EE 705
VLSI Design Lab
Course Project
Traffic Light Controller System

Anuranan Das 18D070037
Parth Makode 18D070056
Pranav Patel 18D070057

Contents

1 Introduction and Objectives **3**

2 Theory – Algorithm **3**

3 Implementation - Components used **5**

4 Software used **6**

5 Application Process **6**

6 Results **6**

6.1 Quartus and iVerilog Simulation 6

6.2 Labsland Implementation 7

7 Summary **10**

8 Code Base **10**

9 References **10**

1 Introduction and Objectives

Traffic signals generally operate with a fixed time for red and green lights. To manage traffic better, the fixed time value can be controlled and changed according to traffic conditions. The aim of the project is to implement a Traffic Light Controller (TLC) which will operate according to the traffic load. It is simulated in quartus and also tested on FPGA at Labsland. The objectives for the project can be broadly as follows :

- Implement a design of a modern FPGA-based Traffic Light Control (TLC) System to manage the road traffic.
- Intelligent peak timing method based on sensors, more efficient than usual fixed time method.
- Hierarchical (Module based) design of the system in verilog and conversion for running in FPGA.

2 Theory – Algorithm

Traffic light controller (TLC) is used to lessen or eliminate conflicts at area shared among multiple traffic streams called intersections; by controlling the access to the intersections and apportioning effective period of time between various users. Goal of this project is to manage the traffic movement of four intersecting roads and to achieve optimum use of the traffic.

Road structure of the traffic intersection is shown is fig. 1. In this structure, there are six traffics, represented by T1, T2, T3, T4, T5 and T6 to be controlled. T1 and T2 have been identified as the main road for the first junction while T4 and T6 are for the second junction. The last two traffic lights, T3 and T5 are the smaller roads. The traffic flows are symbolized by the arrows in the fig. 1.

The image of the square which was used for this implementation is shown below.

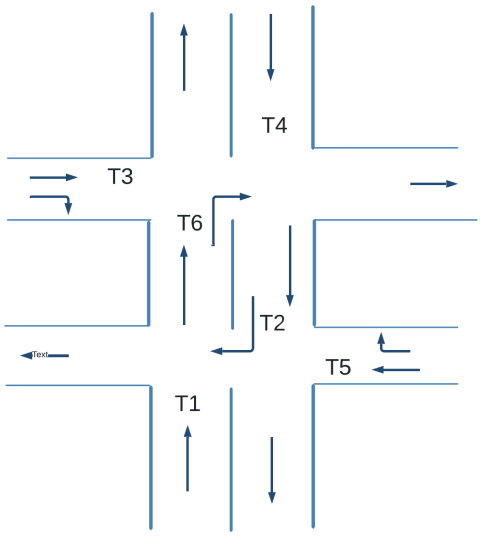

Figure 1: Road system considered in this project

The timing settings for the TLC are as follows:-

- Peak Hours
  - T1 and T6 remain green for 32s
  - T2 and T4 remain green for 32s

  - T3 and T5 remain green for 16s

- Non - Peak hours
  - T1 and T6 remain green for 16s
  - T2 and T4 remain green for 16s
  - T3 and T5 depend on sensor values. Refer to fig. 2 .

Some other important conventions followed are noted below,

- Signals are in active low format, 00 = Green , 01 = Yellow and 10 = Red.
- Reset is active low and resets the system to start for TL1 - TL6 cycle.
- The vertical roads are main roads which are always busy.

The Flowchart below explains the state flow of the TLC system.

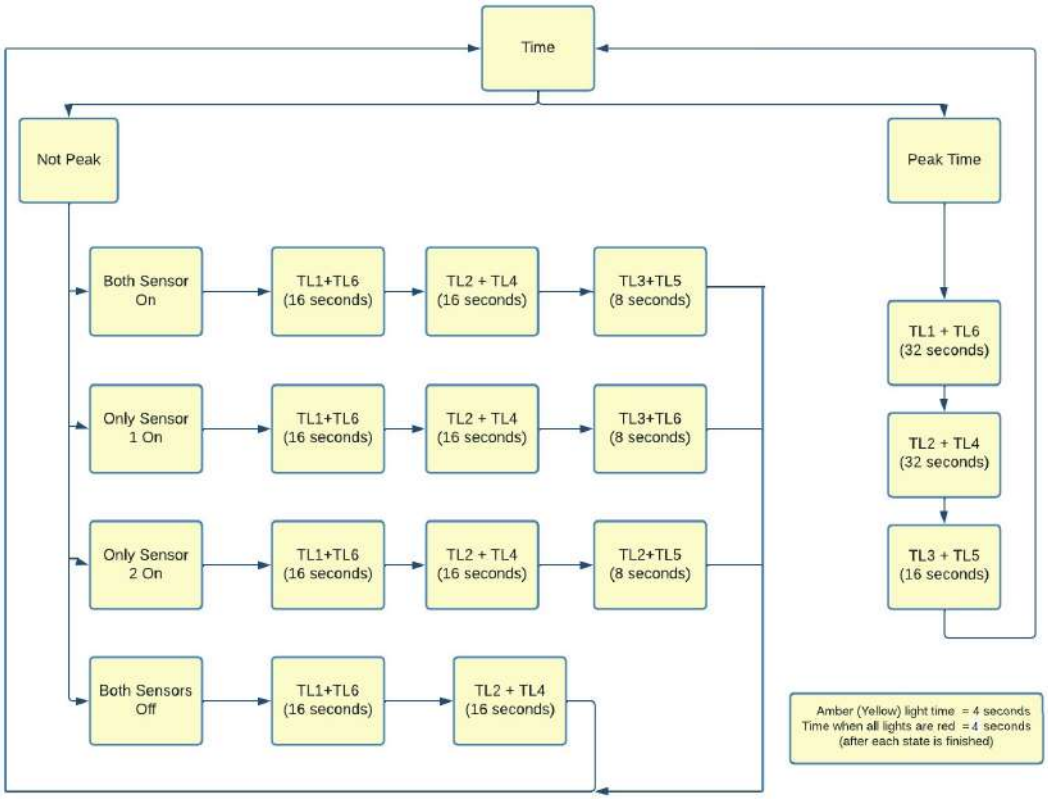

Figure 2: State flow for the TLC system. To note here, T5 and T3 cycle depends on the sensor values during off-peak hour.

Horizontal roads, i.e. T3 and T5 crossings have sensors, which observe the load and turn on/off appropriately for control. Some peak hour values are preset, and the sensors can always turn on in case of unexpected load. The time is kept in check with a 12 hour clock with a flag to denote *am or pm*.

3 Implementation - Components used

The code is implemented in the form of four modules:

- Clock module – A twelve hour BCD clock with am and pm indicator.
- TLC main module – The state macheine of fig. 2 is implemented here.
- Peak - Off-peak module – Determines the timings of the day to be considered as peak traffic time or not.
- Top module to integrate all the submodules.

The input signals to the overall design is as follows :

- *clk* – System clock, set according to time granularity required.
- *sensor1* and *sensor2* – Sensor values need to be fed externally through hardware interfacing.
- *ena* – Enable for clock.
- *reset* – Restart the entire system while set.

The output signals are :

- *TL1 to TL6* – Traffic light outputs, each two bit values with encoding as mentioned earlier.

A detailed interconnection of modules is illustrated in fig. 3 .

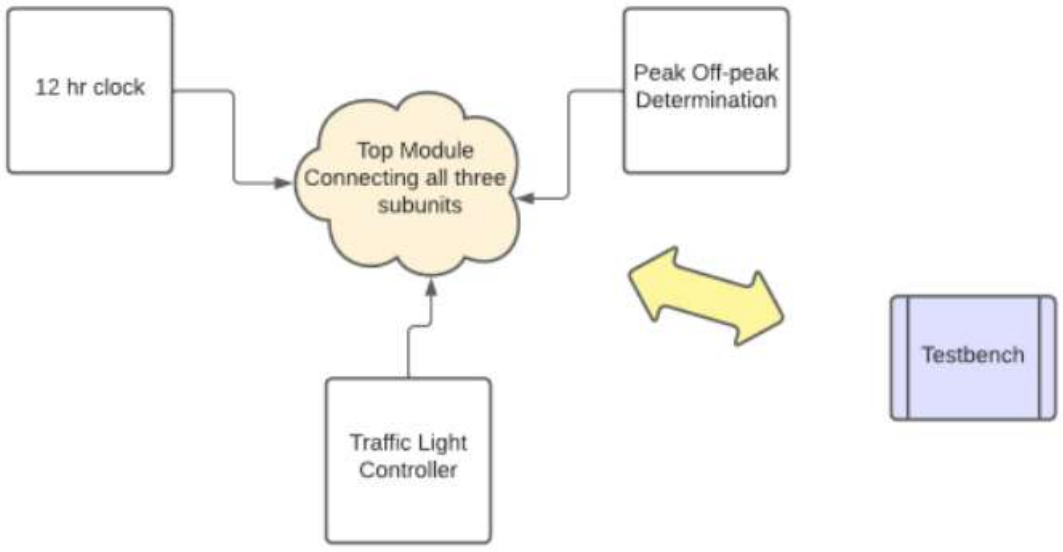

Figure 3: Module Hierarchy Planning

It is important to note here that, an equal importance was attached in formulating the testbench to verify the working of design correctly. Manually picked test-cases were used however, designed by the team members.

4 Software used

The following software and tools were used in carrying out the project successfully.

- Simulation – Quartus, Modelsim and iverilog.
- FPGA implementation – Labsland, DE2-115.
- File versioning – Github.

5 Application Process

The project ideation and application was managed in the following manner.

1. Initially Learnt the concept of dynamic time approach from ref. [1].
2. Decided we are going to simulate in verilog.
3. Wrote the independent submodules.
4. Parallely we wrote testcases.
5. Then Topmodule was created joining all submodules and linked to testbench.
6. Once simulations were perfected, we moved on to modify code for labsland.
7. Same testcases were verified with labsland.
8. Documented results and demonstrated in video.

6 Results

6.1 Quartus and iVerilog Simulation

Quartus was used to get the RTL netlist, as shown in fig. 4. The simulation was run in iverilog (can be also done on Modelsim). For this, a simple testbench wass written. The clock and sensor values were initialized as per the requirement to verify the working of the system based on testcases designed.

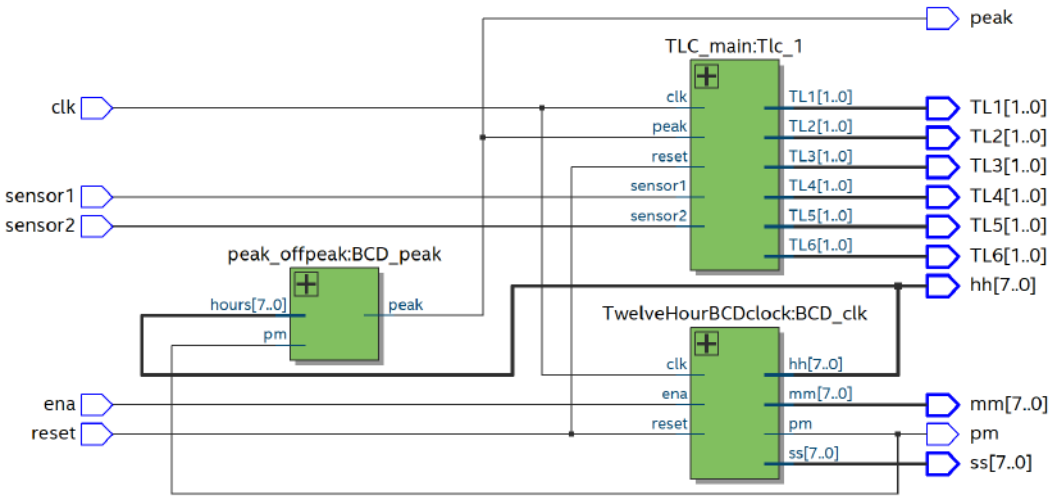

Figure 4: RTL View of the Controller, extracted from Quartus

The peak hours were considered to be 0700 - 0900 , 1200 - 1400 and 1700 - 1900. These values can be changed anytime, as per the requirement of the traffic intersection. The simulation results from iverilog are shown below.

To see the video and screenshots for this part, click on the link below.

Demo and screenshots link : **Link**

One of the cases is detailed in the figure below. The time is 3 pm, and both sensors are off.

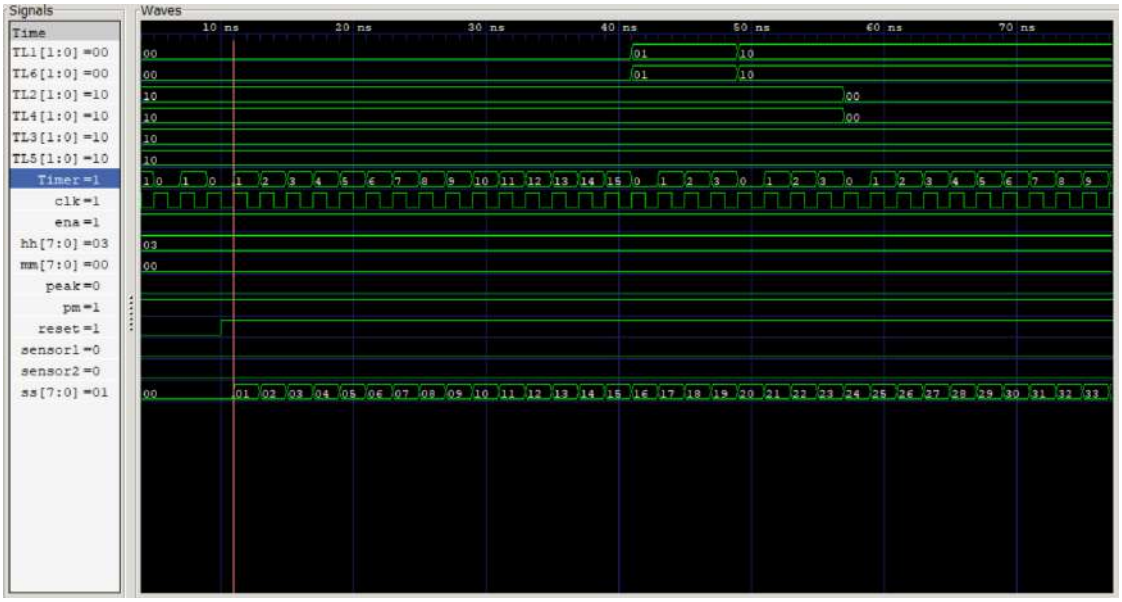

Figure 5: TLC output for non-peak hour

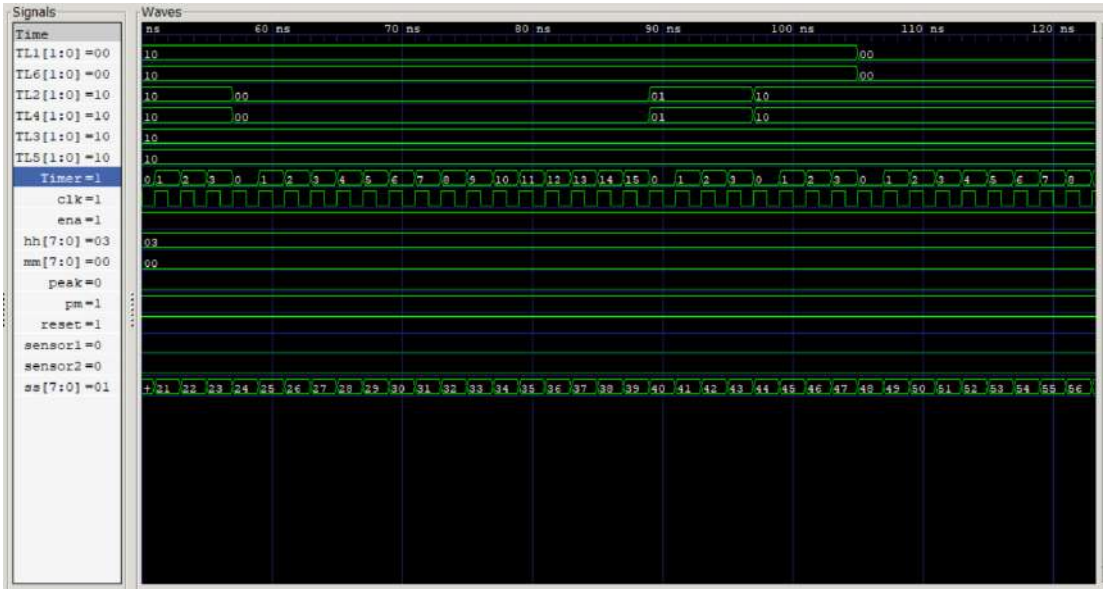

Figure 6: TLC output for non-peak hour

6.2 Labsland Implementation

The TLC code was modified to work in Labsland and verified for the same testcases. The system was designed using the following mapping in labsland (as depicted in fig. **??**). The clock input was slowed down to give the system changes enough time to be visible to bare eyes.

Signal name	Signal type	DE2-115 mapping	Brief Role
clk	Input	CLOCK_50(slowed version)	System clock
Reset	Input	SW[0]	For Reset
Ena	Input	SW[1]	For Count
Sensor1	Input	SW[2]	Sensor Indication
Sensor2	Input	SW[3]	Sensor Indication
Peak	Input	SW[4]	Peak hour indication
Peak1	Input	SW[5]	Peak hour indication
LEDR-TL6	Output	LED[17:15]	TL6
LEDR-TL5	Output	LED[14:12]	TL5
LEDR-TL4	Output	LED[11:9]	TL4
LEDR-TL3	Output	LED[8:6]	TL3
LEDR-TL2	Output	LED[5:3]	TL2
LEDR-TL1	Output	LED[2:0]	TL1
an	Output	LEDG[0]	am/pm
SW[4]	Output	LEDG[1]	Peak hour indication
SW[5]	Output	LEDG[2]	Peak hour indication
HEX0-HEX7	Output	HEX[0]-HEX[7]	For Display

Table 1: Signal mapping table for LabsLand

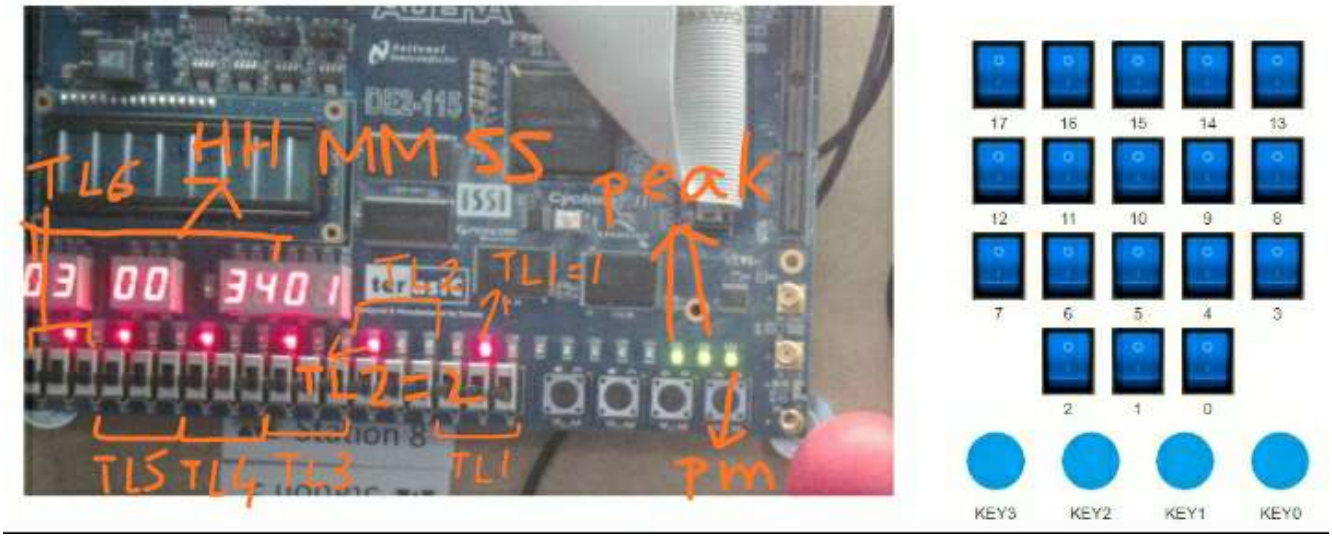

Figure 7: System during peak hours

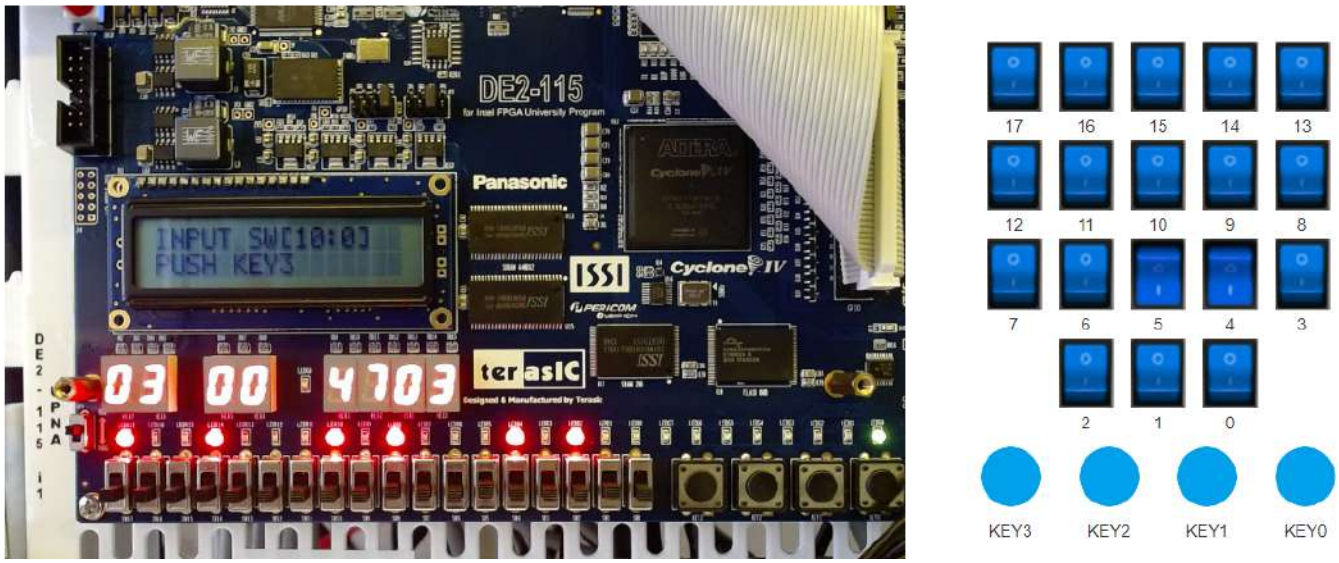

Figure 8: System during off-peak hours and when both sensors are deactivated

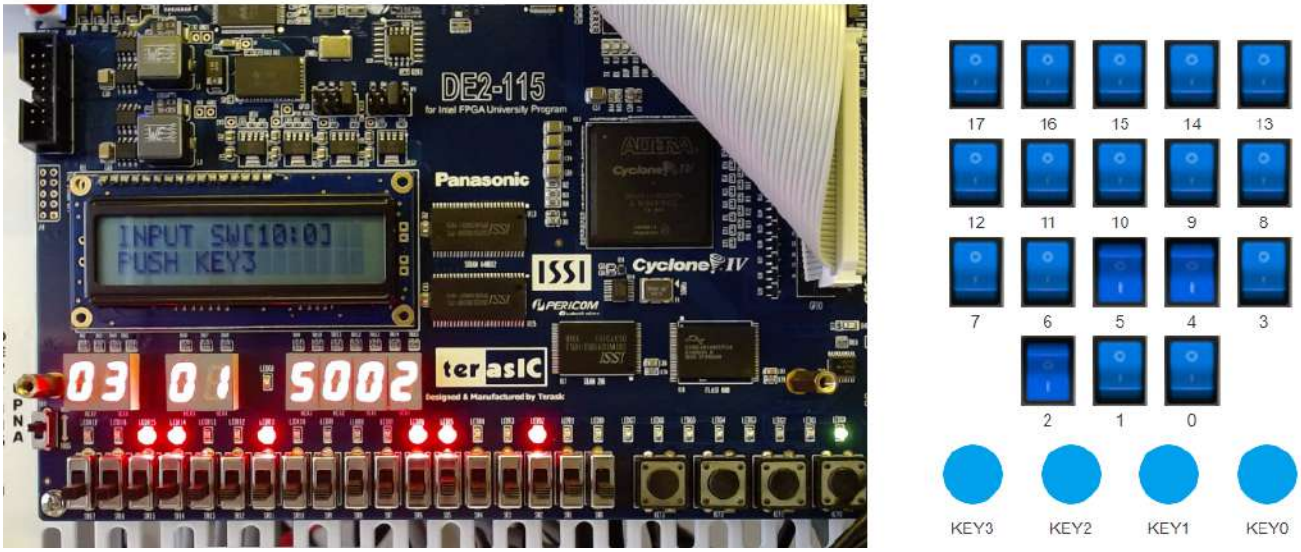

Figure 9: System during off-peak hours and sensor 1 is activated and sensor 2 is off

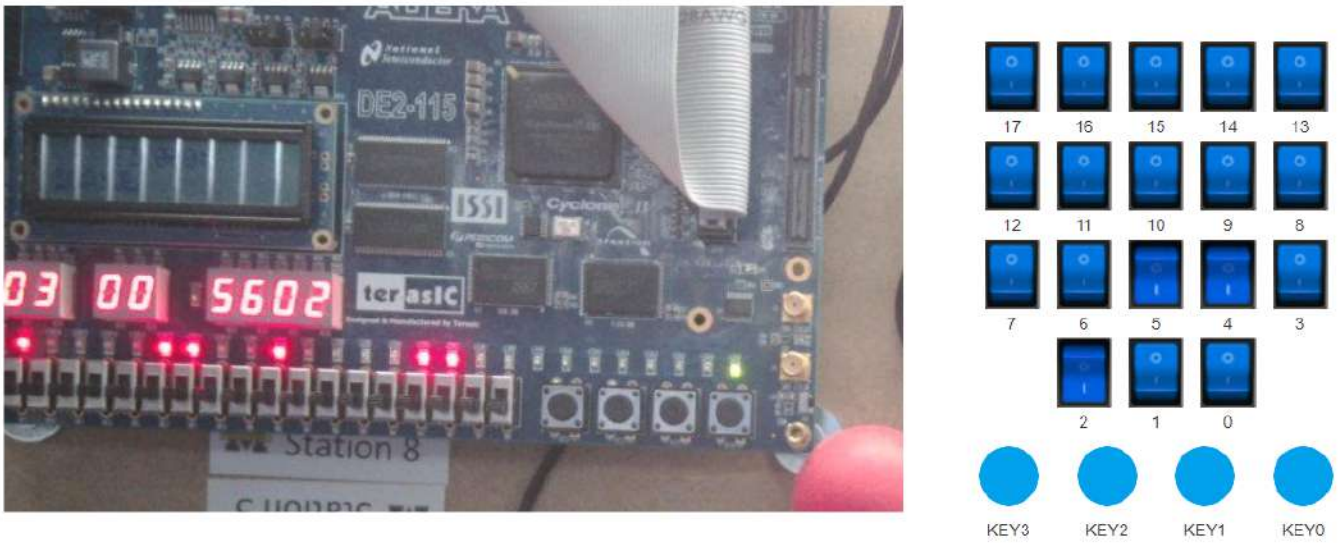

Figure 10: System during off-peak hours and sensor 2 is activated and sensor 1 is off

Video Link : **Video**

We have taken into notice the glitches in latency in Labsland.

7 Summary

As discussed in the above section, we were able to successfully implement the design in verilog both in terms of simulation and in terms of labsland FPGA respectively. 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.

The work could be extended with minor modifications for pedestrian control as well. Finally, it can be written into an embedded chip to carry out the traffic control in a city.

8 Code Base

Link to github repository: `https://github.com/AnDa-creator/TLC_ee705Verilog`

9 References

[1] M.F.M. Sabri, M.H. Husin, W.A.W.Z. Abidin, K.M. Tay, and H.M. Basri. Design of fpga-based traffic light controller system. In *2011 IEEE International Conference on Computer Science and Automation Engineering*, volume 4, pages 114–118, 2011.