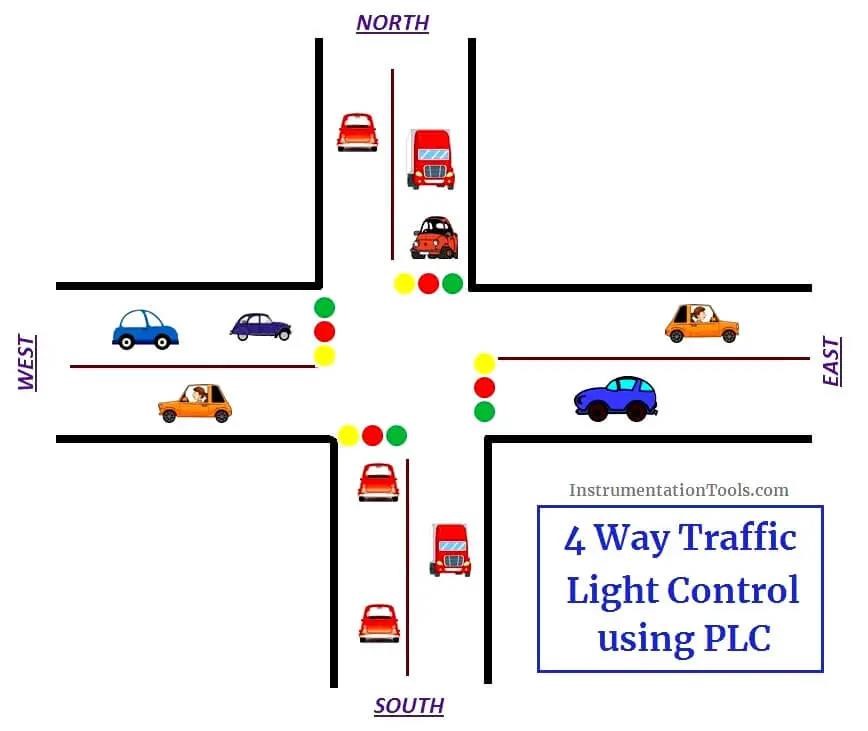


**Fig 2.4: Process In Software**

* 1. **Understand Traffic Light Controller Logic:**

Understand the logic behind a traffic light controller. Typically, a traffic light has three colors: red, yellow, and green. The controller must manage the transition between these colors based on a predefined sequence. we are going to make a “Four Way Traffic Light Circuit”. Several decades back, when people used to travel on foot, there were, of course, no proper roads, and as vehicles got invented, the roads also got constructed.

And, as technology increases, the construction of roads also gets more innovative. Today, there is not just a one or two-way route at one place, but also a 4-way road with 4-way traffic lights.

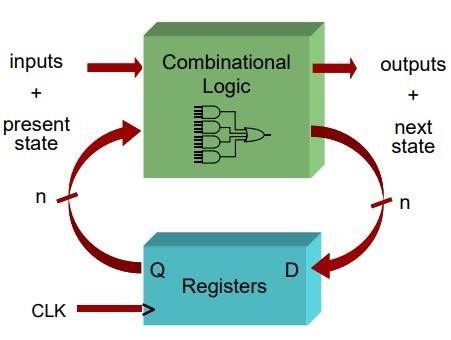


**Fig 2.5: Traffic Light Control**

* 1. **Develop Finite State Machines :**

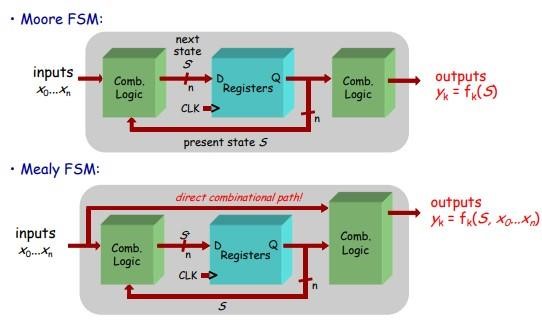
Implement a finite state machine (FSM) to model the behavior of the traffic light controller. Define states for each traffic light color and transitions between them. You can use tools like state diagrams or VHDL/Verilog code to represent the FSM.

* + - * Finite State Machines (FSMs) are a useful abstraction for sequential circuits with centralized “states” of operation
      * At each clock edge, combinational logic computes outputs and next state as a function of inputs and present state



**Fig 2.6: FSM Machines**

❖ They are Two Types of State Machines :



**Fig 2.6.1:Moore &Mealy Machines**

### CHAPTER-3

**3.1. IMPLEMENTATION :**

Write VHDL/Verilog Code:

Write VHDL or Verilog code to describe the logic of your traffic light controller. This code will be synthesized to configure the FPGA. Include modules for input/output handling, clock management, and any additional features like pedestrian signals.

`timescale 1ns / 1ps 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'b010; end default: begin light\_M1<=3'b000; light\_M2<=3'b000; light\_MT<=3'b000; light\_S<=3'b000; end

endcase end endmodule.

**3.2. SOURCE CODE :TEST BENCH**

Program the FPGA:

Program the FPGA with the generated bitstream file using the programming tools provided by the FPGA vendor. This typically involves connecting the FPGA board to your computer and using a programming cable.

module prjctr(state,reset,grn,ylw,rd,an); input[1:0] state; input reset; output reg [3:0]grn,ylw,rd; output reg [3:0]an; always@(reset or state) begin if(reset==0) begin grn=4'b0000; //ylw=4'b0000; ylw=4'b1111; rd=4'b1111; end

else case (state) 2'b00: begin grn=4'b0001; //ylw=4'b0010; ylw=4'b1101; rd=4'b1100; end 2'b01: begin grn=4'b0010; //ylw=4'b0100; ylw=4'b1011; rd=4'b1001; end 2'b10: begin grn=4'b0100; //ylw=4'b1000; ylw=4'b0111; rd=4'b0011; end default:

begin grn=4'b1000; //ylw=4'b0001; ylw=4'b1110; rd=4'b0110; end endcase an=4'b1110; end endmodule Test bench : module tb; // Inputs reg [1:0] state; reg reset; // Outputs wire [4:1] grn; wire [4:1] ylw; wire [4:1] rd;

// Instantiate the Unit Under Test (UUT) prjctr uut (

.state(state),

.reset(reset),

.grn(grn),

.ylw(ylw),

.rd(rd)

);

initial begin // Initialize Inputs state = 2'b00; reset = 1;

// Wait 100 ns for global reset to finish

#100; state = 2'b01; reset = 0;

// Wait 100 ns for global reset to finish

#100; state = 2'b01; reset = 1;

// Wait 100 ns for global reset to finish

#100; state = 2'b10; reset = 1;

// Wait 100 ns for global reset to finish

#100; state = 2'b11; reset = 1;

// Wait 100 ns for global reset to finish #100; state = 2'b00; reset = 1;

// Wait 100 ns for global reset to finish

#100; state = 2'b01; reset = 1;

// Wait 100 ns for global reset to finish

#100; state = 2'b10; reset = 1;

// Wait 100 ns for global reset to finish

#100; state = 2'b11; reset = 1;

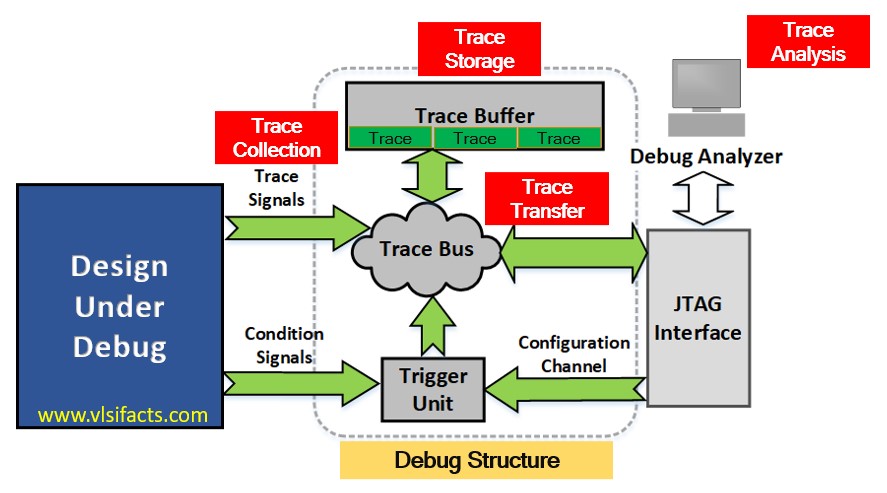
// Wait 100 ns for global reset to finish

#100; state = 2'b00; reset = 1;

// Wait 100 ns for global reset to finish

#100; end endmodule **Debug and Iteration:**

If any issues arise, debug your design by reviewing simulation results and hardware behavior. Make necessary modifications to your code and repeat the synthesis and implementation steps as needed. Debugging is one of the major bottlenecks in the current VLSI design process as design size and complexity increase. Efficient automation of debugging procedures helps to reduce debugging time and to increase diagnosis accuracy. This work proposes an approach for automating the design debugging procedures by integrating SATbased debugging with testbench-based verification. The diagnosis accuracy increases by iterating debugging and counterexample generation, i.e., the total number of fault candidates decreases. The experimental results show that our approach while not requiring a formal specification is as accurate as exact formal debugging in 71% of the experiments.



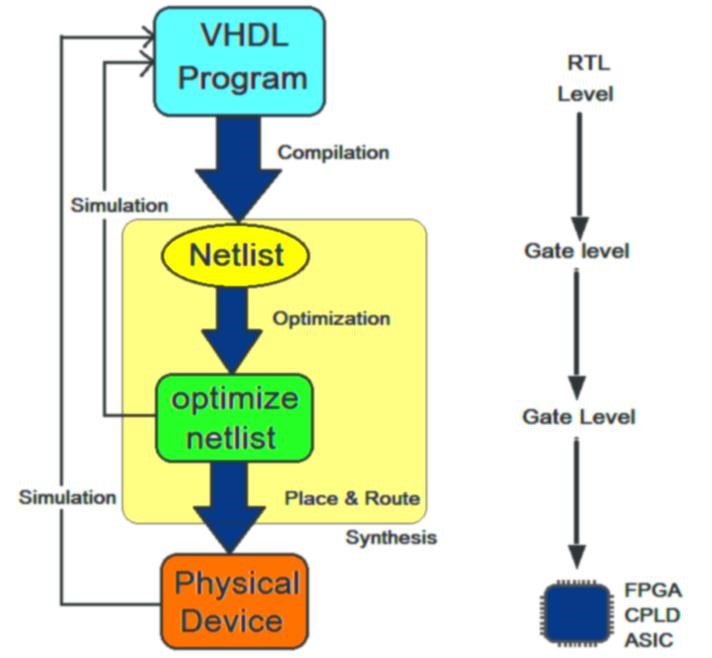
**Fig 3.2: Debug Structure**

Major components of debug structure are as follows:

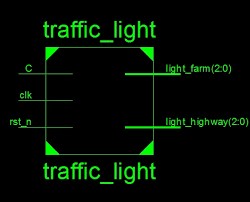
|  |  |
| --- | --- |
| • | Embedded Trace Buffer (ETB) |
| • | Trigger Unit |
| • | Trace Bus |
| • | Trace Port |
| • | Debug Bus |
| • | Debug Port |
| • | Debugger / Debug Analyzer |

**3.3. Document and Optimize:**

Document your design, including the state machine, VHDL/Verilog code, and any relevant details. Optimize your design for resource usage, power consumption, and performance if necessary. The netlist file is a representation of the customized logic used in the Quartus IIsoftware. The file provides the connectivity of architectural elements in the megafunction but may not represent true functionality. This information enables certain third-party synthesis tools to better report timing and resource estimates. In addition, synthesis tools can use the timing information to focus timing-driven optimizations and improve the quality of results.



**Fig 3.3:Debug Design Flow**



**Fig 3.3:RTL Schematic**

**Define I/O Ports:**

Specify input and output ports for the FPGA to interface with external components such as sensors, buttons, and LED displays.

**Simulate Design:**

Use simulation tools provided by your development environment to verify the correctness of your VHDL/Verilog code.

**Implement Design:**

Utilize the implementation tools to map the synthesized design onto the FPGA, taking into account constraints and resource utilization.

**Generate Bitstream:**

Create a bitstream file that represents the configuration of your FPGA with the implemented design.

### PROTOCOL

* The protocol or the design rules we incorporated in designing a traffic light controller are laid down:

* We too have the same three standard signals of a traffic light controller that is RED, GREEN, and YELLOW which carry their usual meanings that of stop go and wait respectively.

* We have two roads the highway road and the side road or country road with the highway road having the higher priority of the two that is it is given more time for motion which implies that the green signal remains for a longer time along the highway side rather than on the country side. We have decided on having a green signal or motion signal on the highway side for a period of 80 seconds and that on the country road of 40 seconds and the yellow signal for a time length of 20 seconds.

* We can have provisions for two exceptions along the roads one along the highway and the other along the country side which interrupt the general cycle.

* Whenever an exception like an emergency vehicle or any such kind of exceptions which have to be addressed quickly. When these interrupts occur the normal sequence is disturbed and the cycle goes into different states depending on its present state and the interrupt occurs.

* We have taken into consideration a two way traffic that is the opposite directions along the highway side will be having the same signals that is the movements along the both direction on a single road will be same at any instant of time. This ensures no jamming of traffic and any accidents at the turnings.

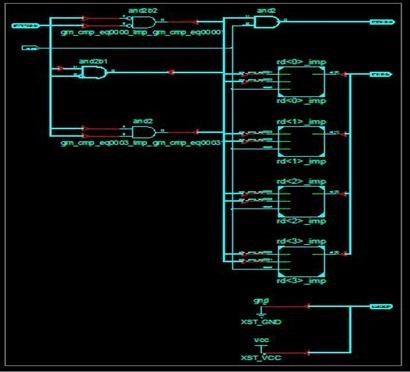
### CHAPTER-4

**SIMULATION RESULTS**

**4.1 The below figure shows the RTL Schematic of the Traffic Light Controller**

**Simulate and Verify:**

Simulate your design using the simulation tools provided by your FPGA development environment. Verify that the traffic light controller behaves as expected in different scenarios.

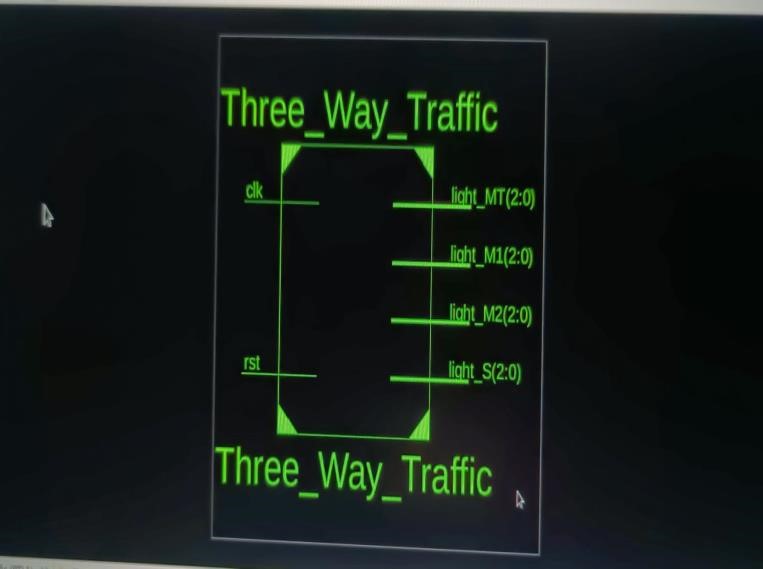


**Fig 4.1 RTL Schematic of Traffic Light Controller**

**4.2. RTL Schematic :**

**Synthesize and Implement:**

Use the synthesis tool in your development environment to generate a configuration file for the FPGA. Implement the design onto the FPGA board. Ensure that your design meets timing constraints and resource utilization.



**Fig 4.2:RTL Schematic**

**4.3. Technology Schematic :**

**The below figure shows the Technology Schematic of the Traffic Light Controller.**

Creating a technology schematic involves using symbols and lines to represent each component and their interconnections.

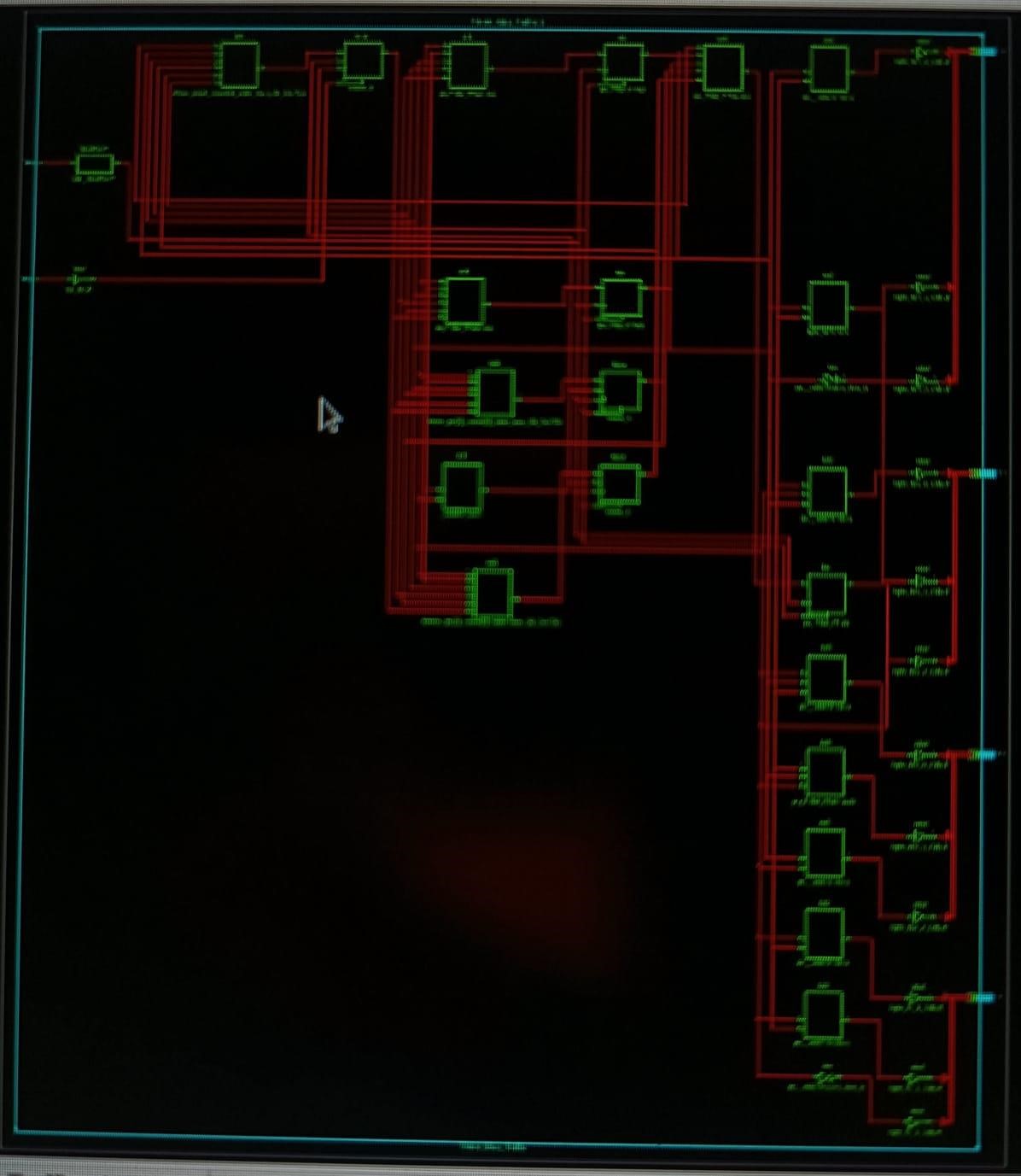
It serves as a valuable tool for engineers, allowing them to understand the overall architecture of the traffic light controller and aiding in troubleshooting, maintenance, and future modifications. The specific details of the schematic may vary based on the complexity of the traffic light control system and the technologies used in its implementation.

**Input Interfaces:**

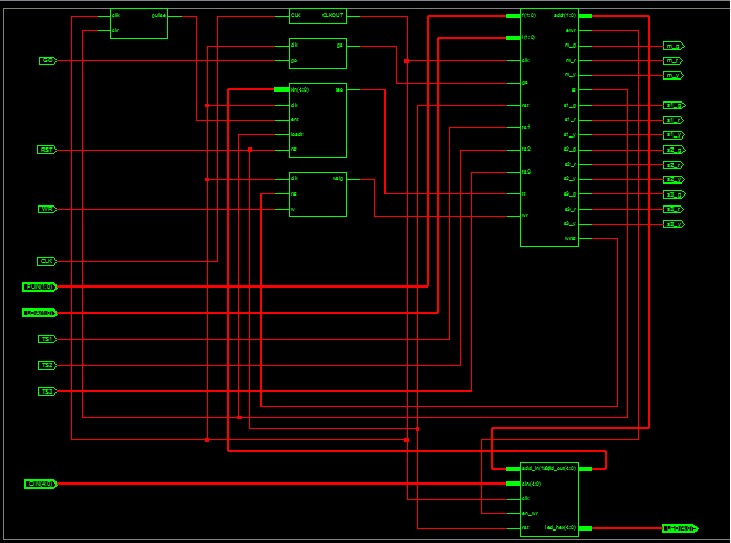
These interfaces connect the controller to external inputs such as sensors, buttons, or cameras. Sensors may include loop detectors embedded in the road to detect the presence of vehicles or pedestrians.

**Clock Source:**

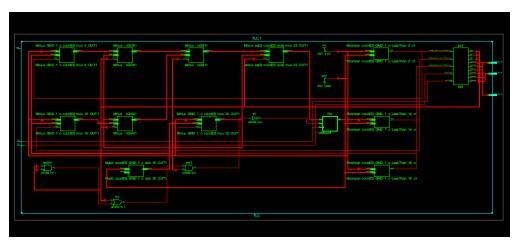
A clock source provides the necessary timing for the controller's operation, ensuring that the state transitions occur at the correct intervals.



**Fig 4.3 Clock Source -1**



**Fig 4.3.1 : Clock Source-2**



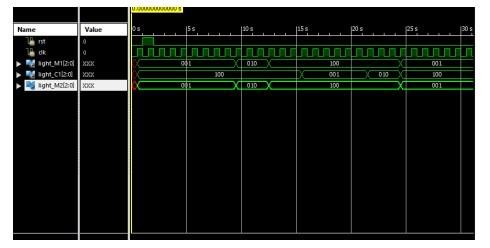
**Fig 4.3.2: Clock Source-3**

**4.4. Wave Forms :**

**The below figure shows the Wave form of the Traffic Light Controller when the test bench is applied to the source code.**

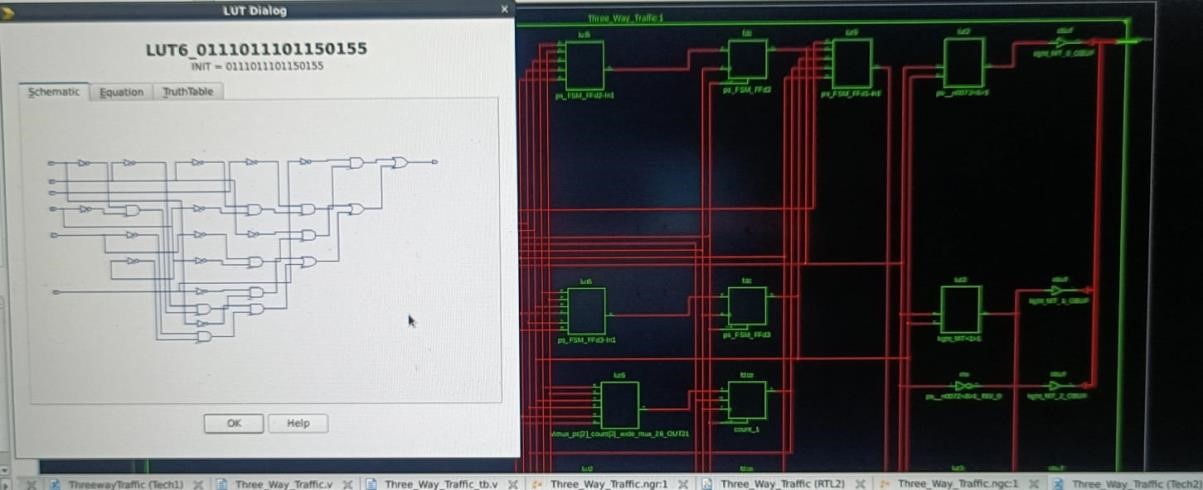


**Fig 4.4: Wave Forms**



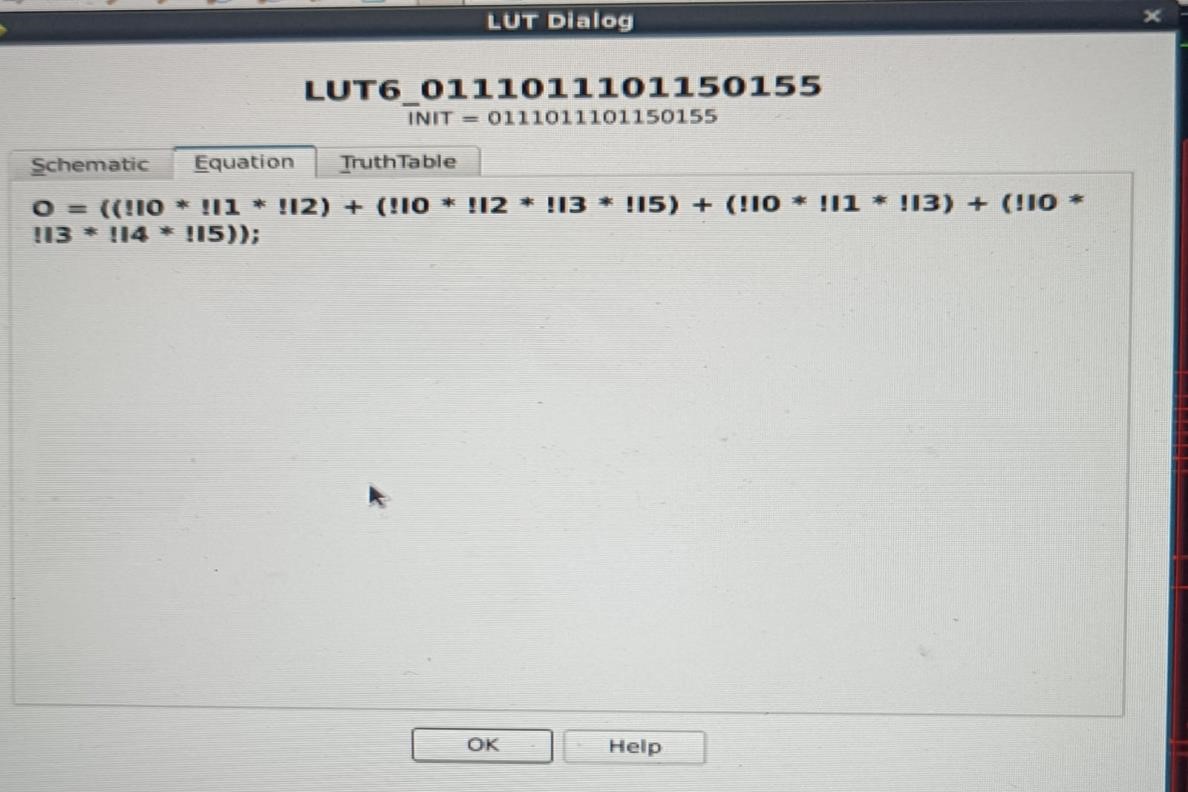
**Fig 4.4: Wave Forms**

**4.5. Schematic Diagram :**



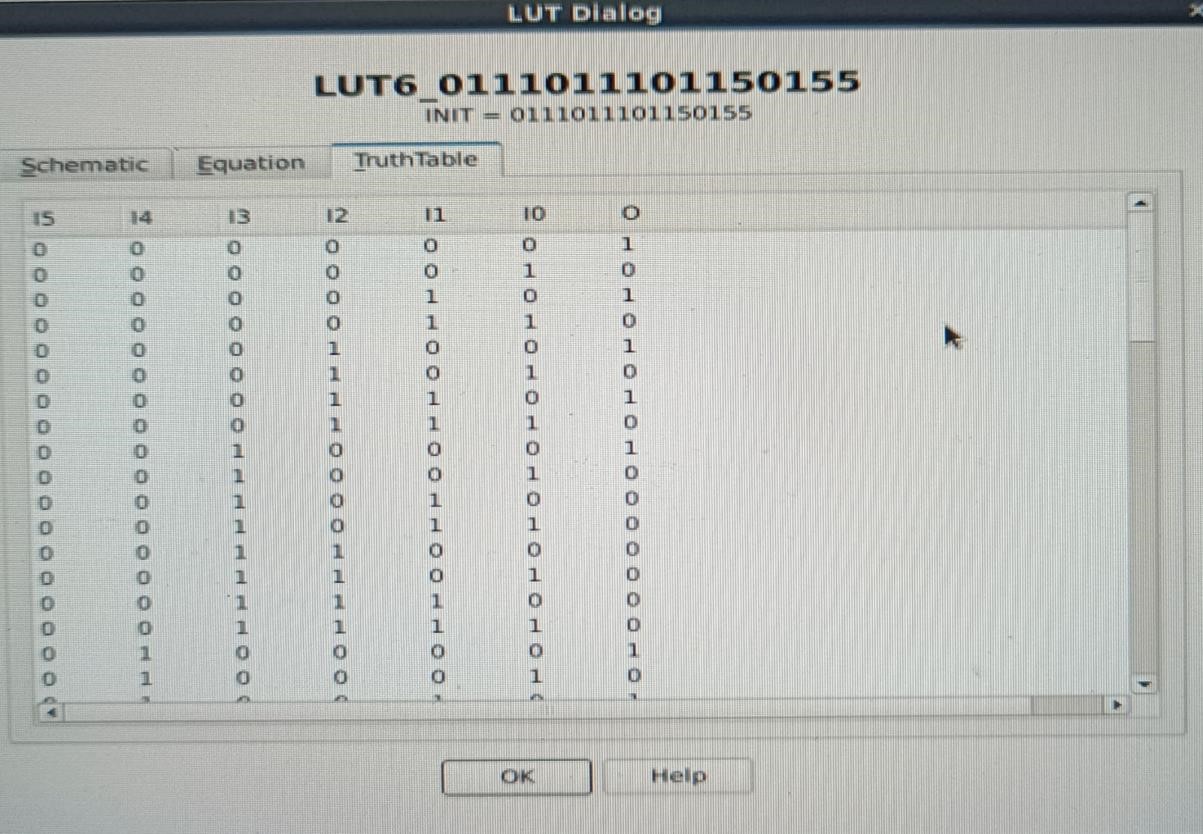
**Fig 4.5: Schematic**

**4.6. Equations :**



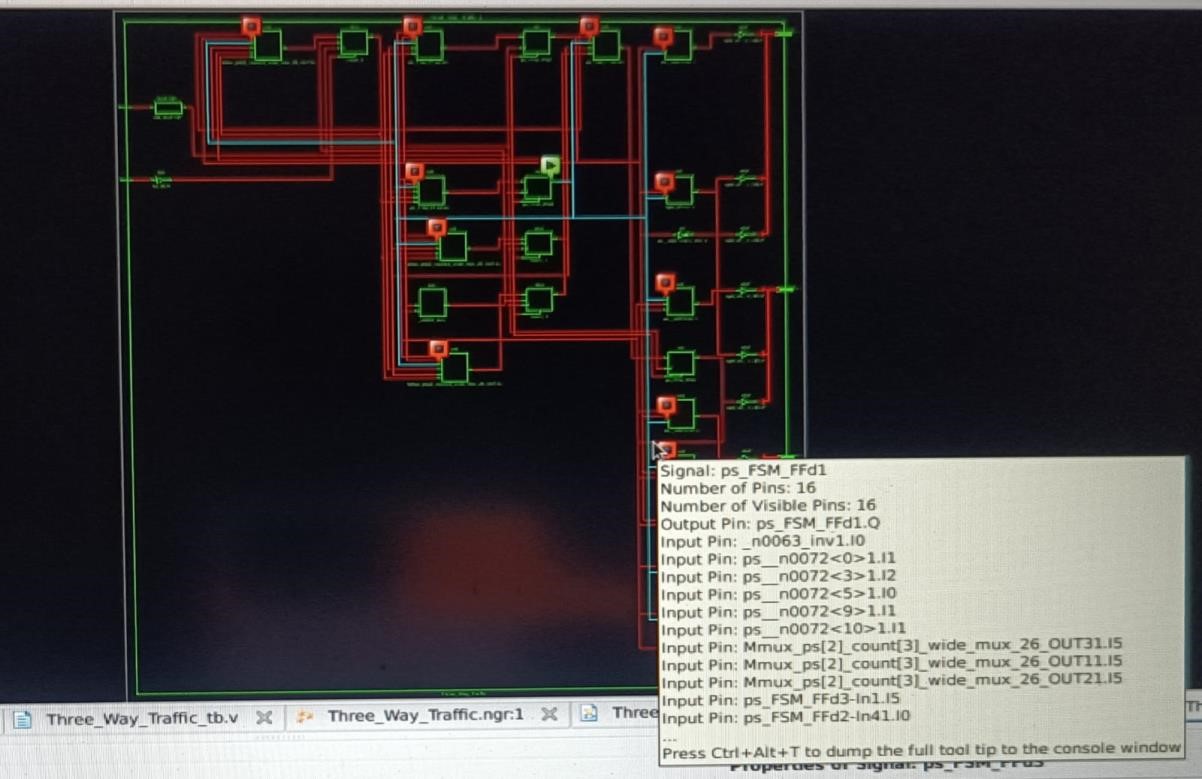
**Fig 4.6: Equations**

**4.7. Truth Table :**



**Fig 4.7:Truth Table**

**4.8. Input & Output pins:**



**Fig 4.8 :Input and Output Pins-**

**CHAPTER-5**

**SUMMARY ,CONCLUSION AND FUTURE SCOPE**

### 5.1 SUMMARY

Field-Programmable Gate Array (FPGA) based traffic controlling system is designed to manage and regulate traffic flow in various settings, utilizing the flexibility and programmability of Field-Programmable Gate Arrays (FPGAs). FPGAs are reconfigurable hardware components that allow for flexible programming of digital circuits. The system integrates various sensors like cameras, radar, and inductive loop detectors to gather real-time traffic data. Using the collected data, the system employs algorithms and AI models to analyse traffic conditions.

In summary, an FPGA-based traffic controlling system is a sophisticated solution leveraging FPGA technology to analyse, manage, and optimize traffic flow by dynamically adjusting signals and patterns based on real-time data, ultimately aiming to improve overall traffic efficiency and safety.

The improvement of town traffic condition is largely dependent on the modern ways of traffic management and control. Advanced traffic signal controllers and control system contribute to the improvement of the urban traffic problem. The intelligent of traffic signal controller that is introduced in this project with powerful functions and hardware interface. Good quality social benefit has been made through the application of the intelligent traffic controller in practice, and the application result shows that the intelligent traffic signal controller will improve

The project holds significant promise for enhancing urban mobility. Its applications include adapting to dynamic traffic scenarios, improving traffic flow, and contributing to the overall safety and efficiency of urban transportation. The project aligns with the evolving needs of urban environments, positioning FPGA-based solutions as valuable contributors to the development of intelligent and responsive traffic control systems.

The FPGA-based traffic controlling system in VLSI project aims to address the challenges of urban traffic by leveraging the versatility and adaptability of Field-Programmable Gate Array (FPGA) technology.

This project is designed to provide real-time traffic optimization, efficient vehicle detection, and data analytics for intelligent transportation systems. With a focus on smart city initiatives and the increasing demand for advanced traffic management solutions,

### 5.2 FUTURE SCOPE

Nowadays there are so many researches that have been done in order to enhance the traffic light function to improve the movement of the vehicle flow on the road especially at the junctions. Based on observation and researches, the condition now is fixed time mode. It means the timer for all of roads are constant timer. It cannot detect a busy and non-busy road.

As the population increases , the number of vehicles also gets increased. To control the huge number of vehicles the intelligent method is to be adopted. In future purpose we can use an image sensor. It does its work by producing the image of the roads and it creates the image by converting the variable attenuation of light into signal that conveys as an image.

Lattice mVision Solutions Stack accelerates low power embedded vision development and includes the modular hardware development boards, design software, embedded vision IP portfolio, and reference designs and demos needed to implement sensor bridging, sensor aggregation, and image processing applications.

The modern ways of four -way junction traffic light controller can improves the traffic condition up to a large extent. Advanced signaling controllers contribute to the improvement in the traffic condition; and also the prevention of the road accidents. This implementation of four junction traffic light controller can challenge any complexity in the traffic. The future scope of this project is it can be directly applied in real time by employing more number of such circuits.

FPGA-based traffic controlling system in VLSI has promising future prospects. It enables real-time reconfigurability, adaptability, and efficient processing for dynamic traffic scenarios. This technology aligns with the growing demand for smart cities and intelligent transportation systems, offering potential applications in traffic optimization, vehicle detection, and data analytics.

As urban areas continue to expand, the need for advanced traffic management solutions makes FPGA-based projects in this domain relevant and impactful.

FPGA-based solutions, positioning such projects at the forefront of addressing the challenges posed by urban mobility in the future.

The project's potential applications include real-time traffic optimization, efficient vehicle detection, and data analytics to enhance overall traffic flow and safety.

**5.3 CONCLUSION**

An smart traffic controller is simulated which works with a normal counter moving green light to adjacent sides whilstothers are red. Priority is given to ambulance, in case detected the particular line is made and the ambulance is allowed to pass. And also if density on one of the sides is more the time of green light is increased to clear the traffic. Further the data from the sensor can be recorded and store in cloud.

This data can be used to analyse the traffic density in the area. The traffic in road crossings /junctions is controlled by switching ON/OFF Red, Green & Amber lights in a particular sequence. The Traffic Light Controller is designed to generate a sequence of digital data called switching sequences that can be used to control the traffic lights of a typical four roads junction in a fixed sequence. It is also proposed to implement the day mode and night mode operations.

The modern ways of four -way junction traffic light controller can improves the traffic condition up to a large extent. Advanced signaling controllers contribute to the improvement in the traffic condition; and also the prevention of the road accidents. This implementation of four junction traffic light controller can challenge any complexity in the traffic. The future scope of this project is it can be directly applied in real time by employing more number of such circuits

The behavior of traffic light controller is analyzed for different situation that can be contributed to the improvement to the urban traffic. Advanced Traffic controllers contribute to the improvement of the urban traffic; which is proportional to the complexity of the controller. The more complex controllers can be well handled using states machines. These methods are used to reduce the states in the state machine which also helps in reducing the required hardware thus leading to low power and area efficient design. The discussion made to implement in Xilinx-Spartan 3E FPGA board, and is functionally verified by using ISim simulator.

### 5.4 REFRENCES

* “FPGA Implementation of an Advanced Traffic Light Controller using Verilog HDL” ➢ CMOS Fabrication Books.
* https://www.ijraset.com/research-paper/advanced-traffic-light-controller-systemusing-fpga-implementation
* Bharathi,Advanced Research in Computer Engineering & Science ➢ Technology; Volume 1, Issue 7, pp: 2278 – 1323,2012.
* Faculty : Mr. K. JOGINAIDU sir
* <https://ieeexplore.ieee.org/document/5952814>
* Shyam Shankaran R,Logesh Rajendran (2021) ‘Real-Time Adaptive Traffic Control System For Smart Cities’, 2021 International Conference on Computer Communication and Informatics (ICCCI) | 978-1-7281-5875-4/21/$31.00 ©2021 IEEEDOI:10.1109/ICCCI50826.2021.9402597
* Amaresh A M, Kavya Shivanand Bhat,Ashwini G,Bhagyashree J , Aishwarya P(2019) ‘Density Based Smart Traffic Control System for Congregating Traffic Information’, Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2019) IEEE Xplore Part Number: CFP19K34-ART; ISBN: 978-1-5386-8113-8.
* Simrit Kaur, Varshitha V, Siddharth J, Tejus V S, Prasanna Kumar (2018) ‘Adaptive Traffic Light Controller using FPGA’, 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT- 2018), 978-1-5386-2440-1/18/$31.00 ©2018 IEEE.
* Nouha Rida, Mohammed Ouadoud, Aberrahim Hasbi, Samira chebli (2018)NAdaptive Traffic Light Control System Using Wireless Sensors Networks’, ICEMI\'09. 9th International Conference on (pp. 1-1040). IEEE 978-1-5386-4385- 3/18/$31.00 ©2018 IEEE.
* Xiao Wen-juan, Liu Jian-feng (2018) ‘Application of Vision Sensing Technology in Urban Intelligent Traffic Control System’, 2018 4th International Conference on Computer and Technology Applications 978-1-5386-6995-2/18/$31.00 ©2018 IEEE..
* T. Bala Obula Reddy, V.Sowmya (2017) ‘An Advanced Traffic Light Controller using Verilog HDL’, International Journal of VLSI System Design and Communication Systems Volume.05, IssueNo.07, July-2017.
* Nath, S., Pal, C., Sau, S., Mukherjee, S., Roy, A., Guchhait, A., Kandar, D. (2012). Design of an FPGA based intelligence trac light controller with VHDL.
* Khalaji, A., Seyedtabaii, S. (2013). Intricacies in digital CMOS implementation of a recongurable fuzzy logic trac light controller. The fifth Conference on Information and Knowledge Technology.
* Traffic light controller Anita Didal, Pragati Sachan, Anchal Katiyar Pallavi Gautam M.Tech Scholar, VLSI, Jayoti Vidyapeeth Womens University Jaipur, Rajasthan, INDIA.