1. Title of the Project

Design a FSM that controls a traffic light system.

2. Introduction

In this report, a Traffic Light Controller will be described as a top-level view using a block diagram, as a finite state machine using a state diagram and it will also be constructedin detail in Vivado software using the IEEE Standard VHDL Language. Finite state machines can be used to model situations and problems in many fields and a traffic light controller is one of them.

This Traffic Light Controller will be able to manage four traffic lights in a road intersection as depicted in Figure 1 below. The design will be implemented with different delays between traffic light states in order to mimic a real-life situation. In other words, the RED and GREEN will stay on for longer periods of time when compared to YELLOW.

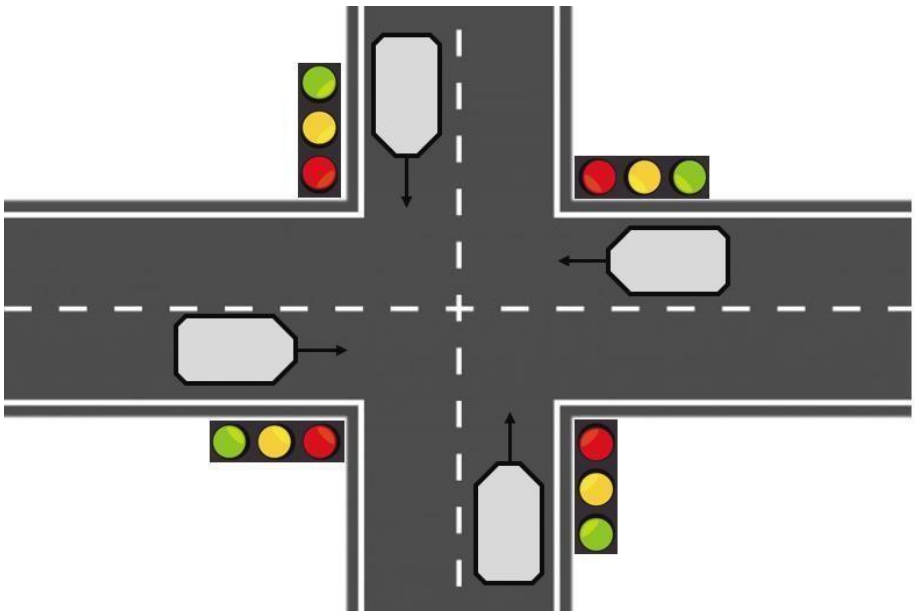


Figure 1. Four Traffic Lights in a Road Intersection

3. Theory

3.1. State Diagram

Since the Traffic Light Controller will be implemented as a finite state machine, a state diagram must be drawn first before constructing a block diagram and starting thecode implementation. A state diagram is used to describe the behavior of a system and must be composed of a finite number of states. In this project, the state diagram for the controller will consist of six states. In contrast to other FSMs, the logic flow inthis controller will be based on a counter rather than an input bit. The diagram can be seen in Figure 2 below.

The clock input to the controller will determine the delay in seconds of each state. Depending on the frequency of the clock, the delays will change. In order to calculatethe delay, the following formula is used:

$$Delay\ [in\ seconds] = count * clock\ frequency + 1$$

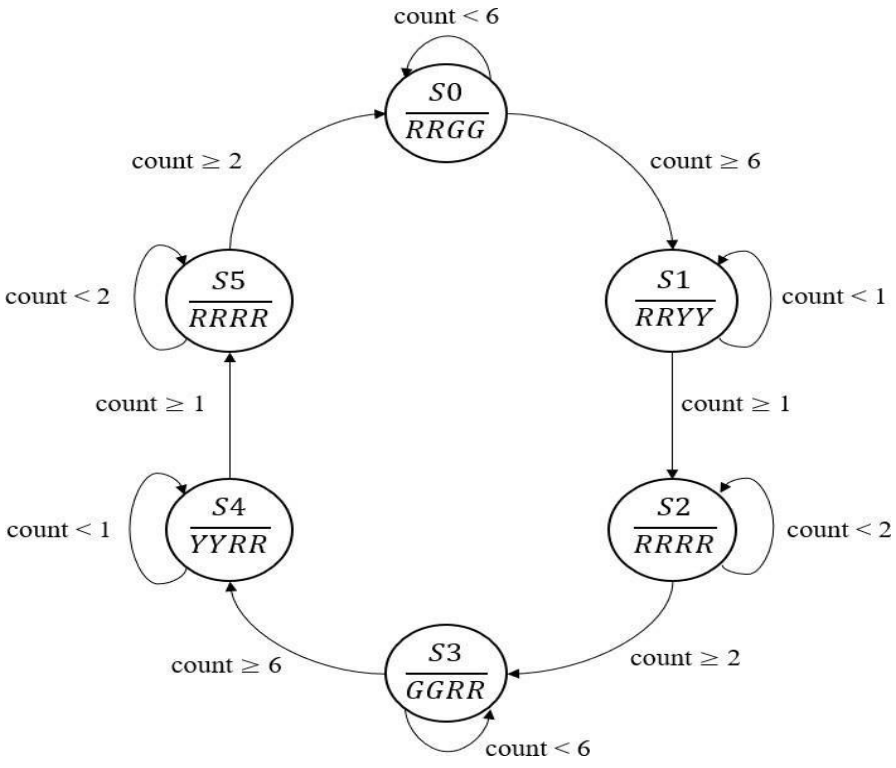


Figure 2. Moore State Diagram of the Traffic Light Controller

All of the six different states shown in the figure above correspond to a situation in the road intersection. For every case, the lights will change their state and manage thetraffic efficiently. The cars will stay at RED light for a period of 12 seconds, at YELLOW light for a period of 3 seconds and at GREEN light for a period of 7 seconds. The description of the states can be seen in Table 1 below.

State	Traffic Light 1			Traffic Light 2			Traffic Light 3			Traffic Light 4			Delays
S0	R			R					G			G	7 [s]
S1	R			R				Y			Y		3 [s]
S2	R			R			R			R			2 [s]
S3			G			G	R			R			7 [s]
S4		Y			Y		R			R			3 [s]
S5	R			R			R			R			2 [s]

Table 1. Description of States and their Delays in seconds

3.2. Block Diagram

The block diagram of the controller will show the different inputs and outputs of thesystem in order to better visualize it.

The controller will have three inputs: CLOCK, ENABLE, and RESET

It will also have four outputs: Out_1, Out_2, Out_3, and Out_4

The block diagram of the controller can be seen in Figure 3 down below.

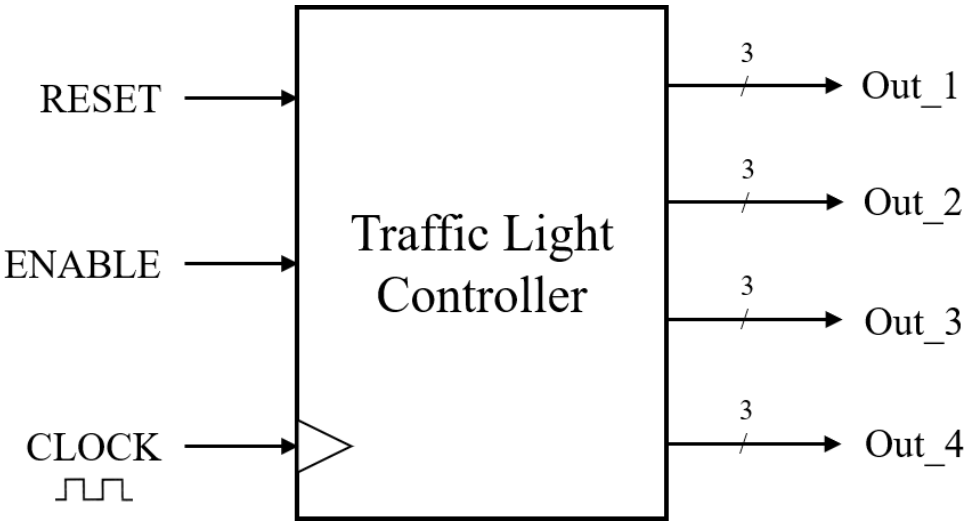


Figure 3. Block Diagram of the Traffic Light Controller

The design will have four different outputs in order to manage the four traffic lights on each road. Each of the outputs is composed of three bits with each bit controlling asingle light (RED, YELLOW, and GREEN).

In addition, in order to calculate the delays easier, the clock will have the followingfrequency:

$$f_{CLOCK} = 1\ [Hz]$$

The diagram on the next page shows how the controller can be connected to the fourtraffic lights and how the output bits control the individual lights.

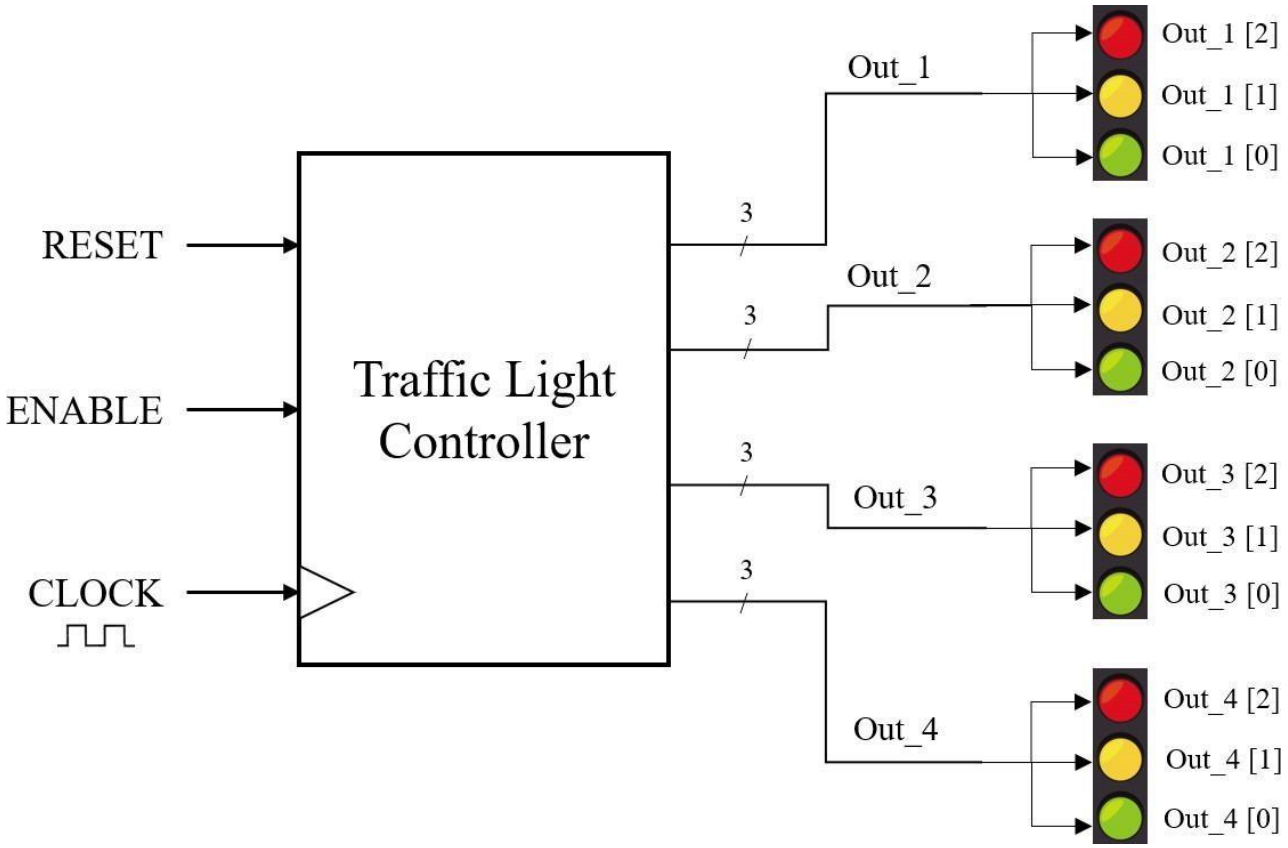


Figure 4. Controller connected to the four Traffic Lights

4. Project Code

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC_STD.ALL;


-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.
```

```
--library UNISIM;
--use UNISIM.VComponents.all;

entity Traffic_Light_Controller is
   Port ( CLOCK : in STD_LOGIC;
          RESET  :  in  STD_LOGIC;
          ENABLE : in STD_LOGIC;
          Out_1 : out STD_LOGIC_VECTOR (2 downto 0);
          Out_2 : out STD_LOGIC_VECTOR (2 downto 0);
          Out_3 : out STD_LOGIC_VECTOR (2 downto 0);
          Out_4 : out STD_LOGIC_VECTOR (2 downto 0));
end Traffic_Light_Controller;

ARCHITECTURE Controller_Arch of Traffic_Light_Controller IS
             TYPE    state_type is (S0, S1, S2, S3, S4, S5);
             SIGNAL  current_state: state_type;
             SIGNAL  count:    std_logic_vector(2 downto 0);  --3 bit counter


             --Initializing constants for each of the lights
             CONSTANT RED:     std_logic_vector(2 downto 0) := "100"; --RED Light
             CONSTANT YELLOW:  std_logic_vector(2 downto 0) := "010"; --YELLOW Light
             CONSTANT GREEN: std_logic_vector(2 downto 0) := "001"; --GREEN Light

BEGIN
             --Combinational and Sequential Logic
```

```
PROCESS(CLOCK, RESET, current_state, count)
BEGIN
        IF (RESET = '1') THEN                           --Resets the whole system
                current_state <= S0;
                count <= "000";
        ELSIF (CLOCK'Event and CLOCK = '1') THEN
                IF (ENABLE = '1') THEN
                        CASE current_state IS           --Actions for each state
                                WHEN S0 =>
                                        IF (count < "110") THEN   --7 seconds delay
                                                current_state <= S0;
                                                count <= count + 1;
                                        ELSE
                                                current_state <= S1;
                                                count <= "000";
                                        END IF;

                                WHEN S1 =>
                                        IF (count < "010") THEN   --3 seconds delay
                                                current_state <= S1;
                                                count <= count + 1;
                                        ELSE
                                                current_state <= S2;
                                                count <= "000";
                                        END IF;
```

```
WHEN S2 =>
        IF (count < "001") THEN   --2 seconds delay
                current_state <= S2;
                count <= count + 1;
        ELSE
                current_state <= S3;
                count <= "000";
        END IF;

WHEN S3 =>
        IF (count < "110") THEN   --7 seconds delay
                current_state <= S3;
                count <= count + 1;
        ELSE
                current_state <= S4;
                count <= "000";
        END IF;

WHEN S4 =>
        IF (count < "010") THEN   --3 seconds delay
                current_state <= S4;
                count <= count + 1;
        ELSE
                current_state <= S5;
                count <= "000";
        END IF;
```

```
                WHEN S5 =>
                        IF (count < "001") THEN   --2 seconds delay
                                current_state <= S5;
                                count <= count + 1;
                        ELSE
                                current_state <= S0;
                                count <= "000";
                        END IF;
                WHEN OTHERS =>
                                current_state <= S0;
            END CASE;
        END IF;
    END IF;
END PROCESS;

--Description of the Output in Each State (Moore)
PROCESS(current_state)
BEGIN
    IF (current_state = S0) THEN
        Out_1 <= RED;
        Out_2 <= RED;
        Out_3 <= GREEN;
        Out_4 <= GREEN;
    ELSIF (current_state = S1) THEN
        Out_1 <= RED;
```

```
            Out_2 <= RED;
            Out_3 <= YELLOW;
            Out_4 <= YELLOW;
        ELSIF (current_state = S2) THEN
            Out_1 <= RED;
            Out_2 <= RED;
            Out_3 <= RED;
            Out_4 <= RED;
        ELSIF (current_state = S3) THEN
            Out_1 <= GREEN;
            Out_2 <= GREEN;
            Out_3 <= RED;
            Out_4 <= RED;
        ELSIF (current_state = S4) THEN
            Out_1 <= YELLOW;
            Out_2 <= YELLOW;
            Out_3 <= RED;
            Out_4 <= RED;
        ELSIF (current_state = S5) THEN
            Out_1 <= RED;
            Out_2 <= RED;
            Out_3 <= RED;
            Out_4 <= RED;
        ELSE
            Out_1 <= RED;
            Out_2 <= RED;
```

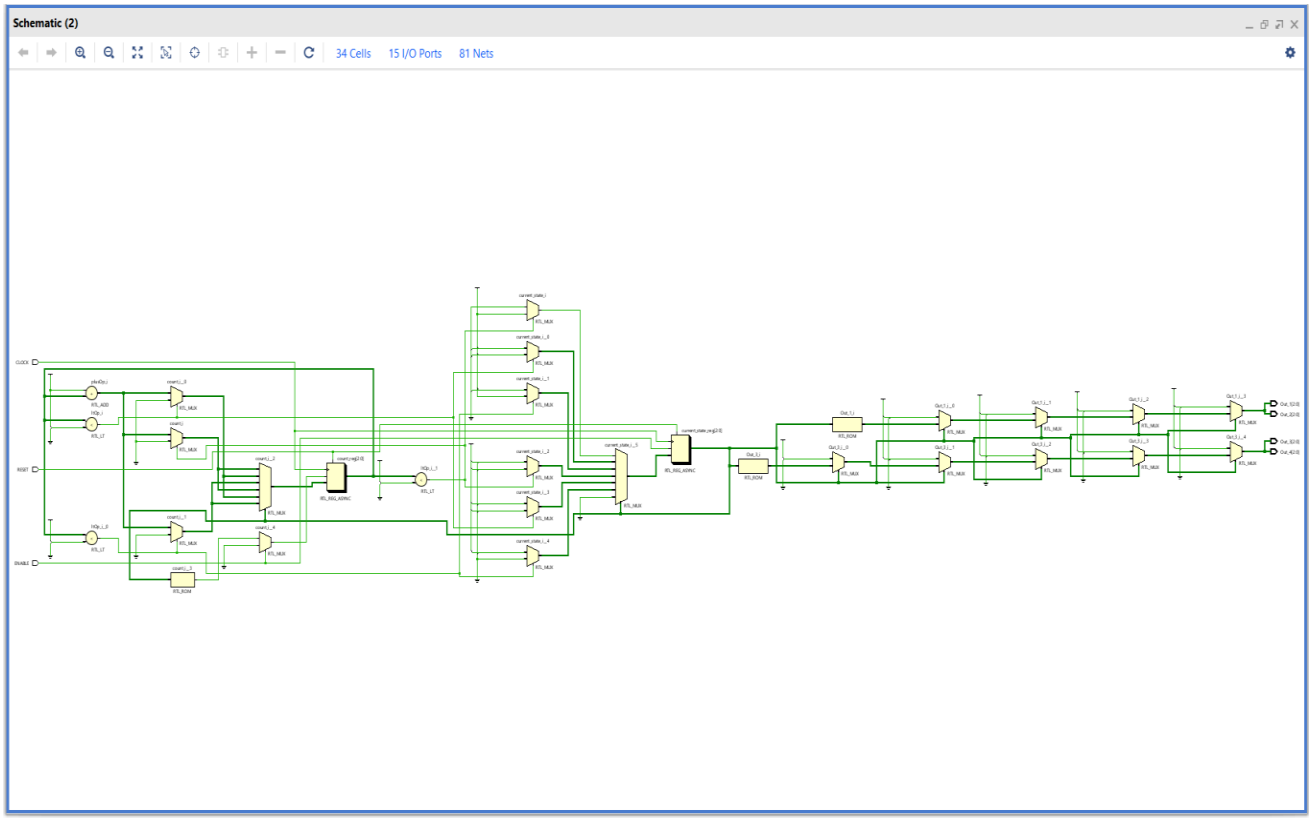
```
                Out_3 <= GREEN;
                Out_4 <= GREEN;
            END IF;
        END PROCESS;
END Controller_Arch;
```

5. Constraint File:

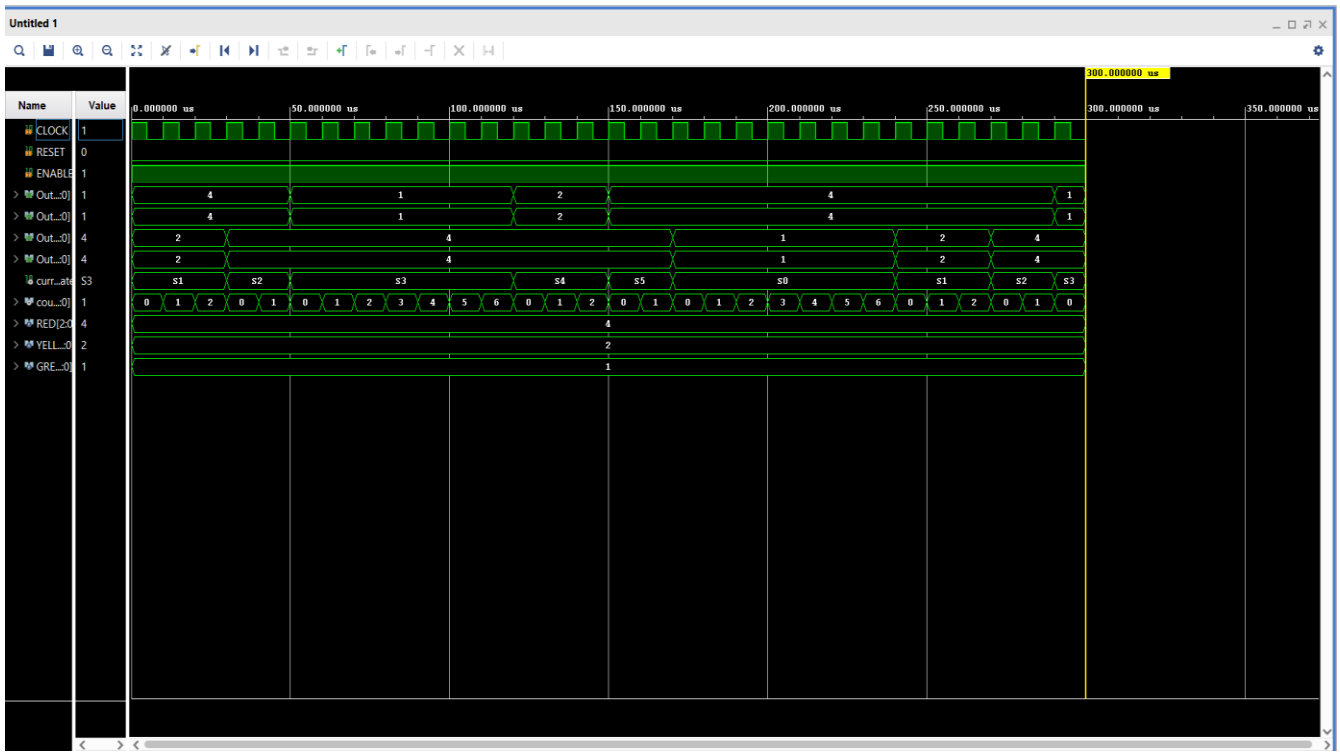
```
## Clock Signal
set_property PACKAGE_PIN E3 [get_ports {CLOCK}]
set_property IOSTANDARD LVCMOS33 [get_ports {CLOCK}]

## Reset Signal (assigned to SW0)
set_property PACKAGE_PIN J15 [get_ports {RESET}]
set_property IOSTANDARD LVCMOS33 [get_ports {RESET}]

## Enable Signal (assigned to SW1)
set_property PACKAGE_PIN L16 [get_ports {ENABLE}]
set_property IOSTANDARD LVCMOS33 [get_ports {ENABLE}]

## Output Signals Mapped to LEDs
# Out_1 (3-bit output) mapped to LD0, LD1, LD2
set_property PACKAGE_PIN H17 [get_ports {Out_1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_1[0]}]
set_property PACKAGE_PIN K15 [get_ports {Out_1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_1[1]}]
```

```
set_property PACKAGE_PIN J13 [get_ports {Out_1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_1[2]}]

# Out_2 (3-bit output) mapped to LD3, LD4, LD5
set_property PACKAGE_PIN N14 [get_ports {Out_2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_2[0]}]
set_property PACKAGE_PIN R18 [get_ports {Out_2[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_2[1]}]
set_property PACKAGE_PIN V17 [get_ports {Out_2[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_2[2]}]

# Out_3 (3-bit output) mapped to LD6, LD7, LD8
set_property PACKAGE_PIN U17 [get_ports {Out_3[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_3[0]}]
set_property PACKAGE_PIN U16 [get_ports {Out_3[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_3[1]}]
set_property PACKAGE_PIN V16 [get_ports {Out_3[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_3[2]}]

# Out_4 (3-bit output) mapped to LD9, LD10, LD11
set_property PACKAGE_PIN T15 [get_ports {Out_4[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_4[0]}]
set_property PACKAGE_PIN U14 [get_ports {Out_4[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_4[1]}]
set_property PACKAGE_PIN T16 [get_ports {Out_4[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Out_4[2]}]
```

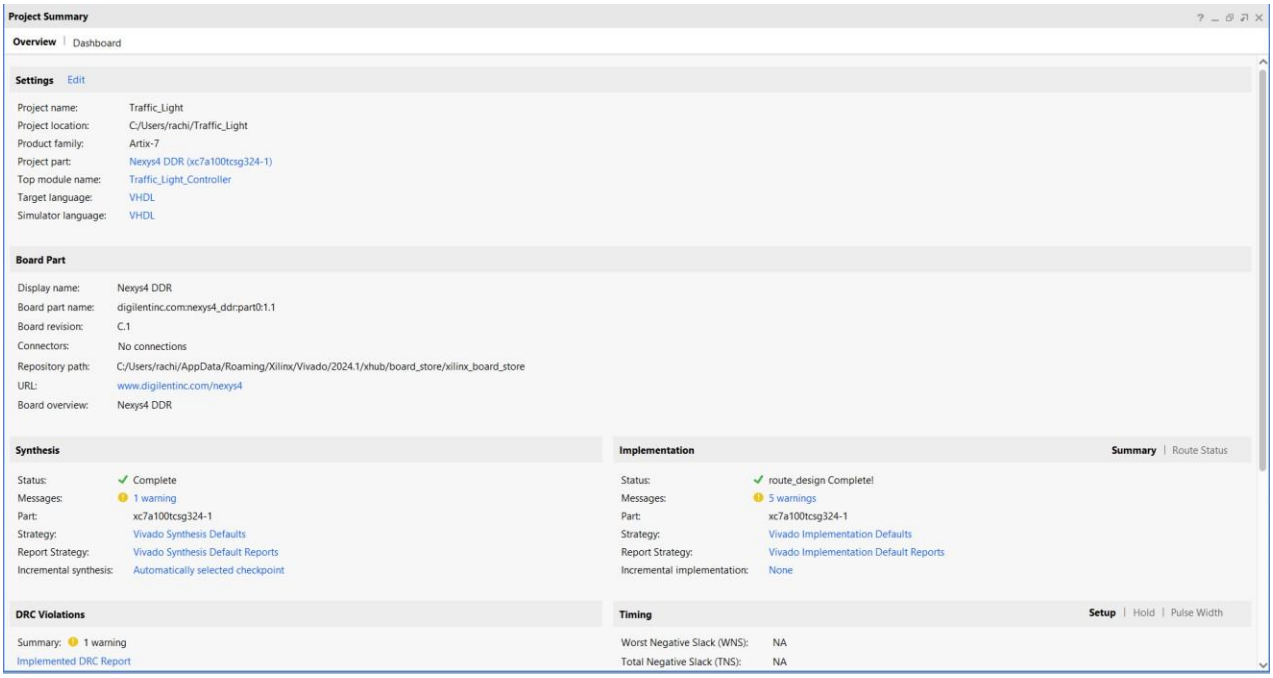
6. RTL Schematic

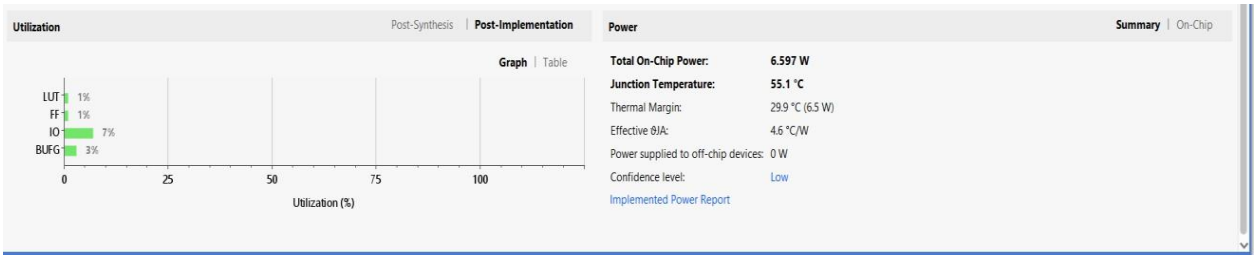


7. Simulation Timing Diagram



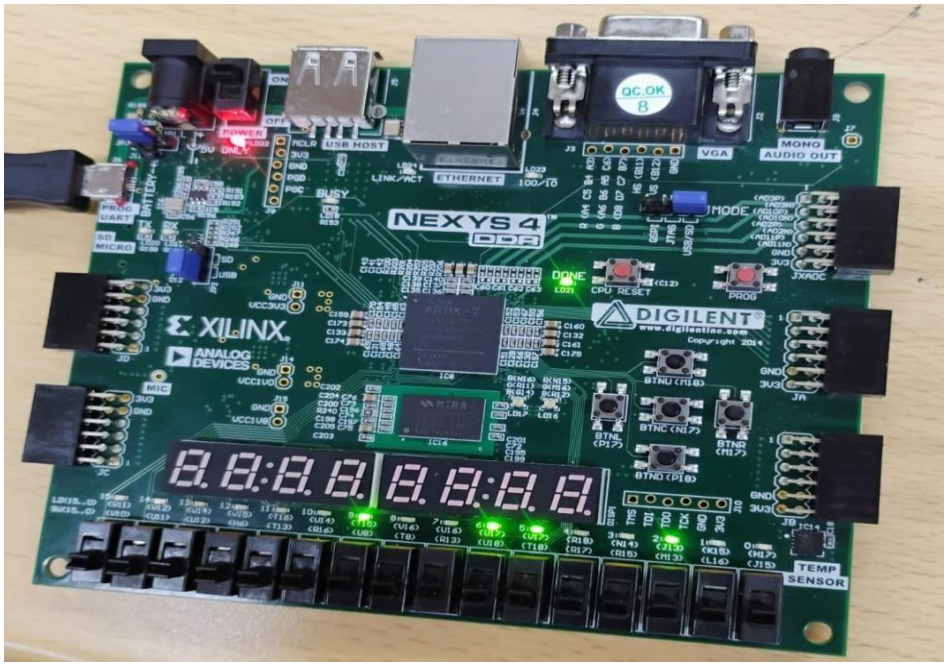
8. Project Summary



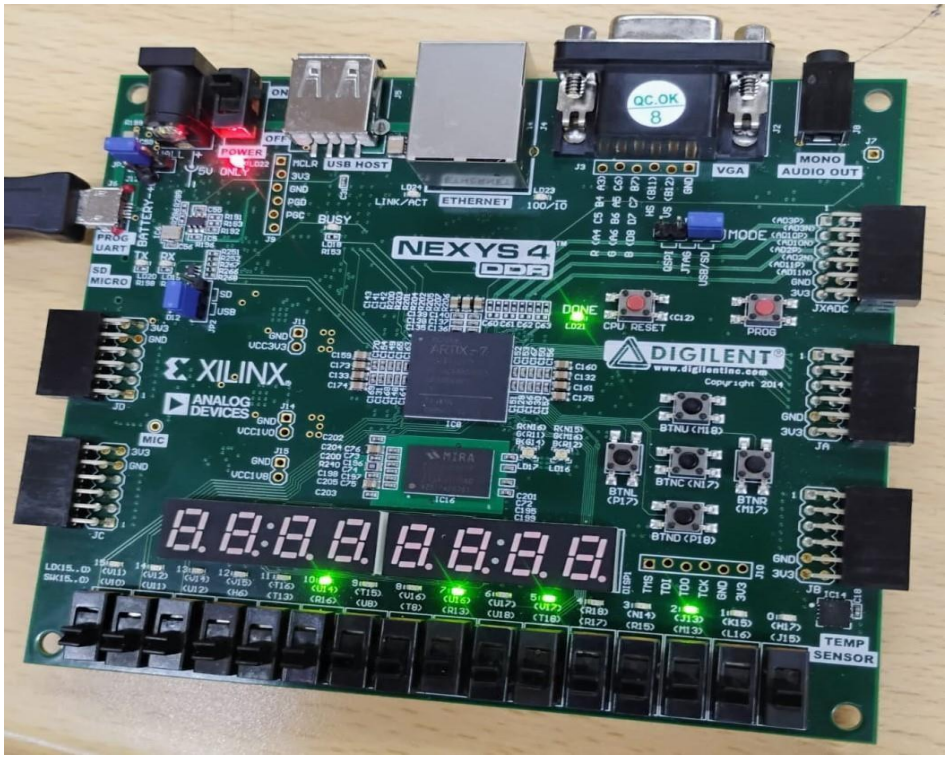


9. FPGA Board Implementation Results Snapshot

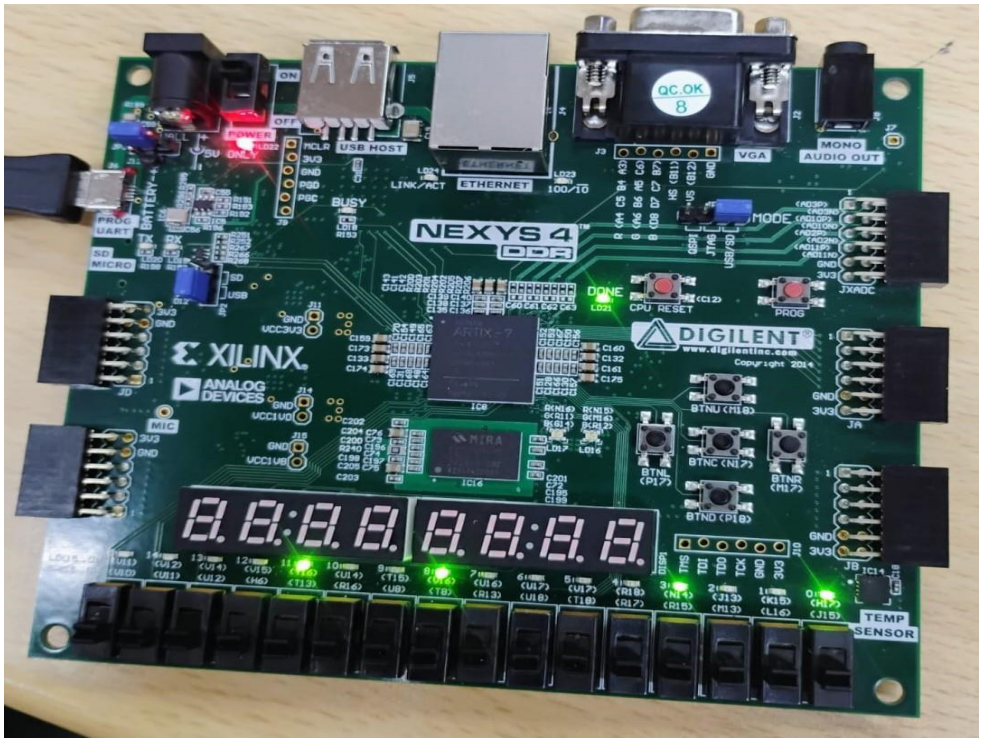
9.1. I/P: Clock, Reset = 0, Enable = 1; O/P: State S0: Red, Red, Green, Green



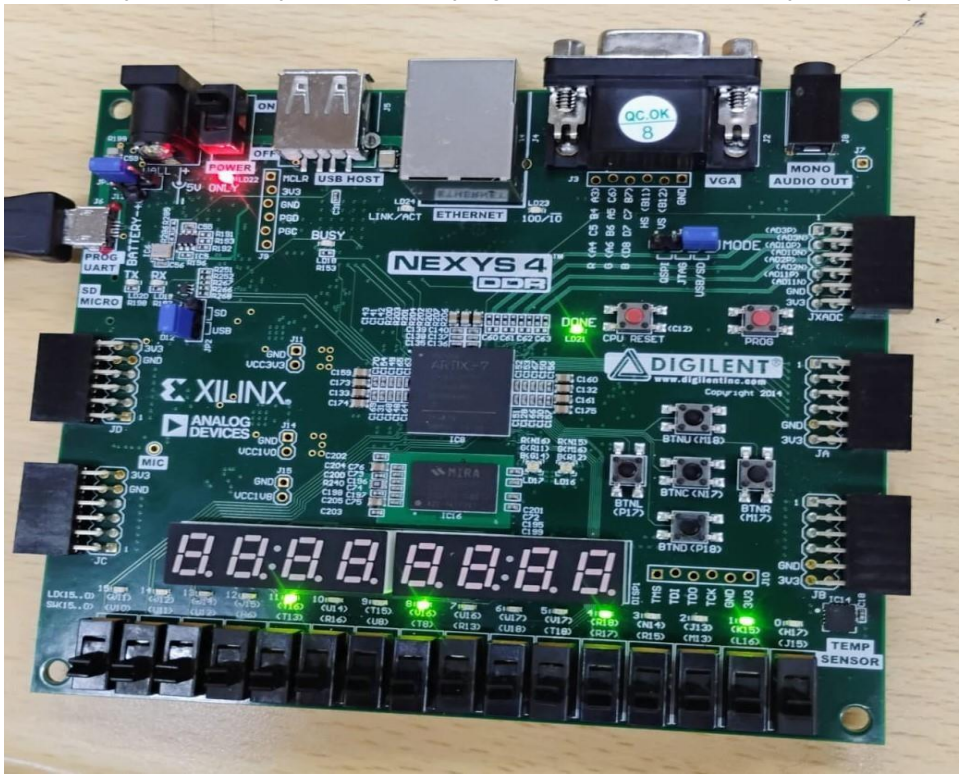
9.2. I/P: Clock, Reset = 0, Enable = 1; O/P: State S1: Red, Red, Yellow, Yellow



9.3. I/P: Clock, Reset = 0, Enable = 1; O/P: State S3: Green, Green, Red, Red



9.4. I/P: Clock, Reset = 0, Enable = 1; O/P: State S4: Yellow, Yellow, Red, Red



10. Conclusion

The objective of this project was to design a Traffic Light Controller for use in road intersections, to implement the design by using the IEEE Standard VHDLLanguage, to simulate the system using the software, to implement the hardware component on FPGA Board (Nexys 4 DDR) and to be able to understand and explain the output results. The report clearly shows that the project fulfilled the objectives, simulation, and the implementation on FPGA Board (Nexys 4 DDR) in order to verify with the real-world application of this controller.

Name of Entity	No. of LUT used	Total On Chip Power
Traffic_Light_Controller	1	6.597W

Table 2. Summary of Total On Chip Power Consumption