A project on

FPGA Based Traffic Lights Controller Using Verilog

Project For Fulfilment

Laboratory Work-II (First Unit), Fourth Year.

Submitted by

Name: Forhad Hossain
Roll: 1515032
Reg:1166
Session: 2015-2016

Dept of Electrical & Electronic Engineering,
Islamic University, Kushtia.

CERTIFICATION

I am pleased to certify that the project work entitled **"FPGA Based Traffic Lights Controller using Verilog"** submitted by **Md. Forhad Hossain**, Roll no. 1515032, Reg. no. 1166, Session: 2015-2016 has performed under my supervision for the fulfilment of the project based on laboratory work-II (First Unit), fourth year of department of Electrical & Electronic Engineering. Islamic University, Kushtia.

Signature........................

Dr. Md. Monjarul Alam
Professor
Dept. of Electrical & Electronic Engineering.
Islamic University, Kushtia.

ABSTRACT

At intersections, traffic is controlled by turning on/off red, green and orange lights in a specific sequence. A traffic light controller is designed to generate a series of digital data, called switching sequences, which can be used to control traffic lights in a specific sequence at a typical four-way intersection. In addition, it is proposed to carry out day and night space activities. It plays an increasingly important role in the modern management and control of urban traffic to reduce traffic accidents and traffic congestion on the roads. It is a sequential machine that is analyzed and programmed through a multi-step process. A device that includes the analysis, timing and synchronization of existing semaphore controllers in series, and demonstration of flashing operation and synthetic sequence. The methods used in this project are circuit design, coding, simulation, synthesis and hardware implementation. For this project, XILINX software was chosen to design the diagram using schematic editing, write the code using a Verilog HDL (Hardware Description Language) text editor, and implement the circuit using programmable logic.

1.0 INTRODUCTION

1.1 Need for Traffic Light Controller

Traffic congestion is a major problem in many modern cities around the world. Traffic congestion has created many critical problems and challenges in the largest and most populous cities. Traveling to different places within the city has become more difficult for passengers using public transport. Because of these congestion problems, people waste time, lose opportunities and get frustrated. Traffic congestion directly affects businesses. Due to traffic congestion, the productivity of workers decreases, business opportunities disappear, deliveries are delayed and thus costs increase. To solve these congestion problems, we need to build new facilities and infrastructure, but at the same time make it smart. The only downside to building new roads into farms is that it makes the environment more congested. That's why we have to change the system instead of creating a new infrastructure twice. Therefore, many countries are trying to manage their existing transport systems to improve mobility, safety and traffic flows to reduce the demand for vehicle use. Therefore, much research has been done on the traffic light system to overcome some complex traffic phenomena, but the existing research is limited on the current traffic system under well-traveled traffic scenarios. The distribution time is fixed at the intersection from east to west or in the opposite direction and from north to south. Field Programmable Gate Arrays (FPGAs) are widely used for rapid prototyping and proof-of-concept design, and are also used in electronic systems when custom IC masks become prohibitively expensive to produce due to small quantities. Many system designs previously built on custom silicon VLSI are now implemented on field programmable gate arrays. This is due to the high cost of manufacturing custom VLSI masks, especially for small quantities.

1.2 What is FPGA?

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects.

1.3 How does FPGA work?

An FPGA consists of internal hardware blocks with user-programmable interfaces to customize operation for a specific application. These interfaces can be reprogrammed, allowing the FPGA to accommodate design changes or support a new application during part of its lifetime. The FPGA is programmed during the manufacturing process, but can be reprogrammed later to reflect changes made to the device.

An FPGA is based on a matrix of configurable logic blocks (CLBs) connected by programmable interfaces. CLBs form the basic element of an FPGA and contain two 16-bit function generators, one 8-bit function generator, two registers (flip-flops or latches) and reprogrammable routing controllers (multiplexers). CLBs are used to execute macros and other scheduled actions.

Unlike CPUs, FPGAs are capable of parallel operation, so that different processing operations do not compete for the same resources. Each independent task is assigned to a special part of the chip and can operate independently without the influence of other logic blocks. So adding features does not change the performance of one part of the program.

The internal configurations of an FPGA are defined by software or, as it is often called, "firmware". FPGAs can be reprogrammed in the field as application or functionality requirements change. FPGAs are designed to be programmed using a hardware description language such as Verilog HDL or VHDL.

1.4 Internal Structure of FPGA

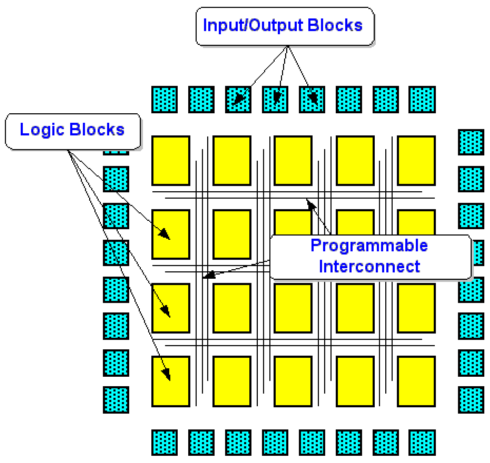


Fig 1: Internal Structure of FPGA

FPGA consists of three major components:

1. Programmable Logic Blocks (PLB).
2. Programmable Input/Output (I/O) block.
3. Programmable Interconnects.

Programmable Logic Blocks (PLB)

Basic logic operations are implemented by components known as programmable logic blocks. Configurable logic blocks, or CLBs, are the name given to logic blocks in Xilinx-based FPGAs, whereas logic array blocks, or LABs, are the name given to a similar structure in Altera-based FPGAs. These logic units also aid the FPGA's storage capabilities. Anything, including a transistor, a NAND gate, multiplexers, a lookup table (LUT), or even processors, can be a basic logic block. Lookup table (LUT)-based logic blocks are used by both Xilinx and Altera to implement logic and storage operations.

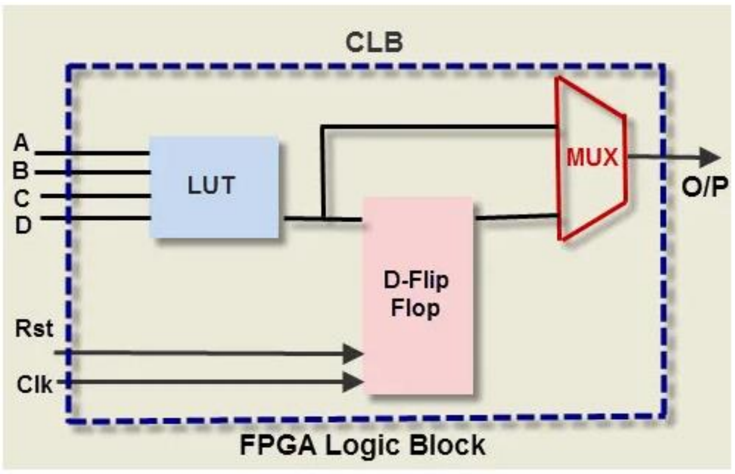
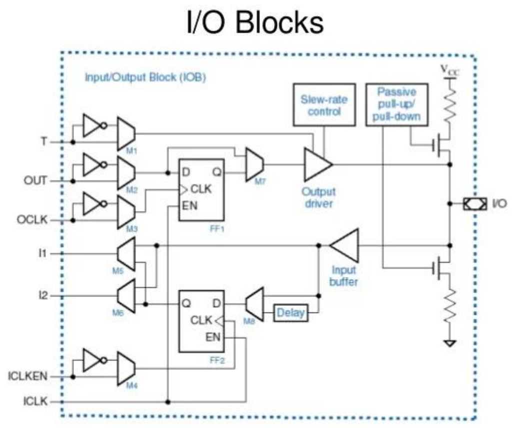


Fig 2: A simple logic block

Programmable Input/Output (I/O) block

A programmable I/O block is used to bring signals to and from the chip. They are used to connect logic blocks and routing architecture to external components. The I/O platform and the logic circuit surrounding it form an I/O cell. It consists of an input buffer and an output buffer with tri-state and open-collector output controls.



Programmable Interconnects

Programmable interconnects or routing connect logic blocks and I/O blocks to form a user-defined design unit. FPGA routing consists of wire segments of varying lengths that can be interconnected by electrically programmable switches. The logic block density used in an FPGA depends on the length and number of wire segments used in routing. It consists of multiplexers, pass transistors and three-state buffers. Gate transistors and multiplexers are used to connect logic elements in a logic cluster.

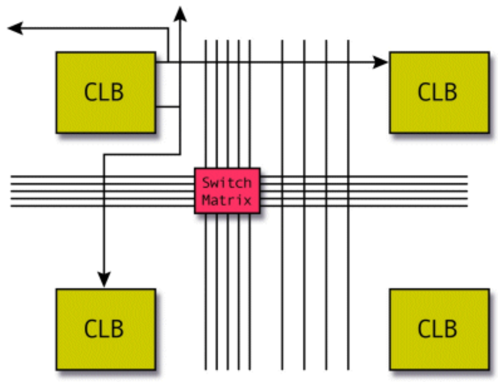


Fig 4: Interconnect network

1.5 Programming the FPGA

Verilog HDL

Verilog, standardized like IEEE 1364, is a hardware description language (HDL) used to model electronic systems.

It was created by Prabhu Goel, Phil Moorby, and Chi-Lai Huang and Douglas Warmke in late 1983 and early 1984.

Verilog is a portmanteau of the words "control" and "logic".

Before we get into the actual programming, we need to familiarize ourselves with the specific types of modeling for digital system design, and we will examine each modeling with the corresponding Verilog code.

The types of modelling are as follows:

1. Dataflow Modelling.
2. Hierarchical Modelling.
3. Structural level Modelling.
4. Switch level Modelling (Special/Auxiliary type).

2.0 DESIGN OF TRAFFIC LIGHT CONTROLLER

The traffic light controller can be designed starting from arbitrary assumptions. Initially, northbound traffic will be allowed to move, and then eastbound, southbound and westbound traffic will be allowed to move. The advantage of writing a Traffic Light Controller program is that it is easy to change the program as per the requirement. it is assumed that traffic should be allowed on the main road more and less time on side roads; then the clock is distributed so that the main road has more clock period and the side roads have less, because the traffic on the main road is heavy compared to the side road. Generally, the TLC system has three lights in each direction (red, green and yellow), where a red light means stop traffic, a green light allows traffic, and a yellow light stops traffic a number of times.

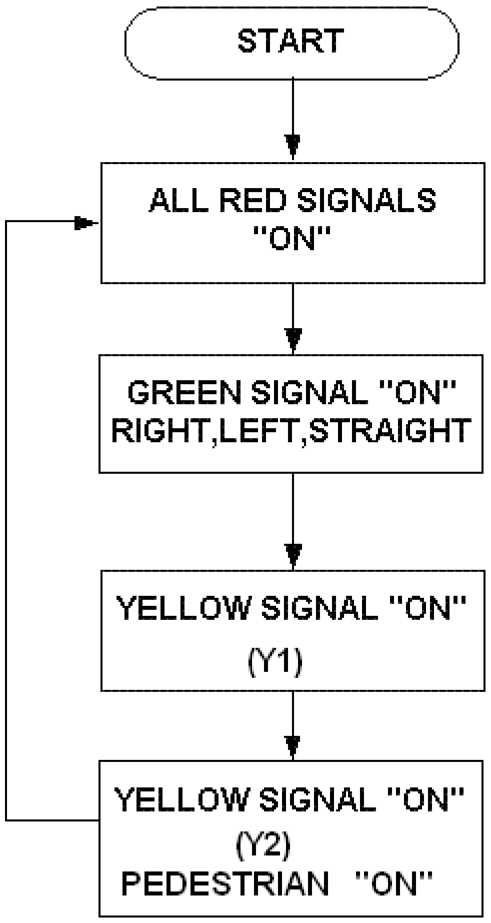

START
ALL RED SIGNALS "ON"
GREEN SIGNAL "ON" RIGHT,LEFT,STRAIGHT
YELLOW SIGNAL "ON" (Y1)
YELLOW SIGNAL "ON" (Y2) PEDESTRIAN "ON"

Fig 2.1 TLC Flow Chart

2.1 Explanation of Traffic Light Controller

In this structure, there are four traffic signals, represented by R1, R2, R3 and R4 to be controlled. All the four signals have same priority as they all are main roads.

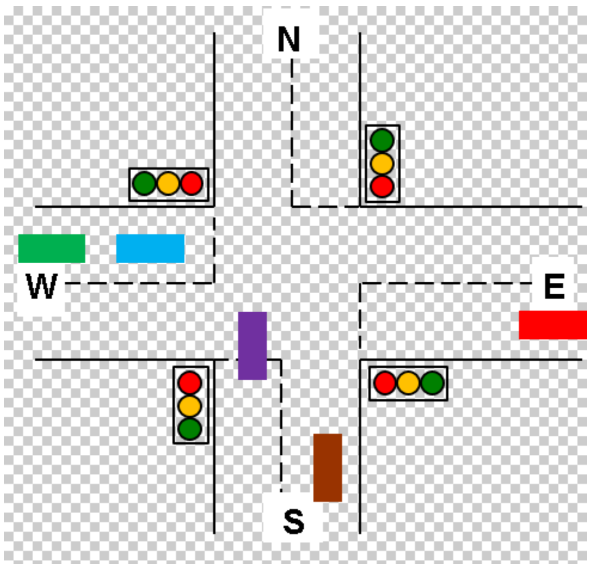


Fig 2.2 Traffic Signals at Junction

First, the signal controller is in reset mode where the path signal (R1) is green while all other paths R2, R3 and R4 are red. We defined this state as S0.
Then, the controller sends control to state S1, where R1 is yellow, while all other signals are still only red. In this mode, the controller checks whether the sensor on path R2, which is X2, is low or not. If the sensor gives a low signal that there is no traffic on the road, then the signal on road R2 is bypassed, moving the controller to state S4, where the signal on road R3 is reversed, while the rest of the signs are red. On the other hand, if there is traffic on road R2, the controller is sent to state S2, which makes the signal of road R2 green, and the rest of the signals are red only if the driver is in state S2 after the green signal is. shown the indicator of the road sign R2 will change from green to yellow, while all other signs will only remain red, which is the operation of S3 mode.
Again, when the controller is in state S3, it checks the response of sensor X3 through R3. If the sensor output is low, the system control moves to state S6, bypassing the signal action on path R3, otherwise, control moves to the corresponding next state S4.
If the traffic light turns green in S4 on road R3, the signs on roads R1, R2 and R4 remain red. Control is then transferred to state S5. When the controller is in state S5, it controls the output of sensor X4 through R4. Depending on the output of X4, the next state change occurs accordingly. If it is low, the control goes to S0, bypassing the operation of the signal on the track R4, otherwise the control is S6. When the controller is in S5 mode, the signal on R3 changes from green to yellow.
When the controller is in S6 mode, traffic sign R4 turns green, while all signs change or remain only red. The controller then moves to state S0.

In S7 mode, the R4 signal changes from green to yellow. At the same time, the output of the first R1 sensor, which is X1, is checked. If the signal is low, the control directly goes to state S2, otherwise the control goes to the default state S0.
These spaces are optional. The user can set the number of modes, the order of the lights and the delay. This is one of the biggest advantages of this project.

2.2 TLC State Diagram

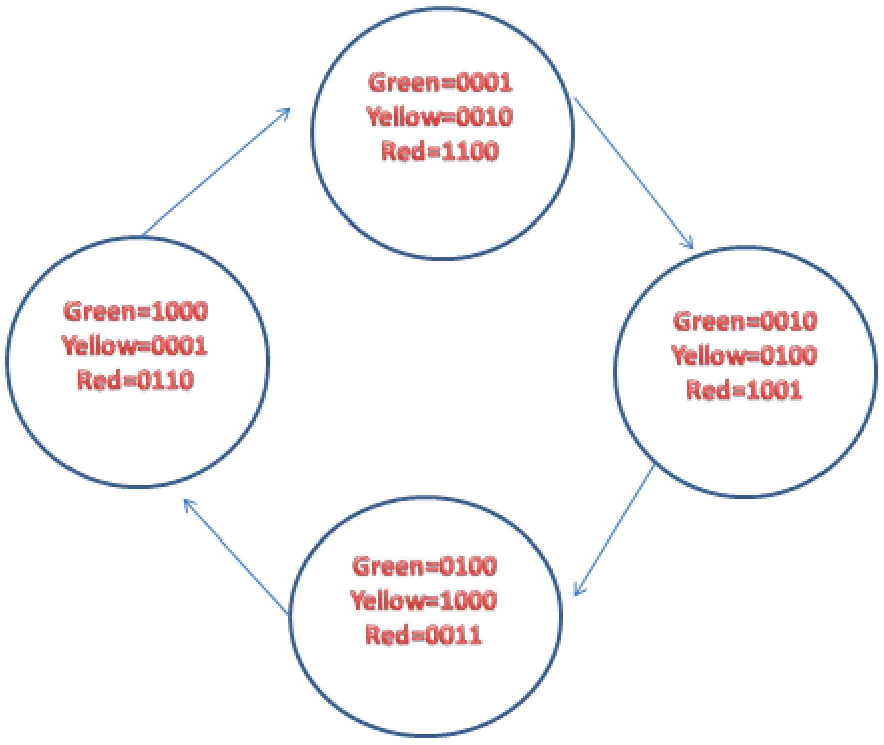


Fig 2.3 TLC State Diagram

The TLC state diagram shown in Figure 2.3 illustrates that whenever cnt=00 and dir=00, the green light is on for a few seconds in the north direction and the red indicator light is on in all other directions ie. west, south and east. on. If cnt=01 and dir=00, the yellow light (y1) will be on for a few seconds, and if cnt=01, the yellow light (y2) and walking north will be on, and then dir will be increased by one and cnt will be reset . . So, if cnt=00 and dir=01, the eastbound green light will be on for a few seconds and all other directions will be red. If cnt=01 and dir=01, the yellow light (y1) will be on for a few seconds, and if cnt=01, the yellow light (y2) will be on and the east pedestrian path will be on, and then dir. will be incremented by one and cnt will be set to zero. So whenever cnt=00 and dir=10, south direction green light will be on for few seconds and all other directions will be red. If cnt=01 and dir=10, the yellow light (y1) will be on for a few seconds, and if cnt=01, the yellow light (y2) and pedestrian south will be on, and then dir will increase by one and cnt will be set to zero So whenever cnt=00 and dir=11, there will be a green light for a few seconds in the west and all the red lights will be on in the other directions. If cnt=01 and dir=11, the yellow light (y1) will be on for a few seconds, and if cnt=01, the yellow light (y2) and the pedestrian west will be on, and dir will be set to 00 and cnt will be zero . This sequence repeats and the traffic flow is controlled using time slots showing in all four directions.

3.0 SIMULATION

3.1 Verilog Code Design

```
module traffic_lc(input rst , clk , output reg [2:0]n,s,e,w);
parameter ns_red = 2'b00;
parameter we_tr = 2'b01;
parameter we_red = 2'b10;
parameter ns_tr = 2'b11;
reg [32:0] count;
reg[1:0] state;


always@(posedge clk or posedge rst)
begin
if(rst)
begin
state<=ns_red;
count<=32'd0;
end
else
begin
case(state)
ns_red : begin
        if (count == 32'h0bebc200)
        begin
        state = we_tr;
        count = 32'd0;
        end
        else
        begin
        count = count+1;
        state = ns_red;
        end
        end
```

```
we_tr    : begin
        if (count == 32'h05f5e100)
        begin
        state = we_red;
        count = 32'b000_000_000_000;
        end
        else
        begin
        count = count+1;
        state = we_tr;
        end
        end

we_red    : begin
          if (count == 32'h0bebc200)
          begin
          state = ns_tr;
          count = 16'b000_000_000_000;
          end
          else
          begin
          count = count+1;
          state = we_red;
          end
          end

ns_tr      : begin
          if (count == 32'h05f5e100)
          begin
          state = ns_red;
          count = 32'd0;
          end
          else
          begin
          count = count+1;
          state = ns_tr;
          end
          end
endcase
end
end
```

```
always@(state)
begin
case(state)
ns_red : begin
         n<=3'b100;
         s<=3'b100;
         e<=3'b001;
         w<=3'b001;
         end
we_tr : begin
        n<=3'b100;
        s<=3'b100;
        e<=3'b010;
        w<=3'b010;
        end
we_red : begin
         n<=3'b001;
         s<=3'b001;
         e<=3'b100;
         w<=3'b100;
         end

ns_tr : begin
        n<=3'b010;
        s<=3'b010;
        e<=3'b100;
        w<=3'b100;
        end
endcase
end
endmodule
```

Test Bench

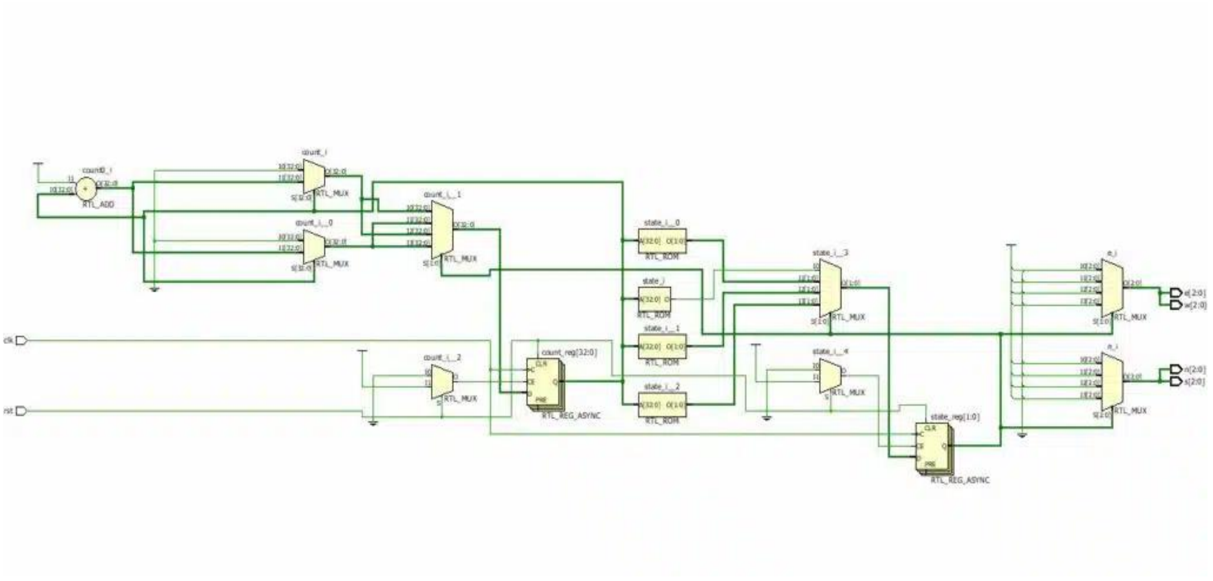
```
module test_traffic_lc;
reg clk,rst;
wire [2:0]n,s,e,w;
traffic_lc DUT (rst , clk , n,s,e,w);
initial
clk = 1'b0;
```

```
always#5 clk = ~clk;

initial begin
  $dumpfile("dump.vcd");
 $dumpvars;
#0 rst = 1'b0;
#10 rst = 1'b1;
#15 rst = 1'b0;
$finish;
end
endmodule
```

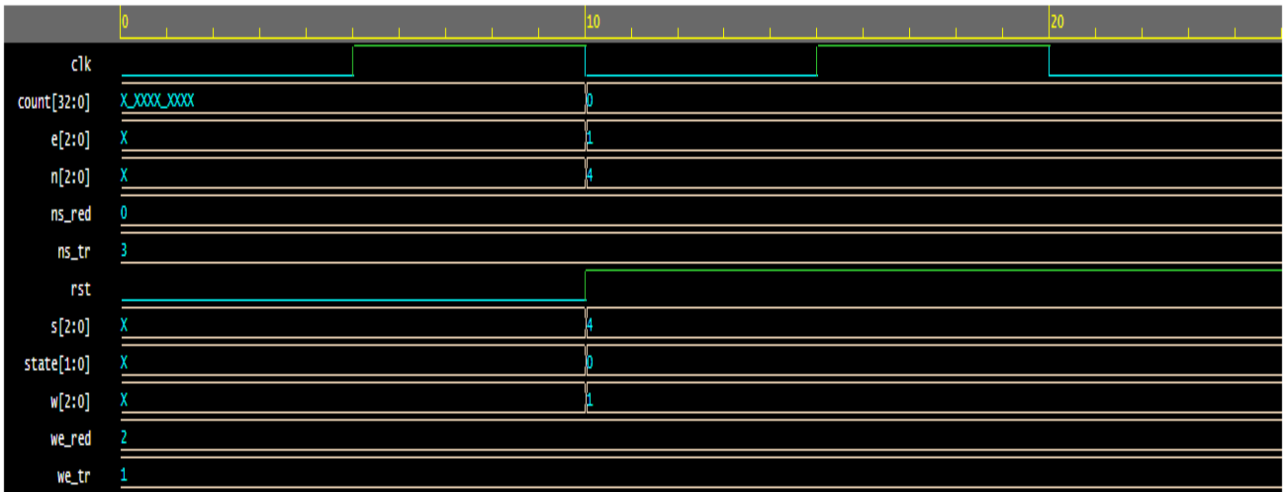
3.2 RTL Schematic

The below figure shows the RTL Schematic of the Traffic Light Controller.



3.3 Wave Form

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



CONCLUSION

The traffic light control system helps to control the smooth movement of vehicles. Many obstacles, high level accidents happen every day. So the semaphore controller prevents such events. However, many areas or small towns do not have traffic lights. Therefore, many accidents happen in these places. Therefore, the primary purpose of such a facility is to control and maintain the area.

The main idea behind this simple project is traffic management. It can be used to avoid vehicle collisions and traffic jams. This project is only a one-way traffic controller, although it can be further modified. Traffic intensity is detected and time is reserved for passing traffic accordingly. Verilog HDL is used to describe the circuit, the code is generated and simulated with xilinx14.5.

We can extend this idea to 4 highways and highways. We can also design 8 traffic lights. Many development ideas for future work can be implemented, such as using solar energy (independent energy source, i.e. saving electricity). Using the GPRS card as a further step in the development work and choosing the best route for emergency and police vehicles.

REFERENCES

1. E. Geetha, V. Viswanadha, and G. Kavitha, "Design of intelligent auto traffic signal controller with emergency override", International journal of engineering science and innovative technology (IJESIT), Vol. 3 , Issue 4, pp. 670-675, July 2014.
2. K. Vidhya, anf A. Banu, "Density based traffic signal system", International journal of innovative research in science, engineering, and technology, Vol. 3, Issue 3, pp. 2218-2223, March 2014.
3. I. Isa, N. Shaari, A. Fayeez, and N. Azlin, "Portable wireless traffic light system (PWTLS)", International journal of research in engineering and technology, Vol. 3, Issue 2, pp. 242-247, Feb 2014.
4. Shaohan Hu,Lu Su,Hengchag Liu,Hongyan Wang and Tarek F.Abdelzaher “SmartRoad: Smartphone-Based Crowd Sensing for Traffic Regulator Detection and Identification”, ACM Transactions on Sensor Networks, Vol. 11, No. 4, Article 55, Publication date: July 2015.
5. https://www.studocu.com/in/document/visvesvaraya-technological university/verilog/traffic-light-controller-using-verilog/31081846
6. https://www.fpga4student.com/2016/11/verilog-code-for-traffic-light-system.html
7. https://github.com/Arjun-Narula/Traffic-Light-Controller-using-Verilog