PROJECT REPORT

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

DR. B.R.AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY

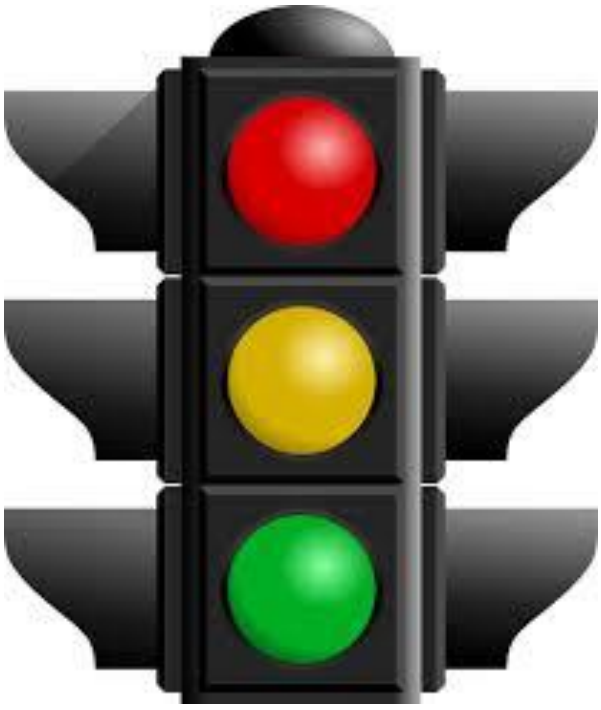
JALANDHAR

SUBMITTED BY:

SOURAV KUMAR

14104033

"DESIGN OF FPGA BASED TRAFFIC LIGHT CONTROLLER SYSTEM INCLUDING CRT"



ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our respected training head at the organisation **CDAC(Centre For Developement of Advance Computing)Mohali**,**Mrs.VEMU SOLOCHANA,** for fostering an excellent academic climate in our institution. I also express my sincere gratitude to our respected Head of the Department **Dr.MAMTA KHOSLA** for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our Project guide at **CDAC Mr.ASHISH RANJAN** for his guidance, encouragement, co-operation and kindness during the entire duration of the course and academics. I am extremely grateful to him for his valuable suggestions and unflinching co-operation throughout project work.

Last but not the least we also thank our friends and family members for helping us in completing the project.

SOURAV KUMAR(14104033)

ABSTRACT

In this project we proposed a design of a modern FPGA-based Traffic Light Control (TLC) System to manage the road traffic. The approach is by controlling the access to areas shared among multiple intersections and allocating effective time between various users, during peak and off-peak hours.

The implementation is based on real location in cities where the existing traffic light controller is a basic fixed-time method. This method is inefficient and almost always leads to traffic congestion during peak hours while drivers are given unnecessary waiting time during off-peak hours. The traffic light controller consists of traffic signals (Red, Yellow/Amber & Green).

Then we have taken the real time waveform as well as the simulated waveform for different frequencies. The proposed design is a more universal and intelligent approach to the situation and has been implemented using FPGA.

Theoretically the waiting time for drivers during off-peak hours has been reduced further, therefore making the system better than the one being used at the moment. Future improvements include addition of other functions to the proposed design to suit various traffic conditions at different locations.

TABLE OF CONTENTS

1. INTRODUCTION TO TRAFFIC LIGHT CONTROLLER SYSTEM 7

1.1 AN OVERVIEW 9

1.2 PROTOCOL

1.3 THE OBJECTIVES 9

2. TECHNOLOGIES USED 10

2.1 FPGA 10

2.1.1 INTRODUCTION TO FPGA'S 10

2.1.2 INTERNAL STRUCTURE 12

2.1.3 APPLICATIONS OF FPGA'S 13

2.2 VLSI 14

2.2.1 INTRODUCTION TO VLSI 14

2.2.2 VLSI AND SYSTEMS 14

2.3 CRT **16**

2.3.1 WORKING OF CRT 16

3. TRAFFIC LIGHT CONTROLLER SYSTEM **17**

3.1 THE STATE MACHINE 18

3.1.1 THE FINITE STATE MACHINE 18

3.1.2 NOTION OF STATES IN SEQUENTIAL 18

3.1.3 WORKING PRINCIPLE OF AN ASM 19

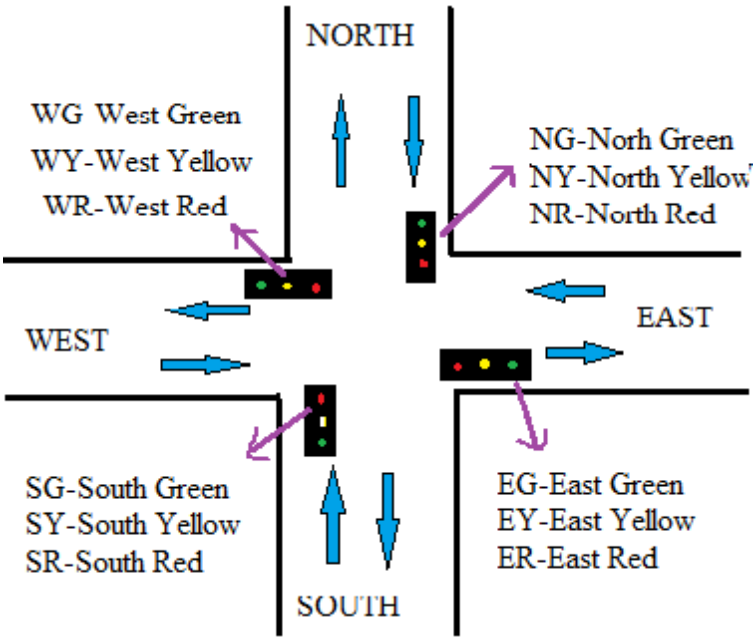
3.1.4 IMPLEMENTATION OF A FSM 20
3.1.5 TYPES OF STATE MACHINES 21
MEALY MACHINE 21
MOORE MACHINE 22
3.2TIMING SETTING 23
3.3SOURCE CODE 24
3.3.1. TLC CODE 24
3.3.2 CRT CODE 28

4 RESULTS 32
4.1 RTL SCHEMATIC 32
4.2 SIMULATION RESULTS 34
5 ADVANTAGES 35
6 CONCLUSION 36
7 REFERENCES 37

1. INTRODUCTION TO TRAFFIC LIGHT CONTROLLER SYSTEM

1.1 AN OVERVIEW

Traffic lights are integral part of modern life. Their proper operation can spell the difference between smooth flowing traffic and four-lane gridlock. Proper operation entails precise timing, cycling through the states correctly, and responding to outside inputs. The traffic light controller is designed to meet a complex specification. That specification documents the requirements that a successful traffic light controller must meet. It consists of an operation specification that describes the different functions the controller must perform, a user interface description specifying what kind of interface the system must present to users, and a detailed protocol for running the traffic lights. Each of these requirements sets imposed new constraints on the design and introduced new problems to solve. The controller to be designed controls the traffic lights of a busy highway (HWY) intersecting a side road (SRD) that has relatively lighter traffic load. Figure 1.1 shows location of the traffic lights. Sensors at the intersection detect the presence of cars on highway and side road.



GENERAL 4-WAY TRAFFIC LANE

The heart of the system is a Finite State Machine (FSM) that directs the unit to light the main and side street lights at appropriate times for the specified time intervals. This unit depends on several inputs which are generated outside the system. In order to safely process these external inputs, we can design an input handler that synchronizes asynchronous inputs to the system clock. The input handler also latches some input signals and guarantees that other input signals will be single pulses, regardless of their duration. This pulsification greatly simplifies the design by ensuring that most external inputs are high for one and only one clock cycle. In addition to the FSM and input handler, the design also includes a slow clock generator. Because the specification requires that timing parameters are specified in seconds, the controller needs to be informed every second that a second of real time has elapsed. The slow clock solves this problem by generating a slow clock pulse that is high for one cycle on the system clock during every second of real time. In addition to generating a once per second pulse, we need to be able to count down from a specified number of seconds. The timer subsystem does that job. When given a particular number of seconds to count down from, it informs the FSM controller after exactly that number of seconds has elapsed. Finally, we have storage and output components. In order to store the users timing parameters, we use a static RAM whose address and control lines are supplied by the FSM. The divider provides a one-second clock that is used by the timer as a count interval. Lastly, the synchronizers ensure that all inputs to the FSM are synchronized to the system clock.

1.2 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 35 seconds and that on the country road of 15 seconds and the yellow signal for a time length of 5 seconds.

- 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.

1.3 THE OBJECTIVES

The following line up as the main objectives of the project.

1. Transform the word description of the protocol into a Finite State Machine transition diagram.
2. Implement a simple finite state machine using VHDL.
3. Simulate the operation of the finite state machine.
4. Implement the design onto a FPGA.

2. TECHNOLOGIES USED

In this project, we are designing an Intelligent Transport System (ITS) application for Traffic Light Controller (TLC) by using Field Programmable Gate Array (FPGA). FPGA have been used for a wide range of applications. After the introduction of the FPGA, the field of programmable logic has expanded exponentially. Due to its ease of design and maintenance, implementation of custom made chips has shifted.

The integration of FPGA and all the small devices will be integrated by using Very Large Scale Integration (VLSI).

The code is written in VHDL design pattern and synthesis is done in XILINX of version 14.6.

Thus, the major technologies used in this project are:

- FPGA
- VLSI
- VHDL
- CRT
- XILINX 14.6 version

2.1. FPGA

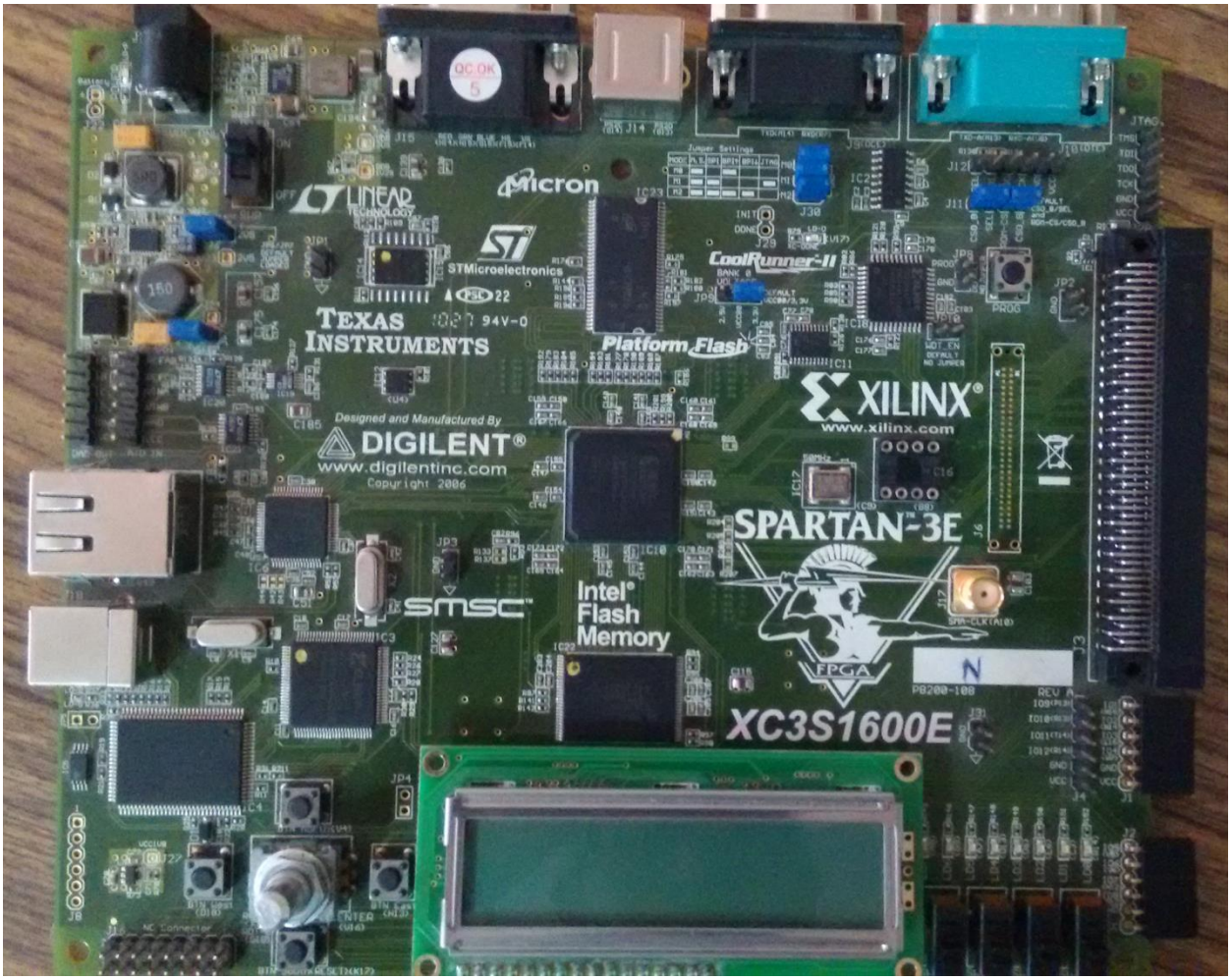
2.1.1 INTRODUCTION TO FPGA'S

A field-programmable gate array (FPGA) is a semiconductor device that can be configured by the customer or designer after manufacturing—hence the name "field-programmable". FPGAs are programmed using a logic circuit diagram or a source code in a hardware description language (HDL) to specify how the chip will work. They can be used to implement any logical function that an application-specific integrated circuit (ASIC) could perform, but the ability to update the functionality after shipping offers advantages for many applications.

Field Programmable Gate Arrays (FPGAs) were first introduced almost two and a half decades ago. Since then they have seen a rapid growth and have become a popular implementation media for digital circuits. The advancement in process technology has greatly enhanced the logic capacity of FPGAs and has in turn made them a viable implementation alternative for larger and complex designs. Further, programmable nature of their logic and routing resources has a dramatic effect on the quality of final device's area, speed, and power consumption.

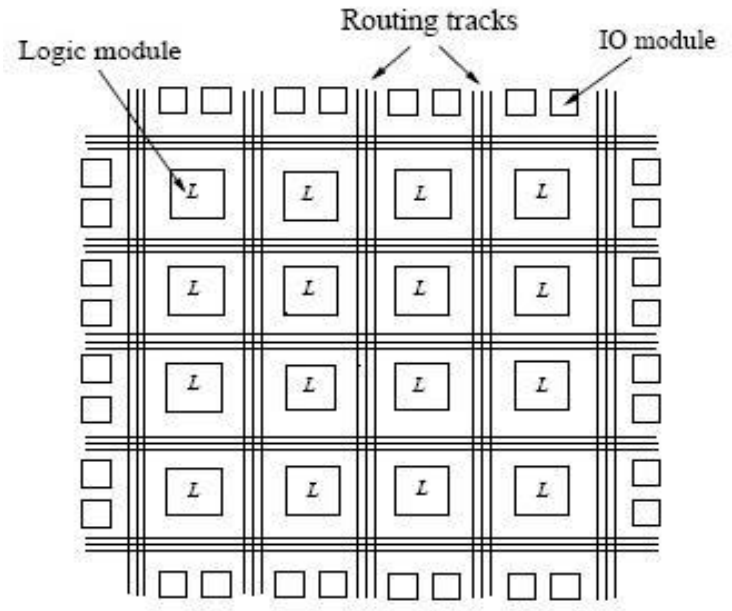
Normally FPGAs comprise of:

- Programmable logic blocks which implement logic functions.
- Programmable routing that connects these logic functions.
- I/O blocks that are connected to logic blocks through routing interconnect and that make off-chip connections.



2.1.2 INTERNAL STRUCTURE OF FPGA

A typical FPGA is composed of three major components: logic modules, routing resources, and input/output (I/O modules) Figure 4.4 depicts the conceptual FPGA model. In an FPGA, an array of logic modules is surrounded or overlapped by general routing resources bounded by I/O modules. The logic modules contain combinational and sequential circuits that implement logic functions. The routing resources comprise pre-fabricated wire segments and programmable switches. The interconnections between the logic modules and the I/O modules are user programmable.



A typical FPGA architecture with three major components: Logic modules, routing resources, and I/O modules.

2.1.3 APPLICATIONS OF FPGA'S

FPGA's have gained rapid acceptance and growth over the past decade because they can be applied to a very wide range of applications. A list of typical applications includes: random logic, integrating multiple SPLDs, device controllers, communication encoding and filtering, small to medium sized systems with SRAM blocks, and many more. Other interesting applications of FPGAs are prototyping of designs later to be implemented in gate arrays, and also emulation of entire large hardware systems. The former of these applications might be possible using only a single large FPGA (which corresponds to a small Gate Array in terms of capacity), and the latter would entail many FPGAs connected by some sort of interconnect; for emulation of hardware, QuickTurn [Wolff90] (and others) has developed products that comprise many FPGAs and the necessary software to partition and map circuits. Another promising area for FPGA application, which is only beginning to be developed, is the usage of FPGAs as custom computing machines. This involves using the programmable parts to "execute" software, rather than compiling the software for execution on a regular CPU. The reader is referred to the FPGA-Based Custom Computing Workshop (FCCM) held for the last four years and published by the IEEE. When designs are mapped into CPLDs, pieces of the design often map naturally to the SPLD-like blocks. However, designs mapped into an FPGA are broken up into logic block-sized pieces and distributed through an area of the FPGA. Depending on the FPGA's interconnect structure, there may be various delays associated with the interconnections between these logic blocks. Thus, FPGA performance often depends more upon how CAD tools map circuits into the chip than is the case for CPLDs. We believe that over time programmable logic will become the dominant form of digital logic design and implementation. Their ease of access, principally through the low cost of the devices, makes them attractive to small firms and small parts of large companies. The fast manufacturing turn-around they provide is an essential element of success in the market. As architecture and CAD tools improve, the disadvantages of FPDs compared to Mask-Programmed Gate Arrays will lessen, and programmable devices will dominate.

2. 2.VLSI

Very-Large-Scale Integration (VLSI) is the process of creating integrated circuits by combining thousands of transistor-based circuits into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. The term is no longer as common as it once was, as chips have increased in complexity into the hundreds of millions of transistors.

2.2.1 INTRODUCTION TO VLSI

VLSI stands for "Very Large Scale Integration". This is the field which involves packing more and more logic devices into smaller and smaller areas.

VLSI

- Simply we say Integrated circuit is many transistors on one chip.
- Design/manufacturing of extremely small, complex circuitry using modified semiconductor material.
- Integrated circuit (IC) may contain millions of transistors, each a few mm in size.
- Wide ranging applications.
- Most electronic logic devices.

2.2.2 VLSI AND SYSTEMS

These advantages of integrated circuits translate into advantages at the system level:

✓ **Smaller physical size:**

Smallness is often an advantage in itself-consider portable televisions or handheld cellular telephones.

✓ **Lower power consumption:**

Replacing a handful of standard parts with a single chip reduces total power consumption. Reducing power consumption has a ripple effect on the rest of the system. A smaller, cheaper power supply can be used, since less power consumption means less heat, a fan may no longer be necessary, a simpler cabinet with less shielding for electromagnetic shielding may be feasible, too.

- ✓ **Reduced cost:**

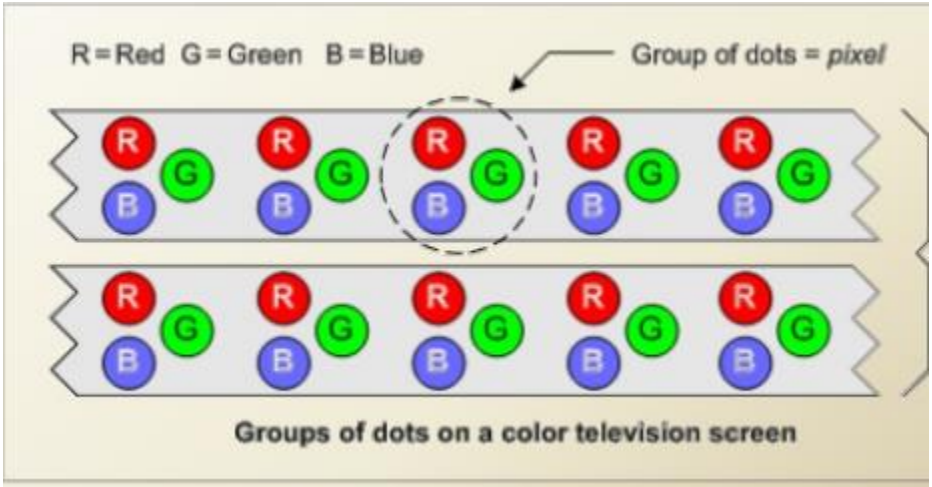
Reducing the number of components, the power supply requirements, cabinet costs, and so on, will inevitably reduce system cost. The ripple effect of integration is such that the cost of a system built from custom ICs can be less, even though the individual ICs cost more than the standard parts they replace.

Understanding why integrated circuit technology has such profound influence on the design of digital systems requires understanding both the technology of IC manufacturing and the economics of ICs and digital systems.

2.3.CRT

The monitors can display different colours by combining various intensities of three beams.

Colours crts are much more complicated. it requires manufacturing very precise geometry.



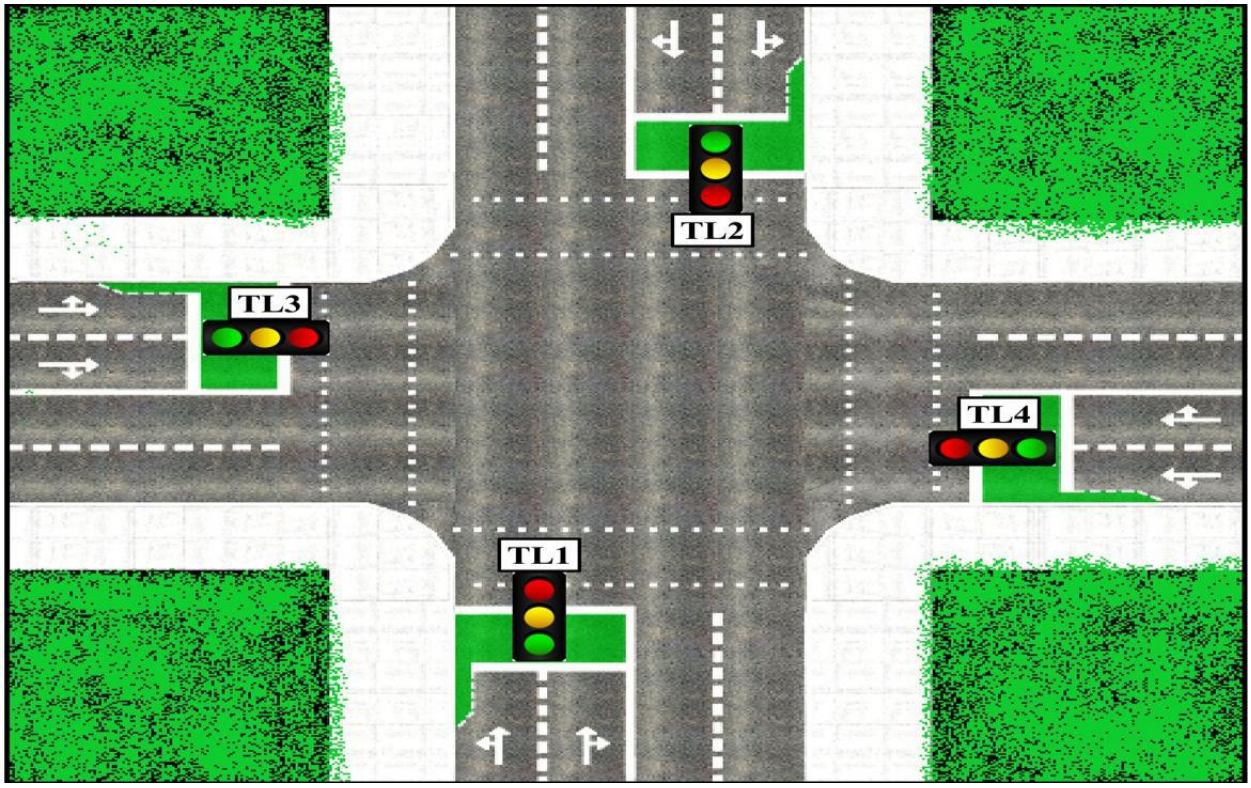
2.3.1.Working of CRT

- Cathode rays are emitted by elecron gun .
- Accelerated by high voltage near the face of tube.
- Forced onto a narrow stream by a focussing system .
- Directed towards a point on the screen
- Hit on to the phosphorus coated screen
- Phosphorous emits visible light whose intensity depends on the number of electrons striking on the screen
- Electron travel through a vaccum scaled container from the cathode to the anode.
- Because the electrons are negtively charged,they are repelled away from the cathode,and Move across the tube to the anode
- The ray can be affected by magnet because of its relation to positive and negative charges
- A crt monitor contains millions of tiny red,green and blue phosphorous dots that glow when struck by an electron beam.Electron beam travels across the screen to create a visible image.

3.TRAFFIC LIGHT CONTROLLER SYSTEM

This method is inefficient and almost always leads to traffic congestion during peak hours while drivers are given unnecessary waiting time during off-peak hours. The traffic light controller consists of traffic signals (Red, Yellow/Amber & Green).

Then we have taken the real time waveform as well as the simulated waveform for different frequencies. The proposed design is a more universal and intelligent approach to the situation and has been implemented using FPGA.



GENERAL 4-WAY TRAFFIC LANE

3.1. THE STATE MACHINE

3.1 .1 THE FINITE STATE MACHINE

The FSM-FINITE STATE MACHINE is the heart of traffic light controller. It responds to the input signals processed by the input handling module and provides the output and control signals needed to make the system function. This design uses a standard two process finite state machine where one process is used to change states on every clock cycle while the other process is used to combinatorically calculate what the next state should be based on the current inputs and the current state. That combinatorial process also sets the outputs for the next clock cycle.

Finite State Machine (FSM) or Finite State Automation, or simply a state machine, is a mathematical abstraction sometimes used to design digital logic or computer programs. It is a behavior model composed of a finite number of states, transitions between those states, and actions, similar to a flow graph in which one can inspect the way logic runs when certain conditions are met. It has finite internal memory, an input feature that reads symbols in a sequence, one at a time without going backward; and an output feature, which may be in the form of a user interface, once the model is implemented. The operation of an FSM begins from one of the states (called a start state), goes through transitions depending on input to different states and can end in any of those available, however only a certain set of states mark a successful flow of operation (called accept states).

3.1.2 NOTION OF STATES IN SEQUENTIAL MACHINES:

A state machine is a type of sequential circuit structure which can allow you to create more elaborate types of digital systems. A state machine consists of:

1. Memory elements (e.g. D flip-flops) to store the current state.
2. Combinational logic to compute the next state.
3. Combinational logic to compute the output.

3.1.3 WORKING PRINCIPLE OF AN FSM

Finite state machines consist of 4 main elements:

- States which define behavior and may produce actions.
- State transitions which are movement from one state to another.
- Rules or conditions which must be met to allow a state transition.
- Input events which are either externally or internally generated, which may possibly trigger rules and lead to state transitions.

A current state is determined by past states of the system. As such, it can be said to record information about the past, i.e., it reflects the input changes from the system start to the present moment. The number and names of the states typically depend on the different possible states of the memory, e.g. if the memory is three bits long, there are 8 possible states. A transition indicates a state change and is described by a condition that would need to be fulfilled to enable the transition. An action is a description of an activity that is to be performed at a given moment. There are several action types:

Entry action

This is performed when entering the state.

Exit action

This is performed when exiting the state.

Input action

This is performed depending on present state and input conditions.

Transition action

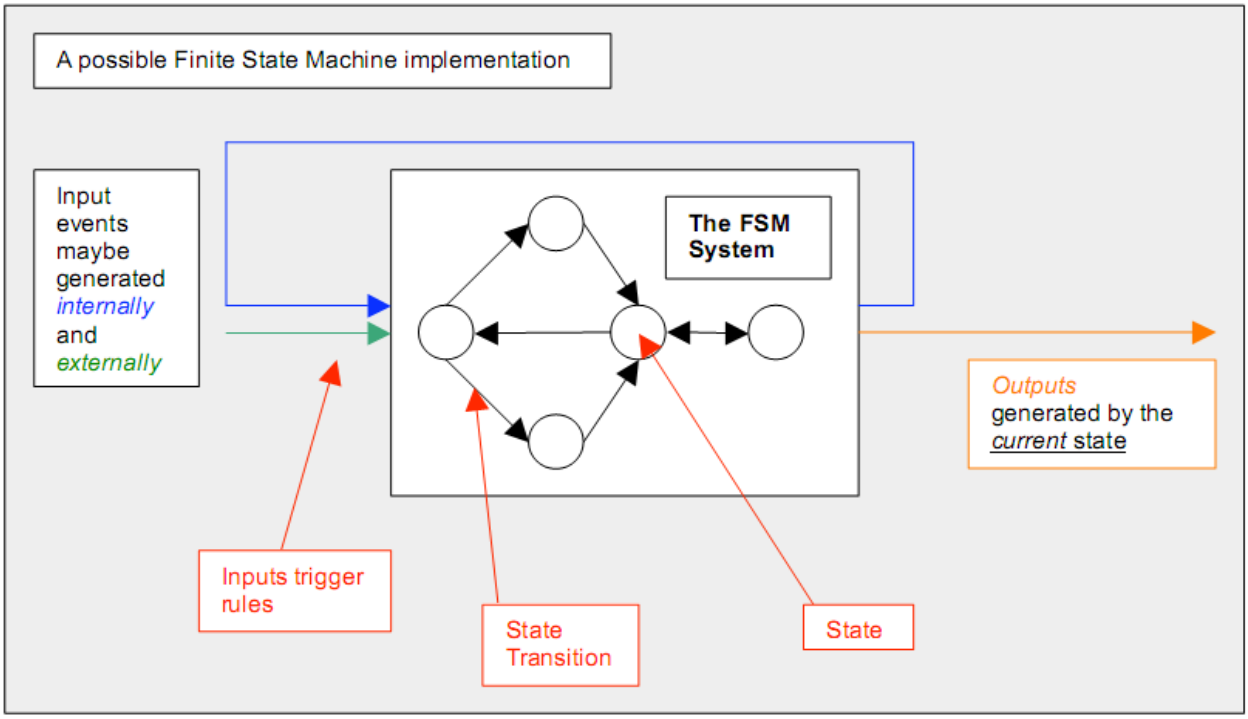
This is performed when performing a certain transition.

3.1.4 IMPLEMENTATION OF A FSM

State Variable: The variable held in the SM (FF) that determines its present state.

A basic FSM has a memory section that holds the present state of the machine (stored in FF) and a control section that controls the next state of the machine (by clocks, inputs, and present state).The outputs and internal flip flops (FF) progress through a predictable sequence of states in response to a clock and other control inputs.

A finite state machine must have an initial state which provides a starting point, and a current state which remembers the product of the last state transition. Received input events act as triggers, which cause an evaluation of some kind of the rules that govern the transitions from the current state to other states. The best way to visualize a FSM is to think of it as a flow chart or a directed graph of states, though as will be shown; there are more accurate abstract modeling techniques that can be used.



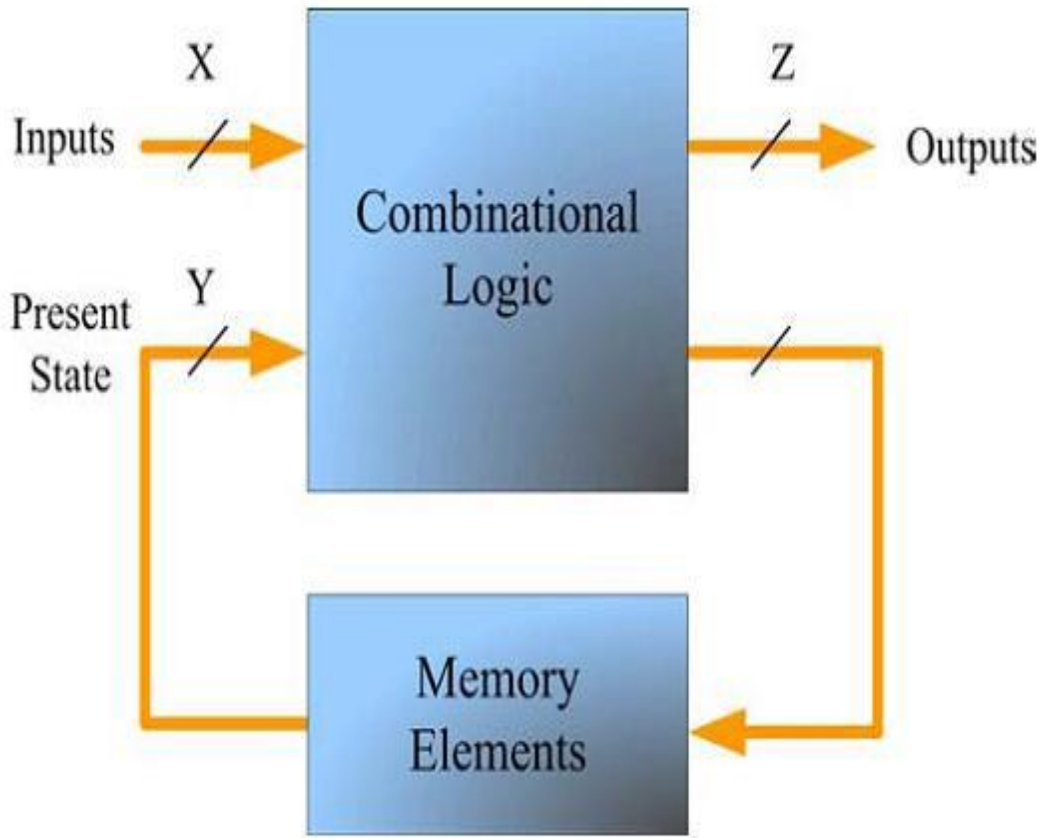
A Possible Finite State Machine

3.1.5 TYPES OF STATE MACHINES

There are two basic ways to design clocked sequential circuits, i.e., the state machines.

MEALY MACHINE

In the theory of computation, a Mealy machine is a finite-state machine whose output values are determined by both its current state and by the values of its inputs. The state diagram for a Mealy machine associates an output value with each transition edge (in contrast to the state diagram for a Moore machine, which associates an output value with each state). In a Mealy machine, the outputs are a function of the present state and the value of the inputs. Accordingly, the outputs may change asynchronously in response to any change in the inputs. A combinational logic block maps the inputs and the current state into the necessary flip-flop inputs to store the appropriate next state just like Mealy machine. However, the outputs are computed by a combinational logic block whose inputs are only the flip-flops state outputs.

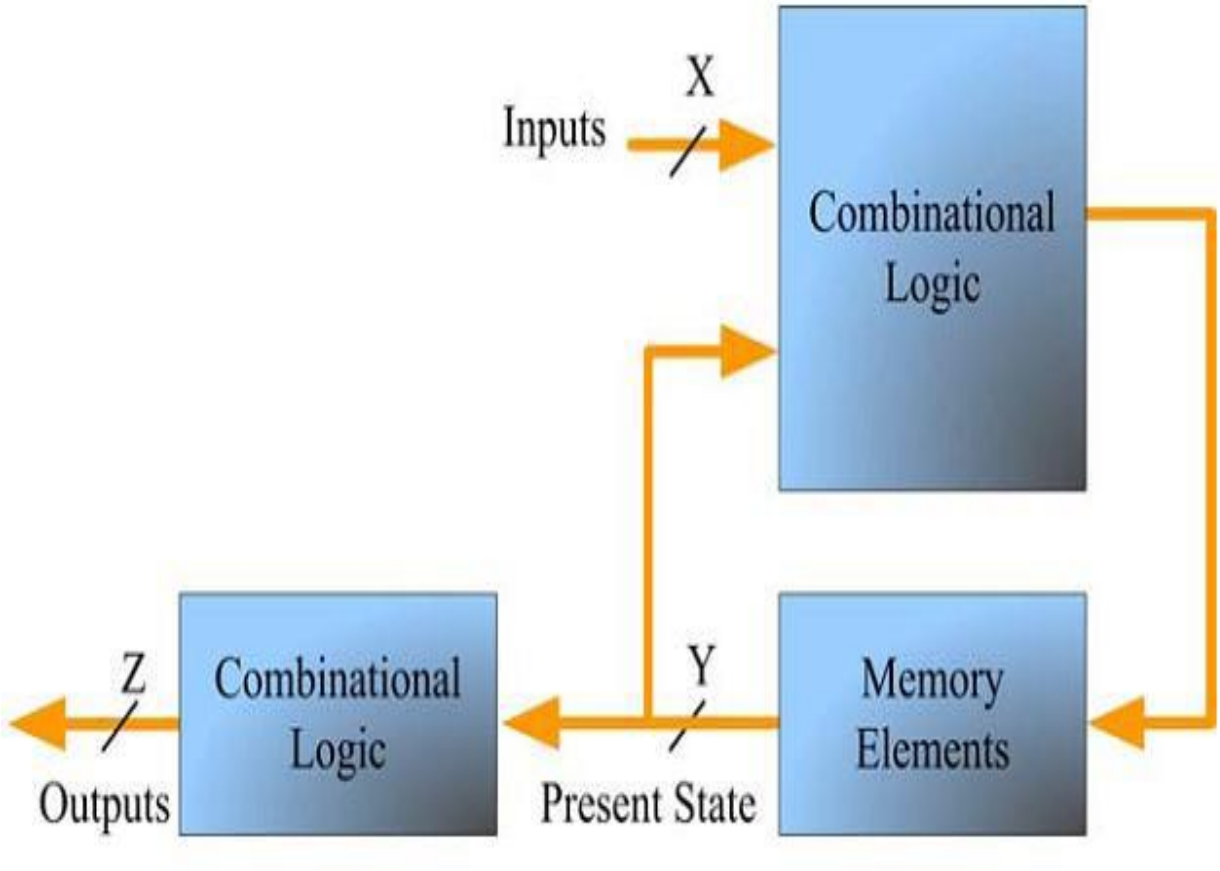


Mealy Machine

MOORE MACHINE

A **Moore machine** is a finite-state machine whose output values are determined solely by its current state. (This is in contrast to a Mealy machine, whose output values are determined both by its current state and by the values of its inputs.) The state diagram for a Moore machine associates an output value with each state (in contrast to the state diagram for a Mealy machine, which associates an output value with each transition edge).

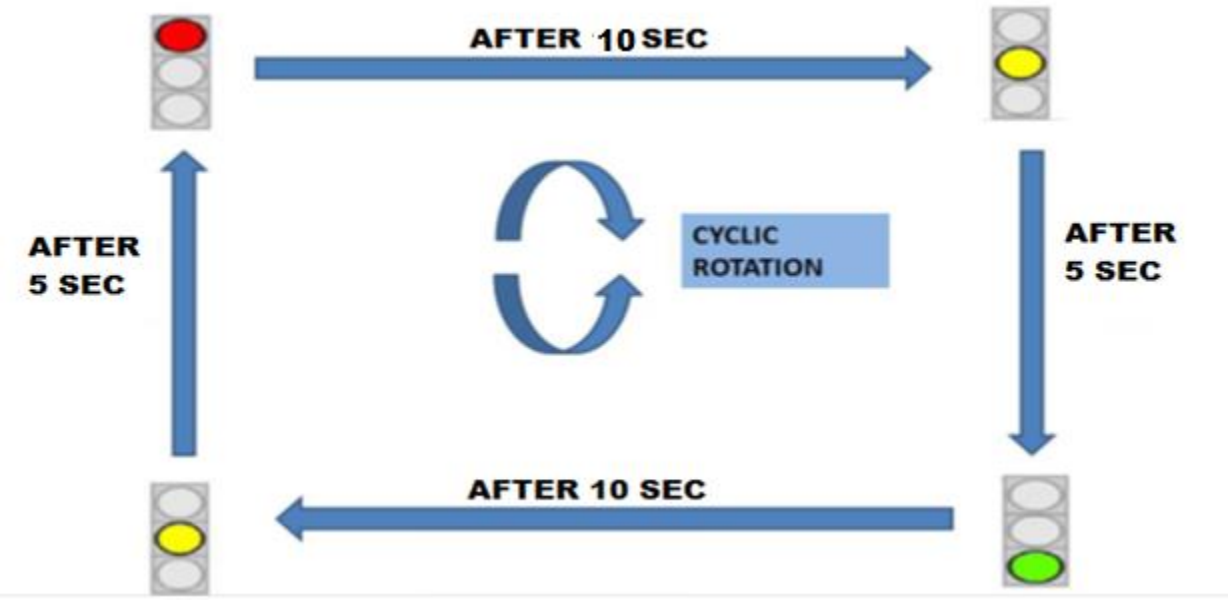
A combinational logic block maps the inputs and the current state into the necessary flip-flop inputs to store the appropriate next state just like Mealy machine. However, the outputs are computed by a combinational logic block whose inputs are only the flip-flops state outputs. The outputs change synchronously with the state transition triggered by the active clock edge.



Moore machine

3.2 TIMING SETTING

Time is an important criterion that must be set accurately and wisely, so that any dangerous situations, for example car accident can be avoided at the intersection. The timing settings for the TLC are as follows:



PROGRAM FLOW DIAGRAM

3.3 SOURCE CODE

3.3.1. TLC CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity final_traffic_light is
  Port (
                clk,rst : in  STD_LOGIC;
        r,g,b,vs,hs: out  STD_LOGIC);
end final_traffic_light;


architecture Behavioral of final_traffic_light is


type state is (st1,st2,st3,st4,st5,st6,st7,st8,st9);
signal count:integer range 0 to 20;
signal pstate,nstate:state;
CONSTANT tr : INTEGER := 10;
CONSTANT ty : INTEGER := 5;
CONSTANT tg : INTEGER := 10;
SIGNAL t: INTEGER RANGE 0 TO 20;
signal clkd,nr,ny,ng, er,ey,eg, sr,sy,sg, wr,wy,wg: std_logic;
signal m : integer range 0 to 25000000;
   component vga_display1 is
    Port ( nr,ny,ng, er,ey,eg, sr,sy,sg, wr,wy,wg, clk,rst : in  STD_LOGIC;
        r,g,b,vs,hs : out  STD_LOGIC);
          end component;
```

```
begin
----------------------------------------
process(clk,rst)
begin
if(rst='1') then
m<=0;
clkd<='1';
elsif(clk'event and clk='1') then     ----------CLK DIVIDER
if(m=25000000) then                   ----------CLK=50MHZ
clkd<=not clkd;                       ----------CLKD=1MHZ
m<=0;
else
m<=m+1;
end if;
end if;
end process;
-----------------------------------------------
process(clkd,rst)
begin
if(rst='1')then
count<=0;
pstate<=st1;

elsif(clkd'event and clkd='1')then
count<= count +1;
```

```
if (count=t) then
pstate<=nstate;
count<=0;
end if;
end if;
end process;

process (pstate,clkd)
begin
if(clkd'event and clkd='1')then
case pstate is
when st1 =>
nr<='1'; er<='1';sr<='1';wr<='1';
ny<='0'; ey<='0';sy<='0';wy<='0';
ng<='0'; eg<='0';sg<='0';wg<='0';
t<=tr;nstate<=st2;

when st2 =>
nr<='0'; er<='1';sr<='1';wr<='1';
ny<='0'; ey<='0';sy<='0';wy<='0';
ng<='1'; eg<='0';sg<='0';wg<='0';
t<=tg;nstate<=st3;

when st3 =>
nr<='0'; er<='0';sr<='1';wr<='1';
ny<='1'; ey<='1';sy<='0';wy<='0';
```

```
ng<='0'; eg<='0';sg<='0';wg<='0';
t<=ty;nstate<=st4;

when st4 =>
nr<='1'; er<='0';sr<='1';wr<='1';
ny<='0'; ey<='0';sy<='0';wy<='0';
ng<='0'; eg<='1';sg<='0';wg<='0';
t<=tg;nstate<=st5;
when st5 =>
nr<='1'; er<='0';sr<='0';wr<='1';
ny<='0'; ey<='1';sy<='1';wy<='0';
ng<='0'; eg<='0';sg<='0';wg<='0';

t<=ty;nstate<=st6;

when st6=>
nr<='1'; er<='1';sr<='0';wr<='1';
ny<='0'; ey<='0';sy<='0';wy<='0';
ng<='0'; eg<='0';sg<='1';wg<='0';
t<=tg;nstate<=st7;

when st7 =>
nr<='1'; er<='1';sr<='0';wr<='0';
ny<='0'; ey<='0';sy<='1';wy<='1';
ng<='0'; eg<='0';sg<='0';wg<='0';
t<=ty;nstate<=st8;
```

```
when st8 =>
nr<='1'; er<='1';sr<='1';wr<='0';
ny<='0'; ey<='0';sy<='0';wy<='0';
ng<='0'; eg<='0';sg<='0';wg<='1';
t<=tg;nstate<=st9;

when st9 =>
nr<='0'; er<='1';sr<='1';wr<='0';
ny<='1'; ey<='0';sy<='0';wy<='1';
ng<='0'; eg<='0';sg<='0';wg<='0';
t<=ty;nstate<=st2;

when others =>
nr<='1'; er<='1';sr<='1';wr<='1';
end case;
end if;
end process;

dout:vga_display1 port map( nr,ny,ng, er,ey,eg, sr,sy,sg, wr,wy,wg ,clk,rst,r,g,b,vs,hs);
end Behavioral;
```

3.3.2. CRT CODE

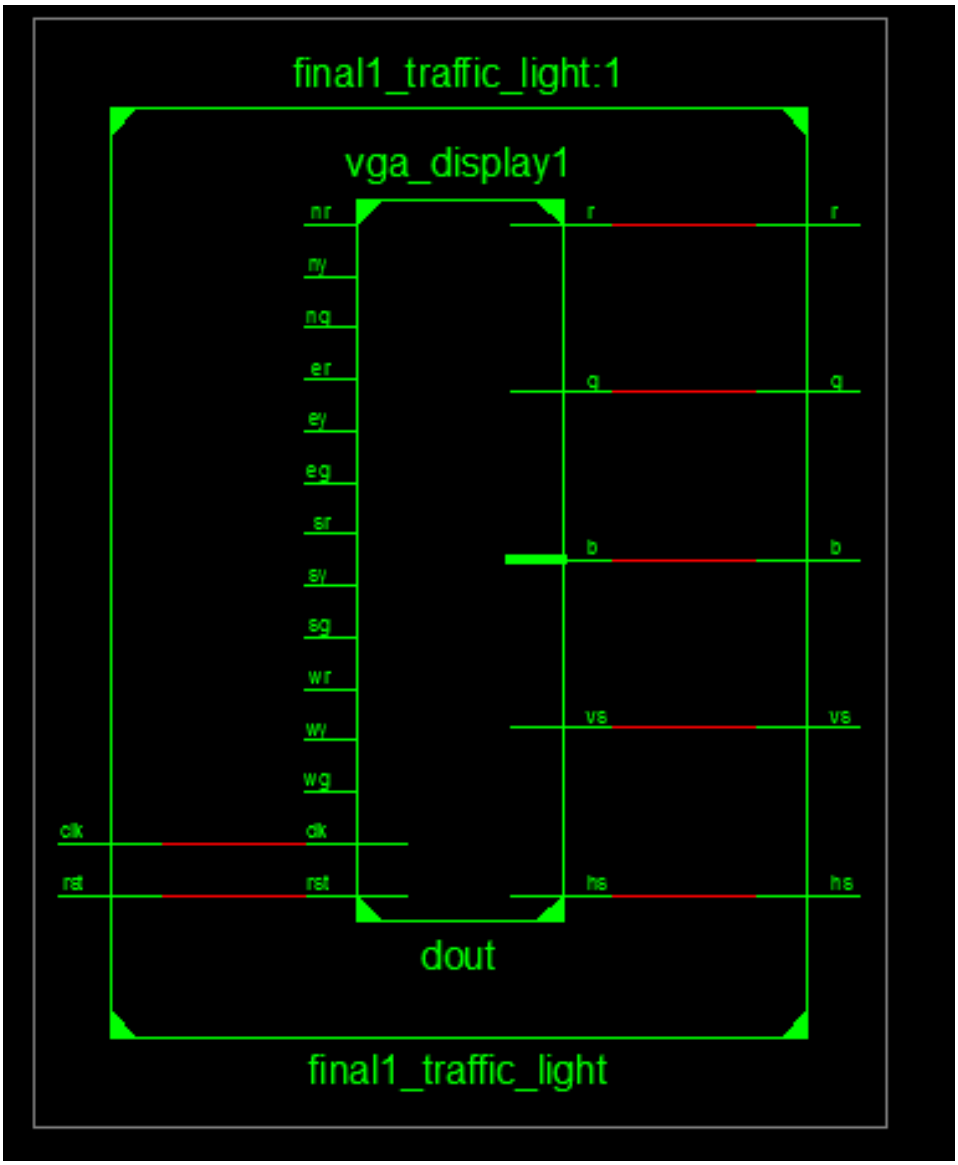
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity vga_display1 is
 Port ( nr,ny,ng, er,ey,eg, sr,sy,sg, wr,wy,wg, clk,rst : in  STD_LOGIC;
        r,g,b,vs,hs : out  STD_LOGIC);
end vga_display1;
architecture Behavioral of vga_display1 is
signal crtclk : std_logic;
signal vc,hc : integer range 0 to 850:= 0;
signal c : std_logic := '0';
begin
----------clk divider
process(clk,rst)
begin
if clk'event and clk = '1' then
c <= not c;
end if;
end process;
crtclk <= c;
-------------------------
process (crtclk,rst,nr,ny,ng, er,ey,eg, sr,sy,sg, wr,wy,wg,vc,hc)
begin
if rst = '1' then
hs <= '0';vs <= '0';r <= '0';g<= '0';b <='0';
elsif crtclk'event and crtclk = '1' then
```

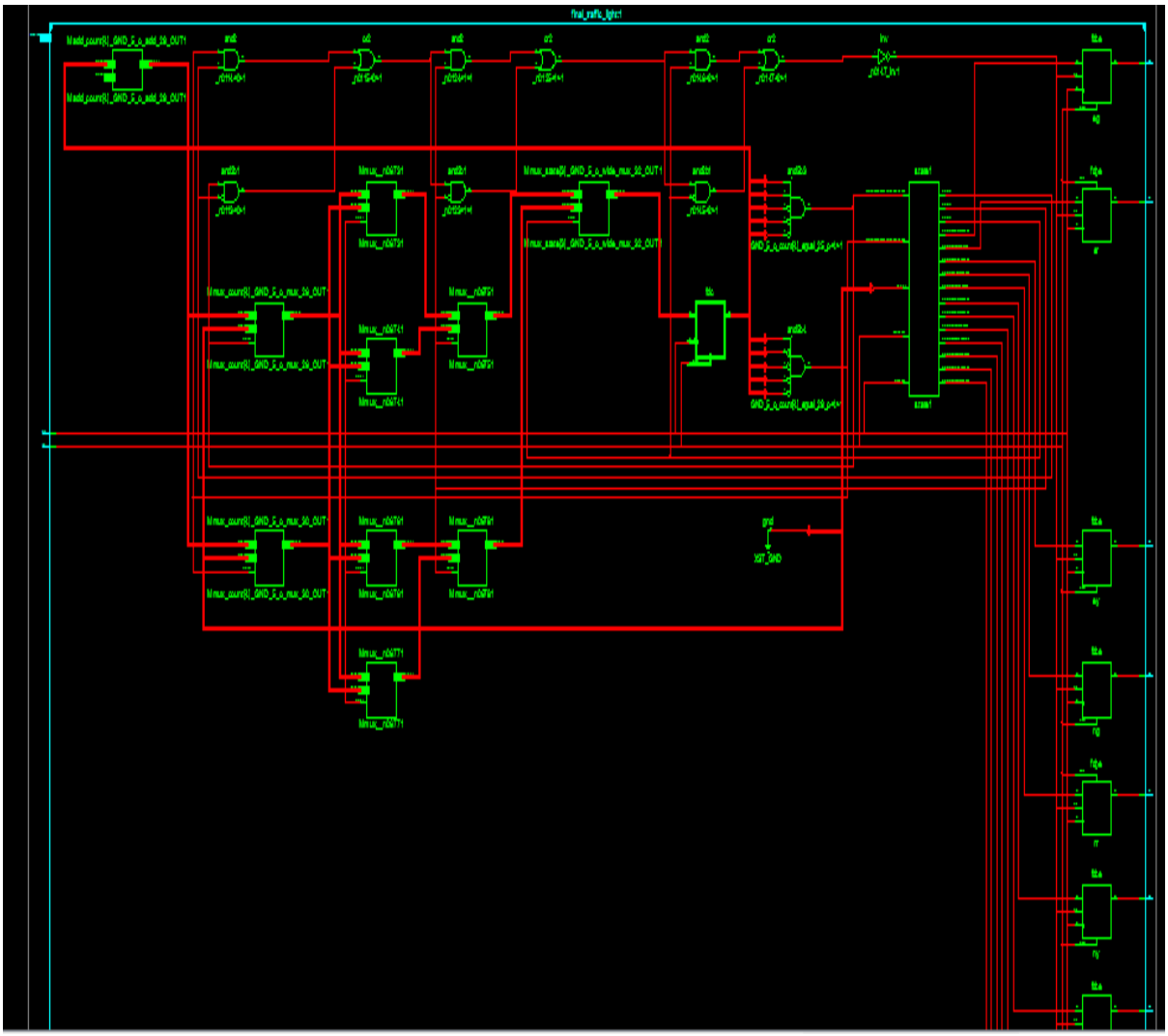
```
if vc = 528 then
vc <= 0;
elsif hc = 808 then
hc <= 0;
vc <= vc + 1;
else
hc <= hc + 1;
end if;

if hc > 664 and hc < 760 then
hs <= '0';
else
hs <= '1';
end if;

if vc > 482 and vc < 484 then
vs <= '0';
else
vs<= '1';
end if;
-------------------------------------------------------
if (ng='1' and hc > 270  and hc < 370 and vc > 0 and vc < 60 )then
r <= '0';g<= '1';b <='0';
elsif (ny='1' and hc > 270  and hc < 370 and vc > 60 and vc < 120 )then
r <= '1';g<= '1';b <='0';
elsif (nr='1' and hc > 270  and hc < 370 and vc > 120 and vc < 180)then
```

```
r <= '1';g<= '0';b <='0';
elsif (sr='1' and hc >270 and hc <370 and vc > 300 and vc < 360)then
r <= '1';g<= '0';b <='0';
elsif (sy='1' and hc >270 and hc <370 and vc > 360 and vc < 420)then
r <= '1';g<= '1';b <='0';
elsif (sg='1' and hc >270 and hc <370 and vc > 420 and vc < 480)then
r <= '0';g<= '1';b <='0';
elsif (er='1' and hc > 370 and hc < 460 and vc > 190 and vc < 290)then
r <= '1';g<= '0';b <='0';
elsif (ey='1' and hc > 460  and hc < 550 and vc > 190 and vc < 290 )then
r <= '1';g<= '1';b <='0';
elsif (eg='1' and hc > 550  and hc < 640 and vc > 190 and vc < 290 )then
r <= '0';g<= '1';b <='0';
elsif (wg='1' and hc > 0 and hc < 90 and vc > 190 and vc < 290)then
r <= '0';g<= '1';b <='0';

elsif (wy='1' and hc > 90 and hc < 180 and vc > 190 and vc < 290)then
r <= '1';g<= '1';b <='0';
elsif (wr='1' and hc > 180 and hc < 270 and vc > 190 and vc < 290)then
r <= '1';g<= '0';b <='0';
else
r<= '0';g<= '0';b <='0';
end if;
end if;
end process;
end Behavioral;
```

4.RESULTS

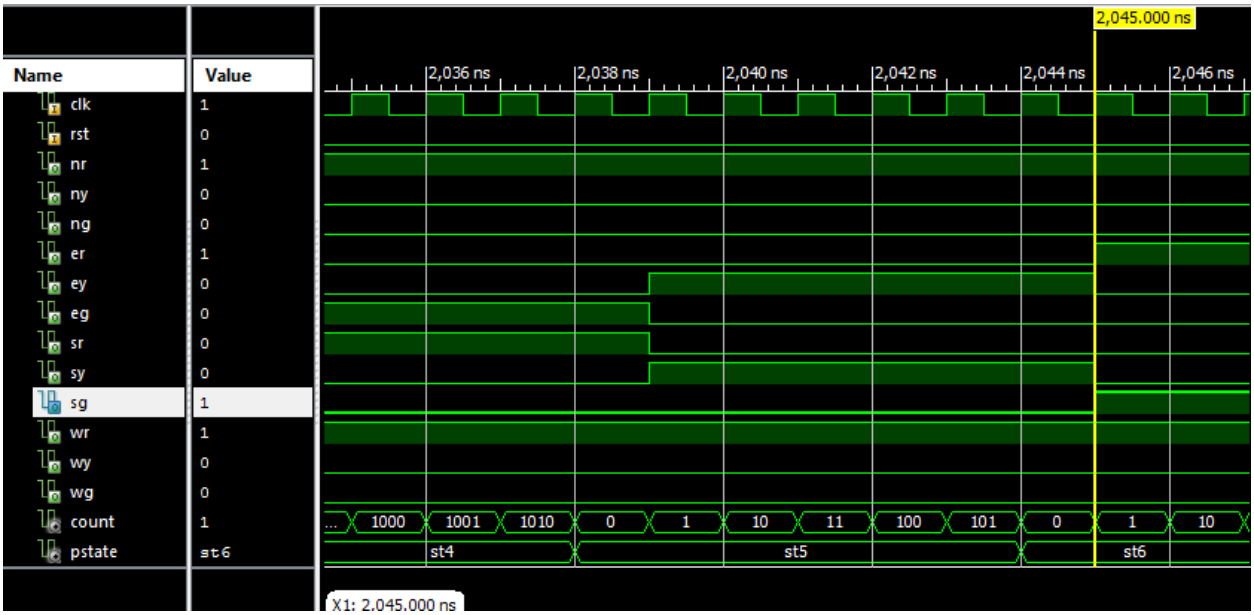
4.1 RTL SCHEMATIC:

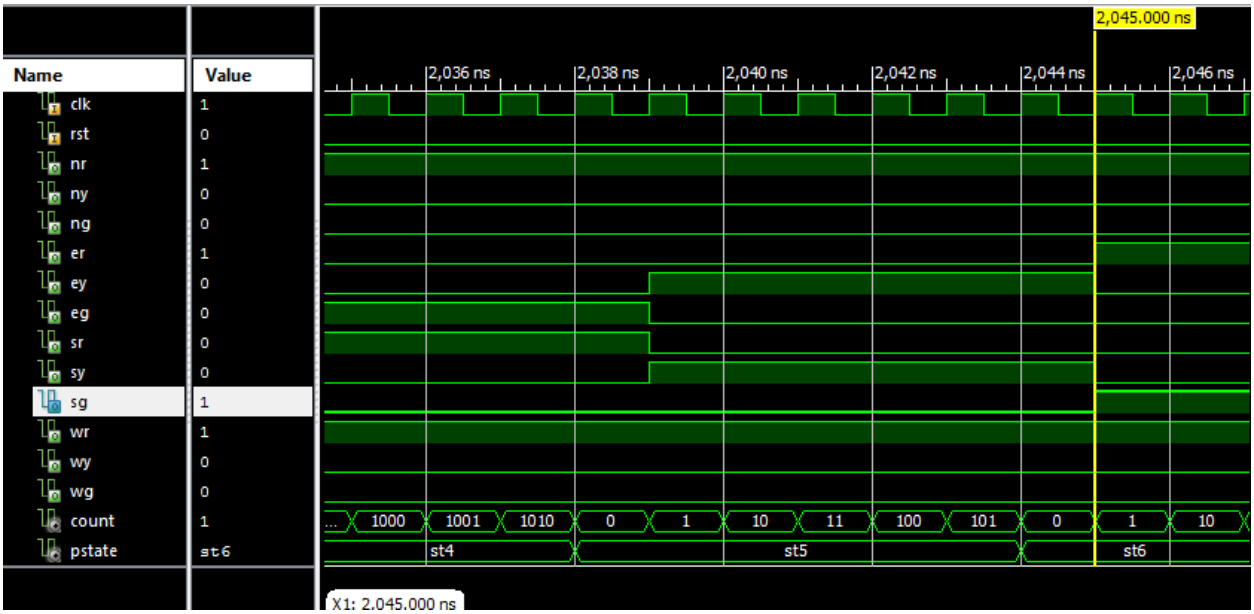




RTL Schematics zoom in

4.1.2 SIMULATION RESULTS:





5. ADVANTAGES

- Speed of operation.
- Low power consumption.
- Easy to implement.
- Less area.
- Less wiring complexity.

APPLICATIONS

Ease to control the traffic signals at any junction by using Intelligent Transport System (ITS) by using Field Programmable Gate Array (FPGA).The code is written in VHDL design pattern and synthesis is done in XILINX of version 14.6.

FUTURE ENHANCEMENT

For future works, the TLC design will include pedestrian crossing lights with the intention that the reliability of the design can be much enhanced. Lastly, a comprehensive and an exceptional TLC design can be made into an embedded circuit board to control the actual traffic flow in the city's traffic intersections.

6. CONCLUSION

An FPGA design of TLC with twelve traffic lights has been simulated. One of the advantage of this design over the existing method is the waiting time of driver during off-peak hour has been reduced, means that the normal design cycle (using fixed-time technique) has been reduced notably, thus ameliorate reliability and flexibility of the TLC.

An FPGA design of a 24-hour traffic light controller system of a 4 way 4 sided roads structure with twelve traffic lights has been simulated, implemented and tested. The system has been designed using VHDL, and implemented on hardware using XILINX Spartan kit. The functionality of this design can be easily enhanced.

Some of these functions are to control more than 12 traffic lights. Also, to allow the user to assign the time for each traffic light (i.e., minimum time to be Green), adding more sensors on each road to count the number of cars in each road and check for the longer queue to increase the timer for that road.

7. REFERENCES

- El-Medany, W.M., & Hussain, M.R. (2007) FPGA-Based Advanced Real Traffic Light Controller System Design.
  Proceeding of 4th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
- Chan, S.S., Deshpande, R.S. & Rana, J.G. (2009) Design of Intelligent Traffic Light Controller Using Embedded System.
  Proceeding of 2nd International Conference on Emerging Trends in Engineering and Technology
- Liu, Y. & Chen. X. (2009) Design of Traffic Lights Controlling System Based on PLC and Configuration Technology.
  Proceeding of International Conference on Multimedia Information Networking and Security
- Wikipedia, "Programmable Logic device,"
  http://en.wikipedia.org/wiki/Programmable_logic_device.
- Wikipedia, "Traffic light,"
  http://en.wikipedia.org/wiki/Traffic_light.