4 - Way Traffic Lights System Implementation Using Nexys-2 FPGA Board

Final Report

25TH August, 2020

By: Hizbullah Khan

Roll Number: [FA19-BCE-080]

Supervisor: Ma'am Asma Ramay

Table of Contents

Introduction

The problem given was to make a 4-Way traffic light system meant for a 4-Way intersection. This was successfully tackled. All the given requirements were met. The project was implemented in Verilog for Nexys-2 FPGA Dev Kit.

There are four sets of traffic lights, one per each road. Each of these sets have three lights each; namely Red, Yellow and Green. These three lights turn ON/OFF based on fixed intervals, unlike the real-world solution in which sensors are used to measure vehicle traffic on the road networks which then provide information for toggling the traffic lights respectively. All of these lights in this project are represented by LEDs of respective colors. In Verilog code, these LEDs or traffic lights are modelled using four *arrays* for four sets of traffic lights. A Finite-State-Machines is then used to cycle through all of the traffic lights and loop back. A clock divider module programmed in Verilog is also used to reduce the internal clock on the Nexys-2 FPGA from *50MHz* to *1Hz*. This provides accurate unit second timings for toggling the traffic lights just like in real world.

Functional Requirements

All of the given functional requirements were met successfully as below:

1. Fully functional Verilog code synthesizable on the Nexys-2 FPGA.
2. Proper timing between the traffic lights as set in the Verilog code.
3. All the hardware components such as LEDs and resistors can easily be managed on a single breadboard neatly.
4. No extra power required for turning on the LEDs. Nexys-2 board's I/O header pins provide adequate voltage (3.3V) for powering all 12 the traffic lights.
5. Sequence of traffic lights that was followed is:
    i. Green
    ii. Yellow
    iii. Red

Optional Feature

One optional feature which was implemented is *Pedestrian Lights*. These pedestrian lights turn ON/OFF automatically based on which road's traffic lights are cycled through at that time. The order is as follows:

- Road 1 >> ON, only Pedestrian Lights for Road 2 >> ON.
- Road 2 >> ON, only Pedestrian Lights for Road 3 >> ON.
- Road 3 >> ON, only Pedestrian Lights for Road 4 >> ON.
- Road 4 >> ON, only Pedestrian Lights for Road 1 >> ON.

After the Road 4's turn, the entire process is repeated again during which the pedestrian lights also toggle ON/OFF according to which Road's traffic light is Green.

Parts of Verilog Code

The Verilog code used followed behavioral modelling. Main parts of the Verilog code are summarized as below:

- Start of Verilog module.
- Declaration of *clk* which takes inputs from the internal *50MHz* clock.
- Declaration of *secclk*, which stands for "seconds clock", representing the divided clock.
- Declaration of 4 sets of traffic lights as 4 *arrays*.
- Declaration of 4 pedestrian lights.
- Declaration of state variable for main FSM.
- Declaration of 4 state variables for sub FSMs.
- Declaration *counter* and *divisor* (counter max limit) variables.
- Instantiating the *clk_div* module for dividing the main clock.
- Start of *always* block.
- Start of *case* statement for main FSM. (*big_state*)
- Five states of main FSM are
    1. Initializing
    2. Sub FSM-1 for Road 1 traffic lights. (*state_1*)
    3. Sub FSM-2 for Road 1 traffic lights. (*state_2*)
    4. Sub FSM-3 for Road 1 traffic lights. (*state_3*)
    5. Sub FSM-4 for Road 1 traffic lights. (*state_4*)
- End of *case* statement for main FSM.
- End of *always* block
- End of Verilog module.

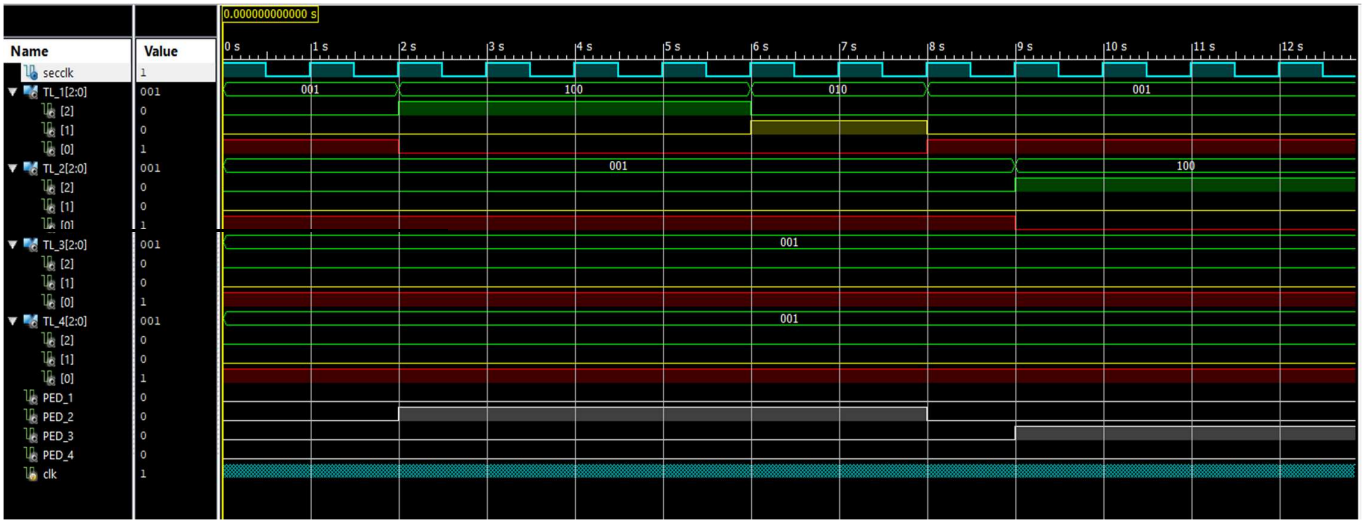
The sub FSM for traffic lights of each road is as follows:

1. First State >> Red light switched ON. others OFF
2. Second State >> Green light switched ON. others OFF
3. Third State >> Yellow light switched ON, others OFF

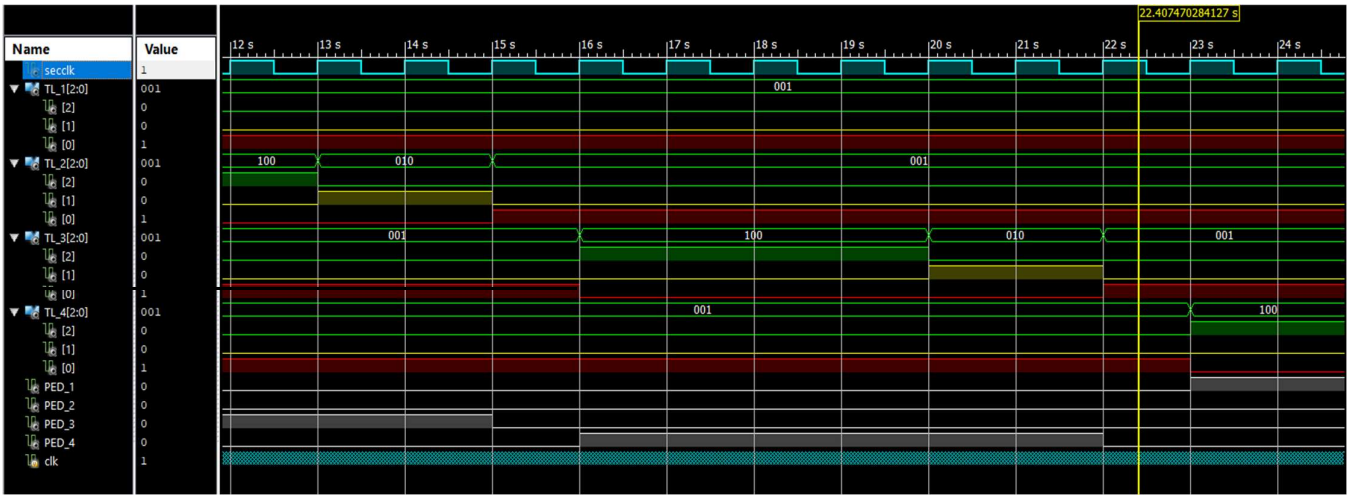
The actual delay between the traffic lights of each road was achieved by using counters. By giving the counter a limit to count up-to, and checking that via a conditional statement, the counter will count until it has reached that limit and while doing so, the state machine will loop through the same state and each time it loops, it will provide a delay of unit second. Thus counting 4 times will give us 4 seconds and so on. The clock divider module which converts the 50MHz clock to 1Hz is also based on the same principle.

Verilog Simulation Result

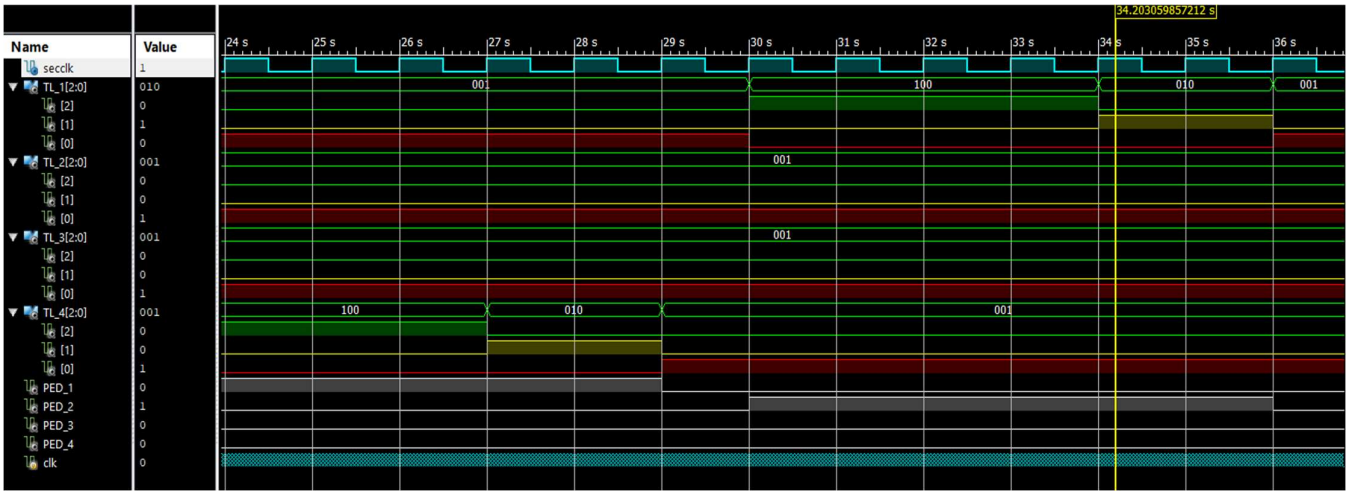
(From 0 to 12 seconds)



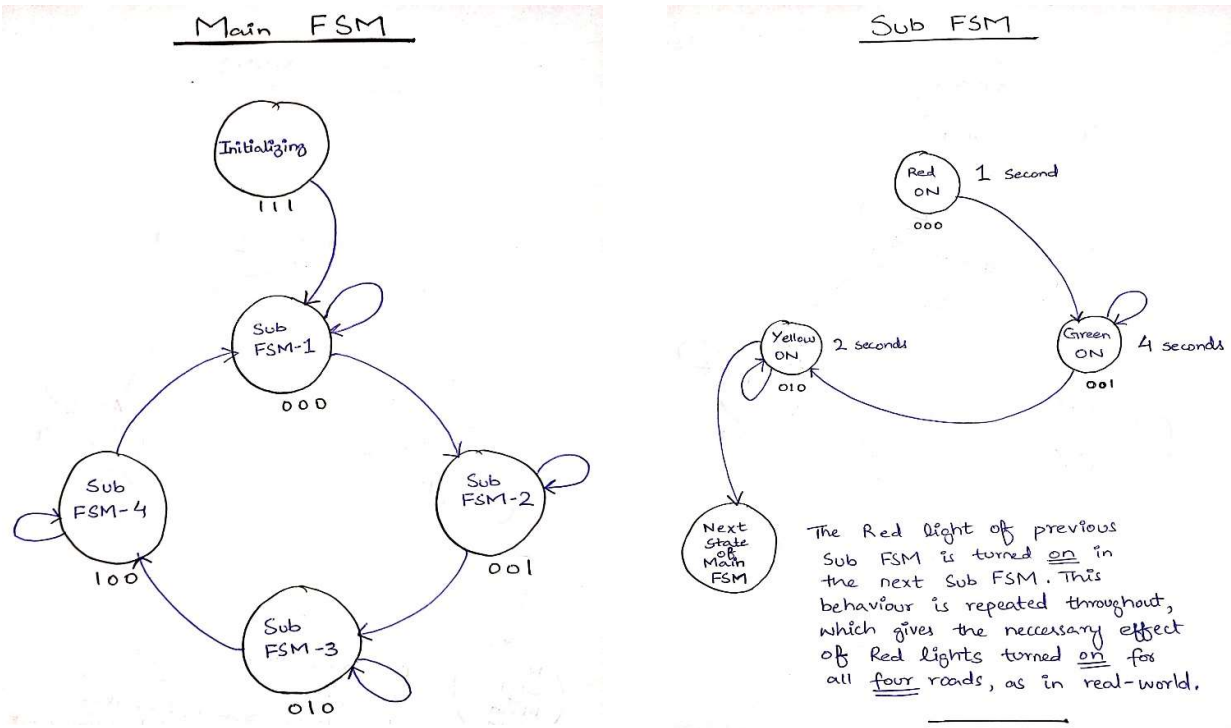
(From 12 to 24 seconds)



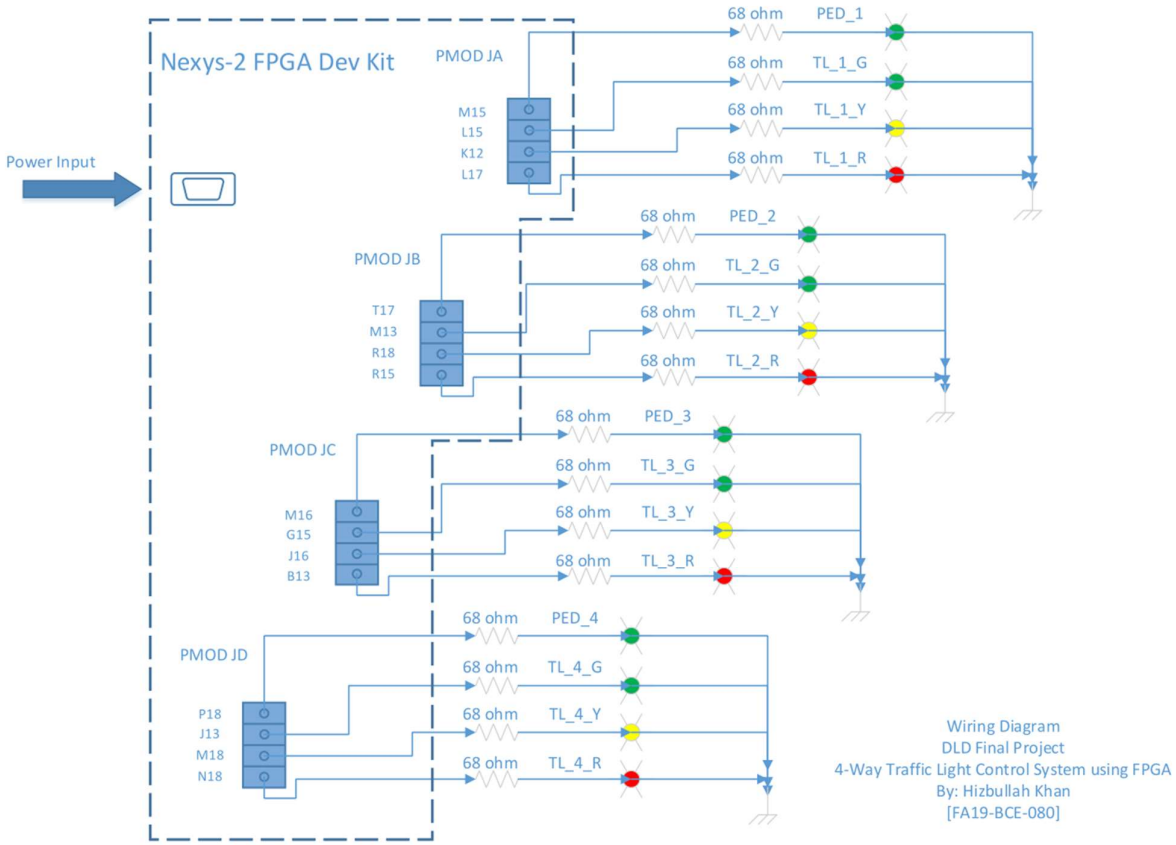
(From 24 to 36 seconds)



State Diagram



Circuit Diagram



The mapping of traffic light LEDs is shown in the circuit diagram. The *Pmod* connecters on the Nexy-2 FPGA were used for this purpose. The respective pins which connecting to each LED of each traffic light set is are also mentioned. For protection, the LED are connected via resistors to limit the excess current. The output voltage received from each pin of *Pmod* connector is 3.3 Volts. The LEDs turn ON or OFF when a value of one or zero is assigned to a *Pmod* pin, respectively.

Further Improvement

Another extra feature which can be implemented is to add a 7-Segment display to the 4-Way traffic light system. This 7-Segment display would show the remaining counter time between each transition of traffic lights and inform the drivers when the given light will turn Yellow or Green, etc. This feature can be easily added in the future which will further improve the overall design and quality of the traffic light system.

Conclusion

This project truly tested my capabilities when it comes to DLD base knowledge, Verilog know-how, and project management skills. Many new concepts were introduced to me on the way as a result of working on this project. I learnt about proper pin mapping and different types of connectors that can be used on this specific Nexys-2 FPGA. I also learnt about very crucial Verilog concepts of which the most important one was the fact that hardware programming approach is very different than regular software programming approaches.

Overall, this project came out to be quite successful as the intended goals were perfectly achieved while also successfully implementing an extra feature.