

**Faculty of Engineering & Technology – Electrical & Computer Engineering Department**

**First Semester 2022 – 2023**

**Advanced Digital – ENCS3310**

***Course Project***

**Name: Amal Ziad**

**ID: 1192141**

**Section: 2**

**Instructor: Dr. AbdAllatif AbuIssa**

**Date: 26th Jan 2023**

Contents

[Table of figures 2](#_Toc125673455)

[1. Introduction and Background 3](#_Toc125673456)

[2. Design Philosophy 4](#_Toc125673457)

[ Basic components 4](#_Toc125673458)

[ Structure code 5](#_Toc125673459)

[ Verification and testing 6](#_Toc125673460)

[3. Simulation & Results 7](#_Toc125673461)

[ Test output 7](#_Toc125673462)

[ System verification no-faulty 7](#_Toc125673463)

[ System verification with faulty 8](#_Toc125673464)

[4. Conclusion and future works 9](#_Toc125673465)

# Table of figures

[Figure ‎1.1 traffic light system circuit 3](#_Toc125673466)

[Figure ‎2.1 FSM state diagram 5](#_Toc125673467)

[Figure ‎2.2 n-bit counter structure 5](#_Toc125673468)

[Figure ‎2.3 Memory module 6](#_Toc125673469)

[Figure ‎2.4 tb\_traffic module 7](#_Toc125673470)

[Figure ‎3.1 Results of testbench 1 7](#_Toc125673471)

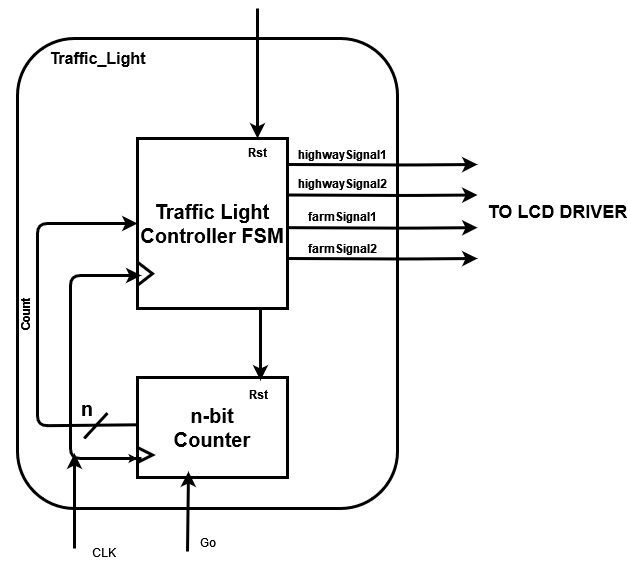
[Figure ‎3.2 Results of testbench 2 7](#_Toc125673472)

[Figure ‎3.3 Console results no faulty 8](#_Toc125673473)

[Figure ‎3.4 Console results faulty 8](#_Toc125673474)

# Introduction and Background

For this project, a solution to the real traffic light for crossroad of two streets is designed and simulated, and later verified for functionality. The real traffic light circuit is basically a circuit of finite state machine (FSM). A finite state machine is an abstract machine that can be in exactly one of a finite number of states at any given time. The traffic light system has 4 signals for two roads: 2 signals for farm way and 2 signals for highway. In this project, the traffic light controller has the design shown in fig.1.1.

**

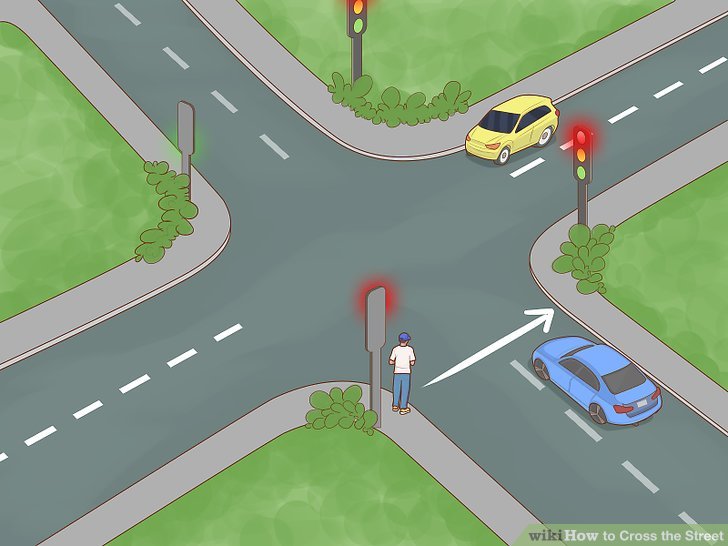


Figure . traffic light system circuit

The unit receives 3 inputs (clock, go, and reset) of size 1 bit for each. The n-bit counter is used to count which state to go next from the present state. The traffic light FSM is used to generate the signals of each road light and after receiving the state number. The reset input returns the machine to the default state, which is S0. The clock used for synchronization. Each signal of the outputs is 2-bit size and the output can be one of the following: 2’b00 (green), 2’b01 (yellow), 2’b10 (red), 2’b11 (red and yellow). All states are shown in the table 1-1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| State | Highway TL1 | Highway TL2 | Farm TL1 | Farm TL2 | Delay [Sec] |
| S0 | Red | Red | Red | Red | 1 |
| S1 | Red-Yellow | Red-Yellow | Red | Red | 2 |
| S2 | Green | Green | Red | Red | 30 |
| S3 | Green | Yellow | Red | Red | 2 |
| S4 | Green | Red | Red | Red | 10 |
| S5 | Yellow | Red | Red | Red | 2 |
| S6 | Red | Red | Red | Red | 1 |
| S7 | Red | Red | Red-Yellow | Red-Yellow | 2 |
| S8 | Red | Red | Green | Green | 15 |
| S9 | Red | Red | Green | Yellow | 2 |
| S10 | Red | Red | Green | Red | 5 |
| S11 | Red | Red | Yellow | Red-Yellow | 2 |
| S12 | Red | Red | Red | Green | 10 |
| S13 | Red | Red | Red | Yellow | 2 |
| S14 | Red | Red | Red | Red | 1 |
| S15 | Red | Red-Yellow | Red | Red | 2 |
| S16 | Red | Green | Red | Red | 15 |
| S17 | Red | Yellow | Red | Red | 3 |

Table ‑ all states of the traffic light

# Design Philosophy

## Basic components

* 1. Finite state machine

A finite state machine (FSM) Footnote1 consists of a set of states si and a set of transitions between pairs of states si, sj. A transition is labeled condition/action: a condition that causes the transition to be taken and an action that is performed when the transition is taken. As shown in fig.2.1. A state is denoted by a circle labeled with the name of the state. The incoming arrow denotes the initial state. A transition is shown as an arrow from the source state to the target state.

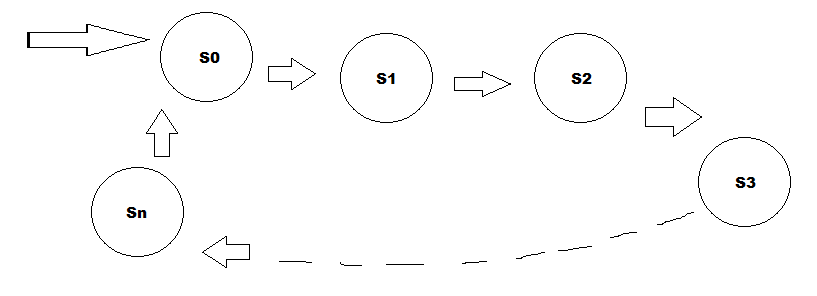


Figure . FSM state diagram

The machine will start in S0, which will be the point of back if reset hits (reset-1). In my project, the FSM is a Moore state machine where the output depends only on the present state. The output will be 4 signals, each signal is different from the other. The output is 2-bit which represents the color of the light. Each state has different case of outputs and different delay.

* 1. N-bit counter

The counter in this system is consistent of 3 inputs (reset, clock, go) and 1 output (Cout) as shown in fig.2.2. The reset returns the system to the starting point, which is S0.the output, consists of n-bit depending on how many states do we have. In my project, I have 18 states so the Cout would be 5-bit. The counter basically is a number of flip-flops with clocks.

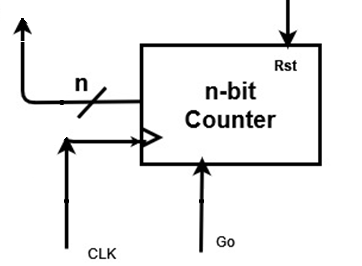
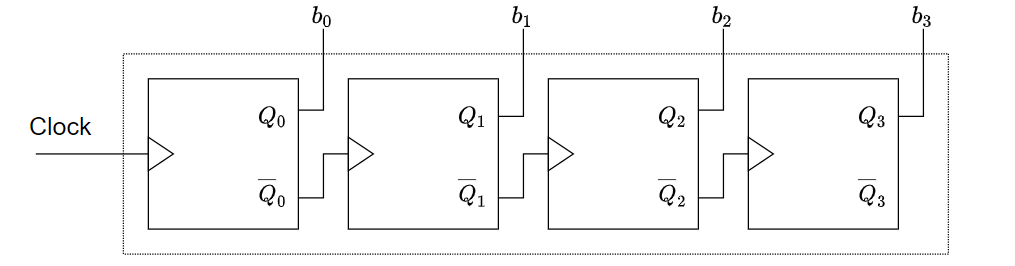


Figure . n-bit counter structure

In the system of traffic light controller, the state while not transit to the next state unless there is a car in the street, which can be presented as go=1. If go=0 then the state will stay in its place without moving, in other words the counter will stop until the go goes to 1.

## Structure code

In the system I have designed, I didn’t use the counter since the delay of each state is different from the other. In case the delay is equal, the system will contain the counter. I used the FSM. The inputs are (go, clock, reset) and outputs are (highway signal 1, highway signal 2, farm way signal 1, farm way signal 2, and present state (which will be used for verification)).

## Verification and testing

The other part of the system aims to verify the performance of the circuit built previously using a complete test. It depends on 2 modules.

**Module memory:**

The module takes 1 input: address, and 1 output: data.

The data here represents {hw1,hw2,fw1,fw2}, where hw is highway signal 1 & 2, and fw is farm way signal 1 & 2. The address is the state of the machine.

I entered the data for each state in memory array in respect to its state position, so that it can compare if the results of states are true and has no faulty. The code is shown in fig.2.3.

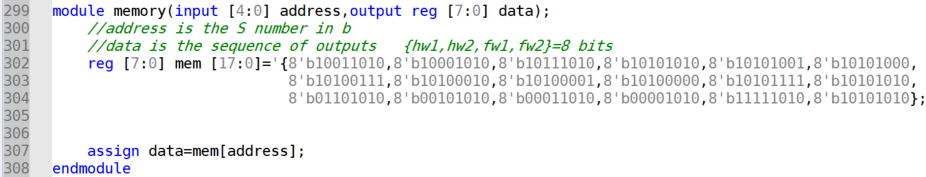


Figure . Memory module

**Module tb\_traffic:**

In testbench of traffic module (our machine), previously timescale was defined as 1s / 1ms. It takes ENDTIME parameter for the time so far. It passes the clk, rst, go, and present state (for verification). At each edge of the clock, (Actual value of state of traffic module) is compared to (Expected value of state of memory module), then it will be shown if it is true or not via message containing the state, actual and expected value. The testbench will also print the time, and outputs for each light signal of the 4 streets. The full code of tb\_traffic is shown in the fig.2.4. First, rst is set to 1 so that to start from State0. Go is set to 1 so that to continue transferring from state to state.



Figure .4 tb\_traffic module

# Simulation & Results

## Test output

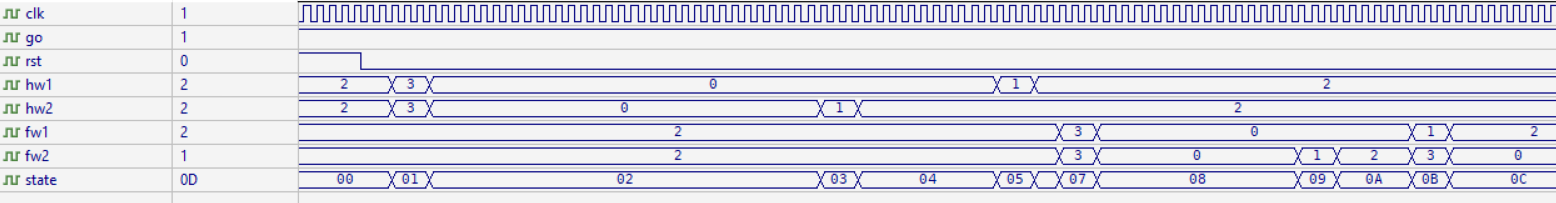


Figure . Results of testbench 1

As shown in fig.3.1, rst firstly was set to 1 so that to start from State 0, then changed to 0 to continue transferring. “state” shows which state we are in hexadecimal. Signal are represented in decimal: 0 for green, 1 for yellow, 2 for red, 3 for red\_yellow. Go is set to 1 to test all states. There is no glitches in the result.

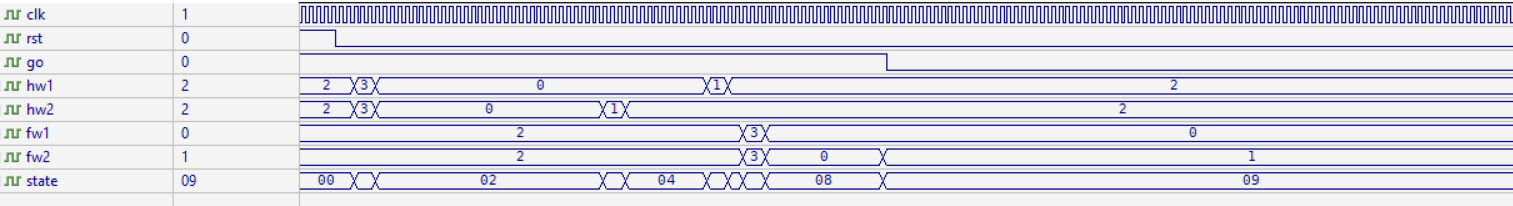


Figure . Results of testbench 2

As shown in fig.3.2, go was changed to 0 after period of time. As we can see, the machine stopped on state 9 as long as it still 0. There is no glitches in the result.

## System verification no-faulty

For this part, the test bench module tb\_traffic in fig.2.4 was simulated. This part depends on both testbench and memory module and their synchronized clocks. As the verification will happen at the positive edges, where the inputs change, regardless of the change of go or rst.

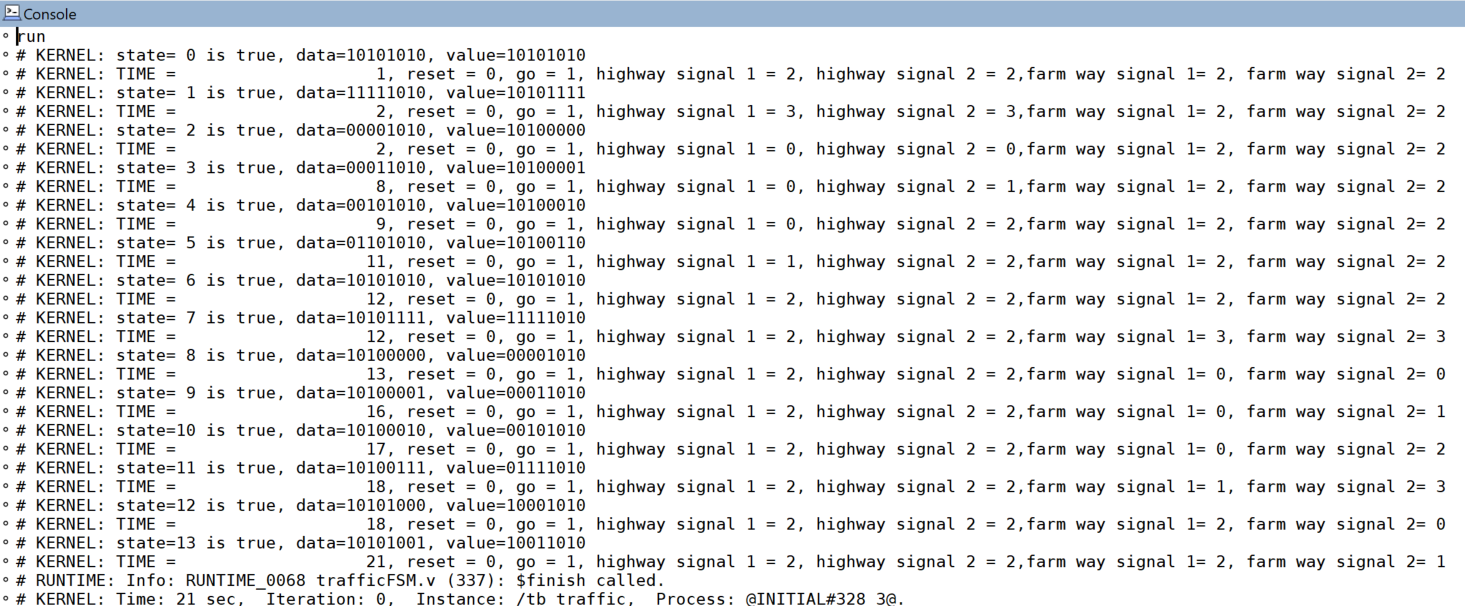


Figure . Console results no faulty

As shown in fig.3.3, all states are true (no faulty), since the expected value equals actual value. The simulation shows from state 0 to state 13.

## System verification with faulty

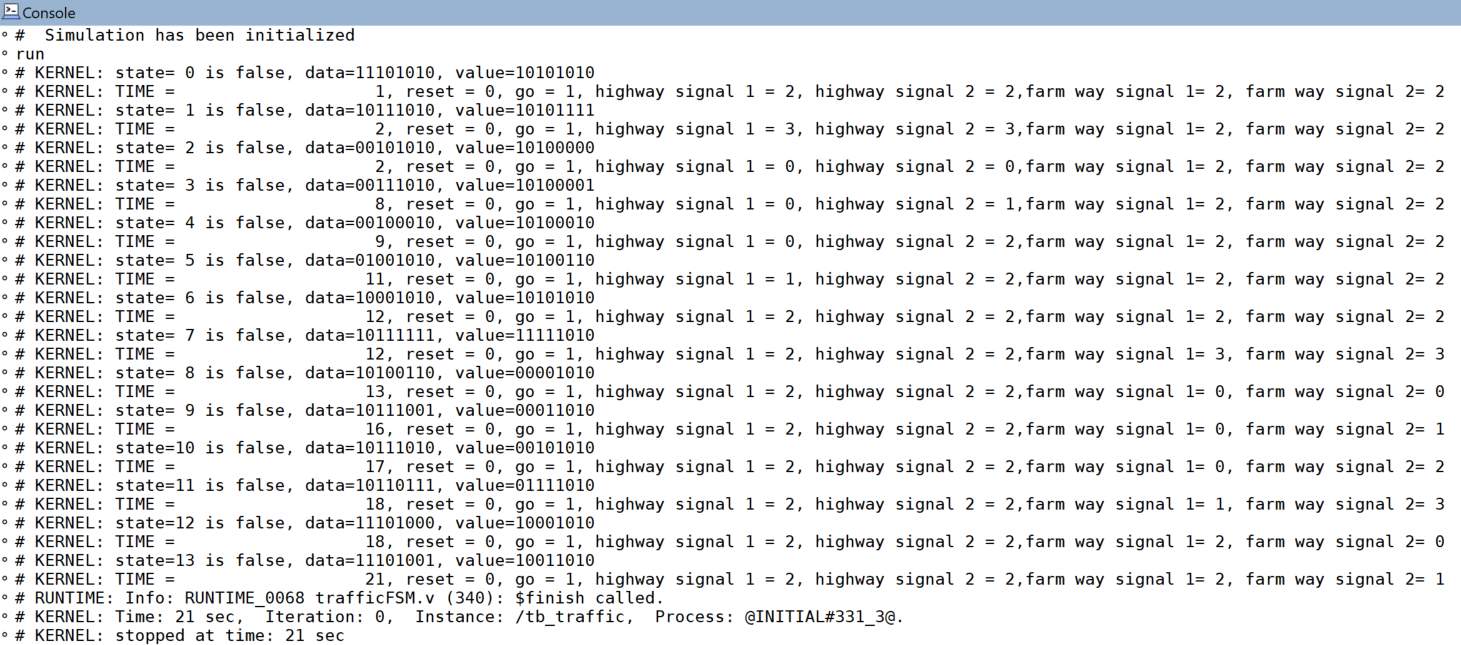


Figure . Console results faulty

As shown in fig.3.4, I changed in the inputs of the memory and made them incorrect. The results are false for all states. The actual and expected values are shown and not the same.

# Conclusion and future works

Finally, building a finite state machine for traffic light controlling for 4 streets of the same crossroad was an application for synchronous circuit with 18 states. I have understood the functionality of the FSM better then simulated the circuit to get the required results. In the future, it would be better if using counter with the machine if there are many states. Final thoughts, it was a great work for applying our knowledge in concepts of synchronous circuits and verifying the work and testing it.