>>> MINIPROJECT <<<

DESIGN TRAFICLIGHT CONTROLER AND IMPLEMENTATION ON FPGA USING VHDL

SANKET BASAVRAJ MUTNALE

ABSTRACT :-

Traffic light controller is a set of rules and instructions that drivers, pilots, train engineers, 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 VHDL. 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. It is easy to use and the cost for the same is also less as compared to the others.

INTRODUCTION :-

The normal function of Traffic Light Controller requires more than slight control and coordination to ensure that traffic moves as smoothly and safely as possible and that pedestrians are protected when they cross the roads. This traffic jam directly impacts the productivity of the workers, traders, suppliers and in all effecting the market and raising the prices of the commodities in a way. To solve these traffic related problems, we have to build new conveniences & infrastructure but at the same time make it smart. The only drawback of making new roads on facilities is that it makes the surroundings more congested, but then this will make a way to have new ways to ease the traffic.. We have to build new facilities and infrastructure making its use smarter for its efficient use. Thus we have to implement the Street light controller using the VHDL code and the later is simulated using Xilinx software and simulation is done with the ISE Simulator.

WHAT IS VHDL :-

VHDL stands for Very High-Speed Integration Circuit HDL (Hardware Description Language). It is an IEEE (Institute of Electrical and Electronics Engineers) standard hardware description language that is used to describe and simulate the behavior of complex digital circuits.

WHAT IS 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). A Spartan FPGA from Xilinx 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 interwired in different configurations.

DESIGN FLOW GRAPH :-

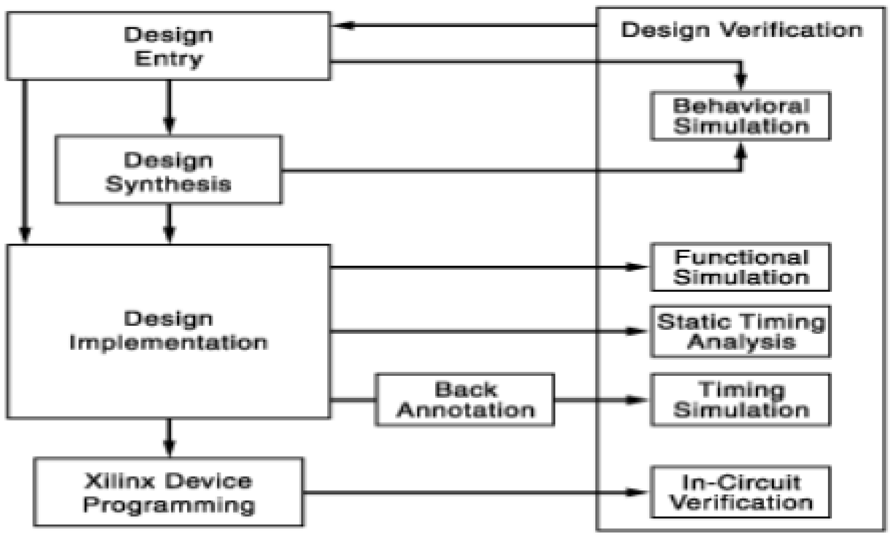


FIG. BLOCK DIAGRAM

VHDL CODE :-

1. **TOP MODULE :-**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;


entity tpm is
        port (clk : in  STD_LOGIC;
                        r1,g1,y1,r2,g2,y2,r3,g3,y3,r4,g4,y4 : out  STD_LOGIC;
                        L1,L2,L3,L4 : out STD_LOGIC);
end tpm;

architecture Behavioral of tpm is


component master_clock
        port(i1 : in  STD_LOGIC;
                        o : out  STD_LOGIC);
end component;

component lights
        port(i2 : in STD_LOGIC;
                        mr1,mg1,my1,mr2,mg2,my2,mr3,mg3,my3,mr4,mg4,my4 : out
STD_LOGIC);

end component;

component left
        port(clk_3 : in STD_LOGIC;
                L11,L12,L13,L14 : out  STD_LOGIC);

end component;

signal q:std_logic;


begin

ms1: master_clock port map(clk,q);
ligh1: lights port map(q,r1,g1,y1,r2,g2,y2,r3,g3,y3,r4,g4,y4);
left1: left port map(q,L1,L2,L3,L4);

end Behavioral;
```

2. CLOCK

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;


entity master_clock is
    Port ( i1 : in  STD_LOGIC;
                                                  o : out  STD_LOGIC);
end master_clock;

 architecture Behavioral of master_clock is

signal temp :STD_LOGIC_vector (23 downto 0):="000000000000000000000000";
-- signal clk_1 :STD_LOGIC;
--signal a: integer:= 0;

begin

process (i1)
                         begin
                                      if (i1' event and i1='1' )then
                                                  if ( temp = "100110001001011010000000")then
                                                              temp <= "000000000000000000000000";
                                      else
                                                  temp <= temp + 1;
                                      end if ;
                                      end if;
                                      end process;
             process (temp)
                         begin
                                      if (temp >= "000000000000000000000000" and temp <=
"010011000100101101000000")then
                                                  o <='1';
                                      else
                                                  o <= '0';
                                      end if;
                         end process;


end Behavioral;
```

3. LIGHT DESIGN :-

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity lights is
   Port (i2 : in std_logic;
          mr1,mg1,my1,mr2,mg2,my2,mr3,mg3,my3,mr4,mg4,my4: out  STD_LOGIC);
end lights;

architecture Behavioral of lights is
type state_type is (a,b,c,d,e,f,g,h);
signal state,next_state:state_type;
signal rst:std_logic :='0';
signal j:integer:= 0;

begin


        process(rst,i2)
                begin
                        if(i2' event and i2 = '1')then
                                if (next_state = state)then
                                                j<= j+1;
                                else
                                        j<=0;end if;

                                if rst = '1' then

                                state<= a;
                                else
                                state <= next_state;
                                end if;
                        end if;

        end process;
process(state,j)
                begin
                        case state is
                                when a=>
                                        if (j <= 20) then
                                                mr1<= '0'; mg1<= '1'; my1<= '0';
                                                mr2<= '1'; mg2<= '0'; my2<= '0';
                                                mr3<= '1'; mg3<= '0'; my3<= '0';
                                                mr4<= '1'; mg4<= '0'; my4<= '0';
                                                next_state <= a;
                                        else
                                                next_state <= b;


                                        end if;
```

```
when b =>

        if j <= 10 then
                mr1<= '0'; mg1<= '0'; my1<= '1';
                mr2<= '1'; mg2<= '0'; my2<= '0';
                mr3<= '1'; mg3<= '0'; my3<= '0';
                mr4<= '1'; mg4<= '0'; my4<= '0';
                next_state <= b;
        else
                next_state <= c;


        end if;


when c =>

        if j<= 20 then
                mr1<= '1'; mg1<= '0'; my1<= '0';
                mr2<= '0'; mg2<= '1'; my2<= '0';
                mr3<= '1'; mg3<= '0'; my3<= '0';
                mr4<= '1'; mg4<= '0'; my4<= '0';
                next_state <= c;
        else
                next_state <= d;


        end if;


when d =>

        if j <= 10 then
                mr1<= '1'; mg1<= '0'; my1<= '0';
                mr2<= '0'; mg2<= '0'; my2<= '1';
                mr3<= '1'; mg3<= '0'; my3<= '0';
                mr4<= '1'; mg4<= '0'; my4<= '0';
                next_state <= d;
        else

        next_state <= e;
        end if;



when e =>

        if j <= 20 then
                mr1<= '1'; mg1<= '0'; my1<= '0';
```

```
                                        mr2<= '1'; mg2<= '0'; my2<= '0';
                                        mr3<= '0'; mg3<= '1'; my3<= '0';
                                        mr4<= '1'; mg4<= '0'; my4<= '0';
                                        next_state <= e;
                               else

                                        next_state <=f;

                               end if;

                      when f =>

                               if j <= 20 then
                                        mr1<= '1'; mg1<= '0'; my1<= '0';
                                        mr2<= '1'; mg2<= '0'; my2<= '0';
                                        mr3<= '0'; mg3<= '0'; my3<= '1';
                                        mr4<= '1'; mg4<= '0'; my4<= '0';
                                        next_state <= f;
                               else

                                        next_state <=g;

                               end if;

                      when g =>

                               if j <= 10 then
                                        mr1<= '1'; mg1<= '0'; my1<= '0';
                                        mr2<= '1'; mg2<= '0'; my2<= '0';
                                        mr3<= '1'; mg3<= '0'; my3<= '0';
                                        mr4<= '0'; mg4<= '1'; my4<= '0';
                                        next_state <= g;
                               else
                                        next_state <=h;

                               end if;

                      when h =>
                               if j <= 10 then
                                        mr1<= '1'; mg1<= '0'; my1<= '0';
                                        mr2<= '1'; mg2<= '0'; my2<= '0';
                                        mr3<= '1'; mg3<= '0'; my3<= '0';
                                        mr4<= '0'; mg4<= '0'; my4<= '1';
                                        next_state <= h;
                               else
                                        next_state <= a;

                               end if;
                      end case;
             end process;
end Behavioral;
```

4. LEFT LIGHT :-

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity left is
    Port ( clk_3 : in  STD_LOGIC;
           L11,L12,L13,L14 : out  STD_LOGIC);
end left;

architecture Behavioral of left is
--signal u :integer:=0;

begin
process(clk_3)
            begin


            if          clk_3 = '1' then
                        L11<='1';
                        L12<='1';
                        L13<='1';
                        L14<='1';
            else
                        L11<='0';
                        L12<='0';
                        L13<='0';
                        L14<='0';
            end if;
end process;
end Behavioral;
```

DESIGN FLOW :-

FPGA design flow comprises the following steps:

design entry, functional verification, time verification and Xilinx device programming. Design verification, which includes both functional verification and timing verification, takes places at different points during the design flow.

DESIGN ENTRY :-

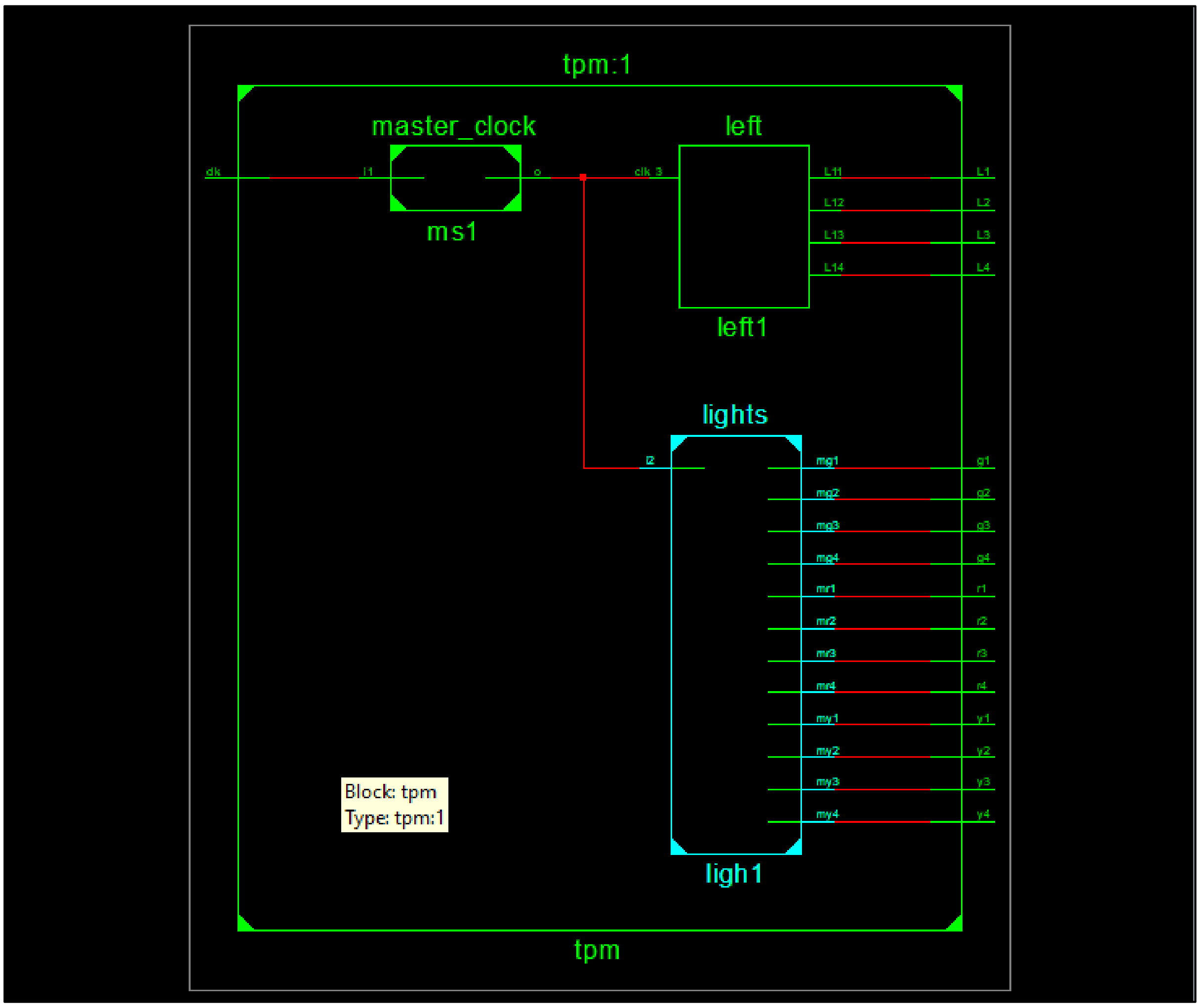
Create an ISE project as follows:

1. Create a project.
2. Create files and add them to your project, including a user constraints (UCF) file.
3. Add any existing files to your project.
4. Assign constraints such as timing constraints, pin assignments, and area constraints.

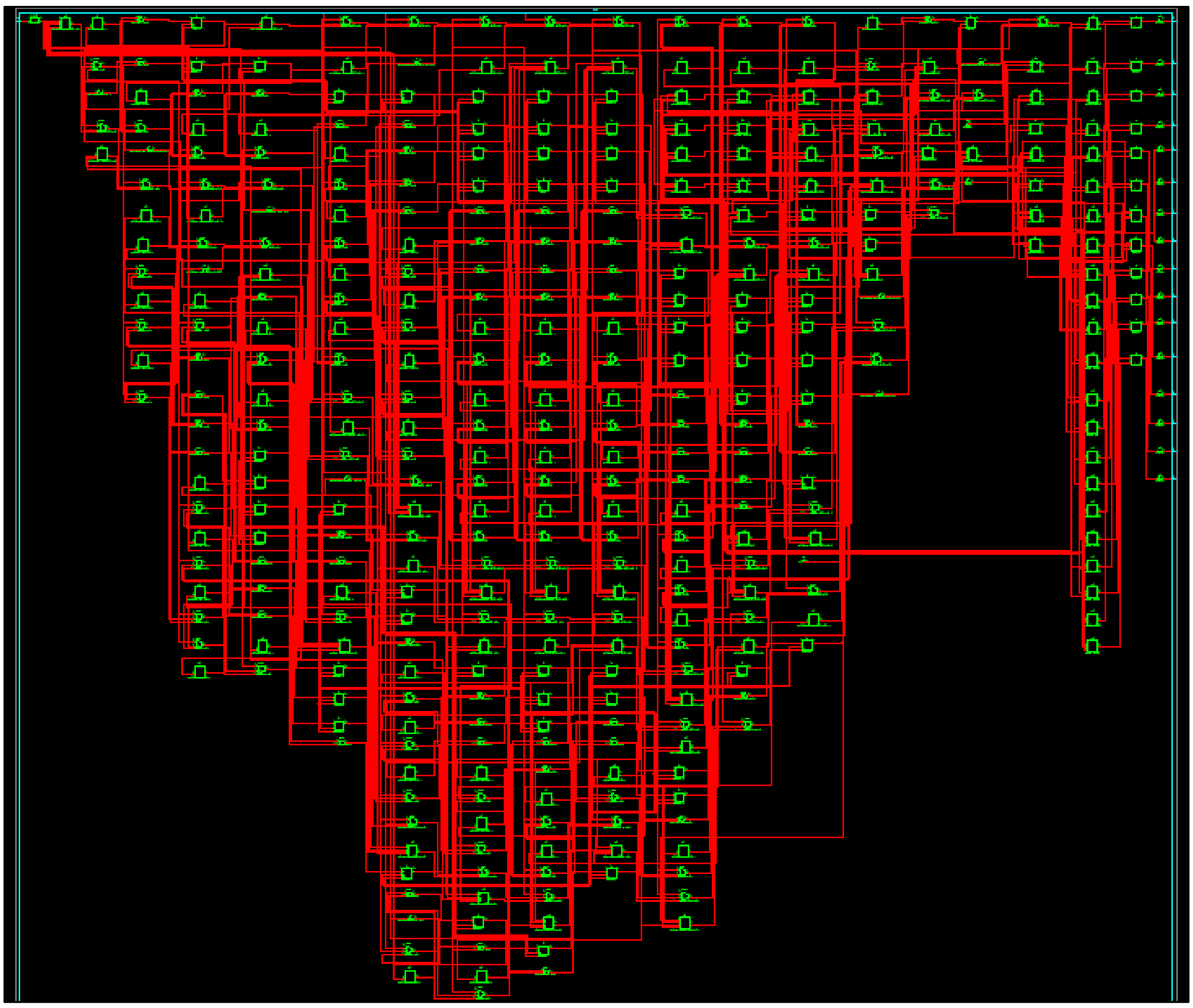
FUNCTIONAL VERIFICATION :-

1. Before synthesis, run behavioral simulation (also known as RTL simulation).
2. After Translate, run functional simulation (also known as gate-level simulation), using the SIMPRIM library.
3. After device programming, run in-circuit verification.

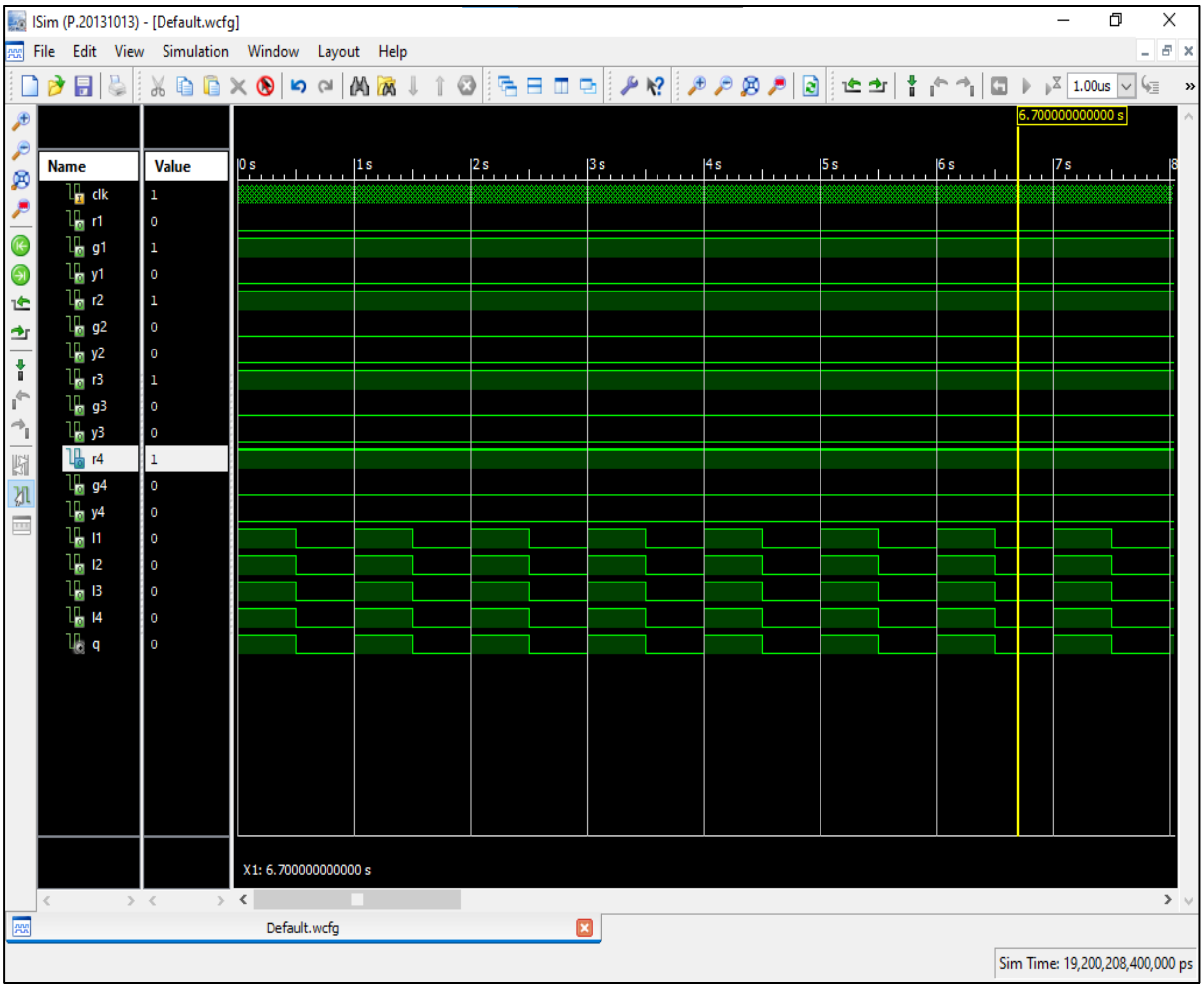
RTL SCHEMATIC :-



TECHNOLOGY SCHEMATIC :-



ISE SIMULATION :-



Name	Value
clk	1
r1	0
g1	1
y1	0
r2	1
g2	0
y2	0
r3	1
g3	0
y3	0
r4	1
g4	0
y4	0
l1	0
l2	0
l3	0
l4	0
q	0

X1: 6.700000000000 s

Sim Time: 19,200,208,400,000 ps

FPGA SPARTAN IMPLEMENTATION :-

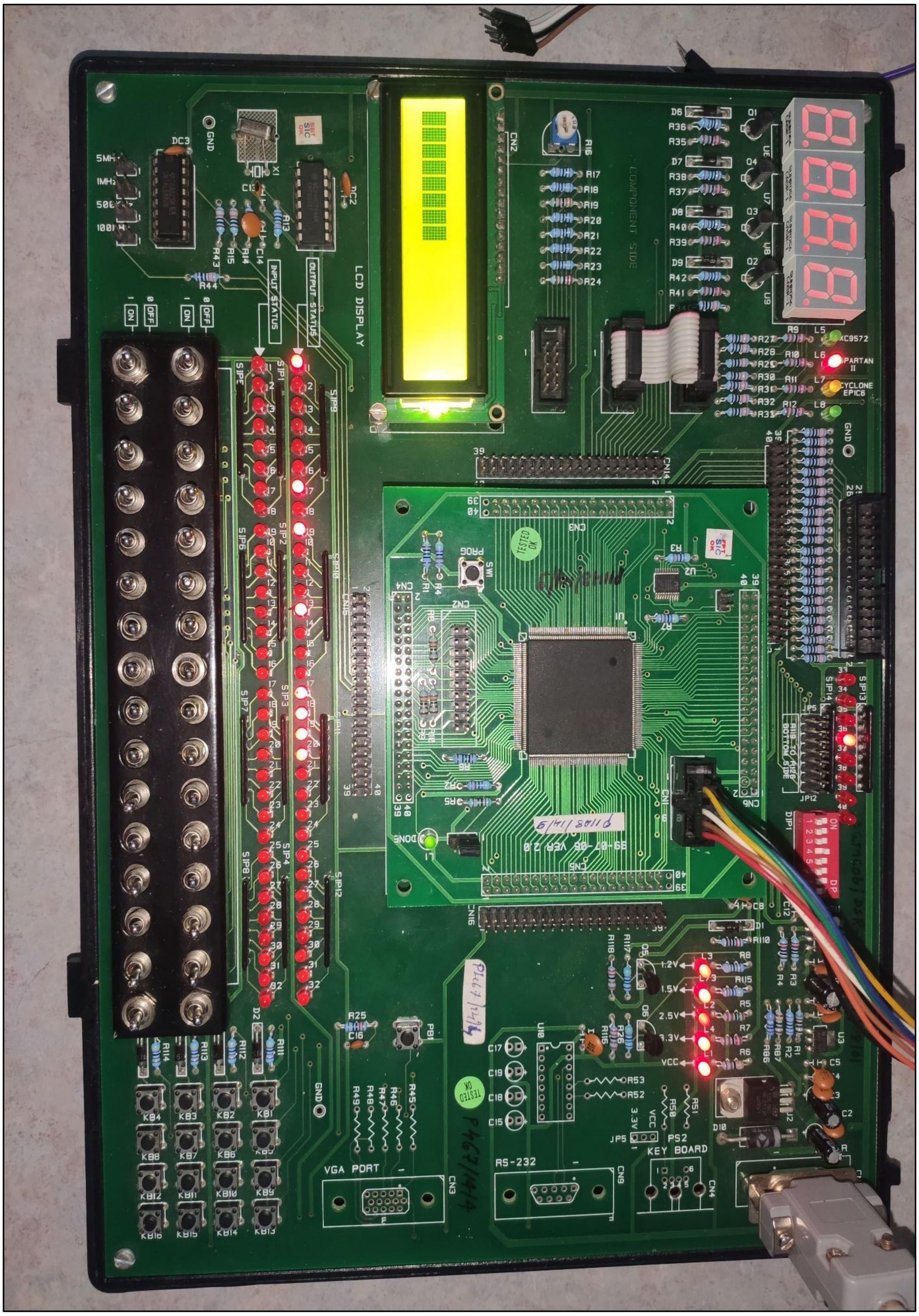
As we know that once the design part is over, it is required to implement that software part into the real time simulation. Thus FPGA board is used in order to provide the real time simulation. The Traffic Light Controller designed using Verilog HDL code and is implemented on the hardware using FPGA. The output of the Traffic light is displayed on the Spartan-X3-S400 FPGA board.

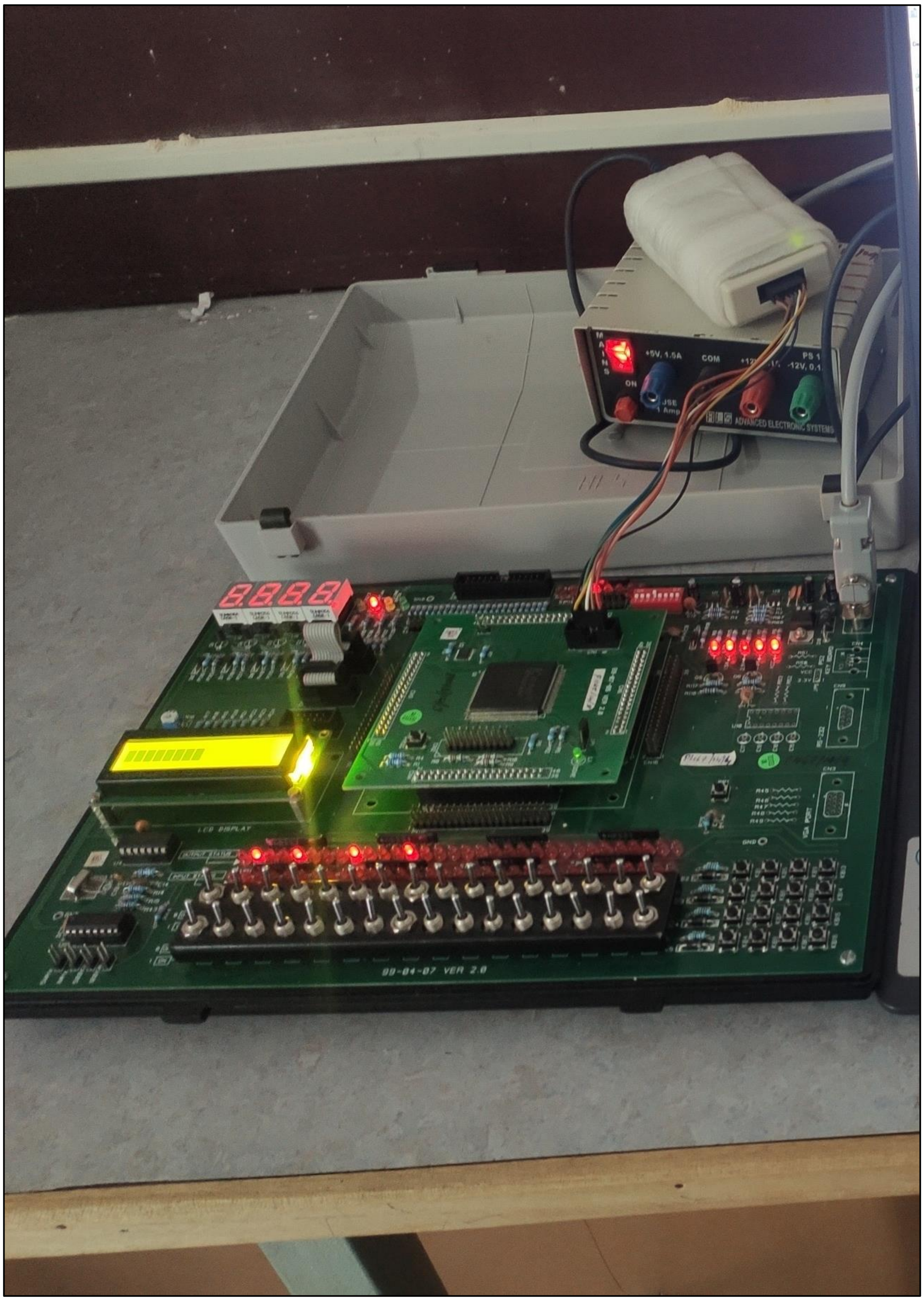
XILINX DEVICE PROGRAMMING :-

Program your Xilinx device as follows:

1. Create a programming file (BIT) to program your FPGA.
2. Generate a PROM or ACE file for debugging or to download to your device.
3. Optionally, create a JTAG file.
4. Use iMPACT to program the device with a programming cable.

FPGA IMPLEMENTATION :-




MAINS
ON
+5V, 1.5A
COM
PS 1
ADVANCED ELECTRONIC SYSTEMS
LCD DISPLAY
99-04-07 VER 2.0

ADVANTAGE :-

1. Significantly enhanced operational tools congestion to effectively manage traffic incidents.
2. It improved Public Transport service.
3. Reduction in emergency response times and safer travel.
4. Improve traffic guidance and traffic flow.
5. Reduce fuel consumption.

SUMMARY :-

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.

CONCLUSION :-

This project is based on a very effective way of optimizing traffic, with redefinition of threshold values for a real time application. This works to control traffic on four way roads according to traffic control barricades . This proposed system will be able to build a developed country with less traffic jams and it will also help the emergency vehicle to reach in time to the destination. So, this intelligent system will help us to control traffic in more autonomous way.

FUTURE SCOPE :-

The system can be expanded with smart traffic light control and congestion avoidance system during emergencies emergency cars such as fire engines and ambulance and have priority over other traffic. This system gives highest priority to emergency vehicle to pass them.

REFERENCE :-

1. Electronics Devices and Circuits, S Salivahanan and N Suresh Kumar, MC Graw Hill Education, Third Edition.
2. Fundamentals of Digital Circuits, A.Anand Kumar, Easter Economy Edition, Third Edition.
3. Stephen Brown and ZvonkoVranesic, “Fundamentals of Digital Logic with VHDL design”, Tata Mc-graw Hill
4. https://en.wikipedia.org/wiki/VHDL#:~:text=The%20VHSIC%20Hardware%20Description%20Language,%2C%20documentation%2C%20and%20verification%20purposes.
5. https://en.wikipedia.org/wiki/Field-programmable_gate_array
6. https://www.slideshare.net/nimmi_abes/traffic-light-controller-33138676

THANK YOU…