|  |  |
| --- | --- |
|  | |
|  | |
| Assignment 4 – Traffic Light Controller | |
| **Module code:** | **EG3205** |
| **Module name:** | **Programming Microelectronic and Multi-Core Systems** |
|  | |
| 27 March 2019 | |
|  | |
| **Author(s):** | **Nishat Tarannum** |
| **Student ID(s):** | **169018358** |
| **Degree:** | **MEng Software and Electronic Engineering** |
| **Tutor/Project supervisor:** | **Dr. Tim Pearce** |
|  | |
| **SUPERVISOR’S COPY/EXAMINER’S COPY [For 3rd/4th year projects only - delete as appropriate]** | |

By submitting this report for assessment I confirm that this assignment is my own work, is not copied from any other person's work (published or unpublished), and that it has not previously been submitted for assessment on any other module or course.

I am aware of the University of Leicester’s policy on plagiarism, and have taken the online tutorial on avoiding plagiarism.  I am aware that plagiarism in this project report may result in the application of severe penalties up to and including expulsion from the University without a degree.

Contents

[1 My Implementation 1](#_Toc2297780)

[a) Code for the test bench 1](#_Toc2297781)

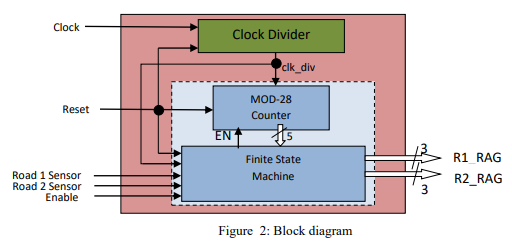
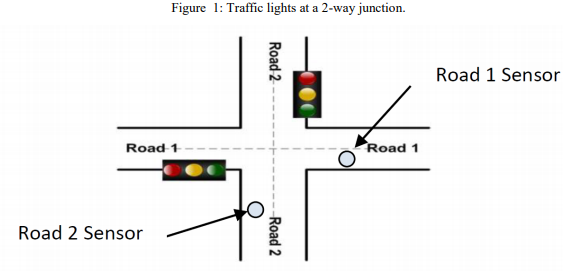
[b) Code for the board 3](#_Toc2297782)

[2 Test Bench 4](#_Toc2297783)

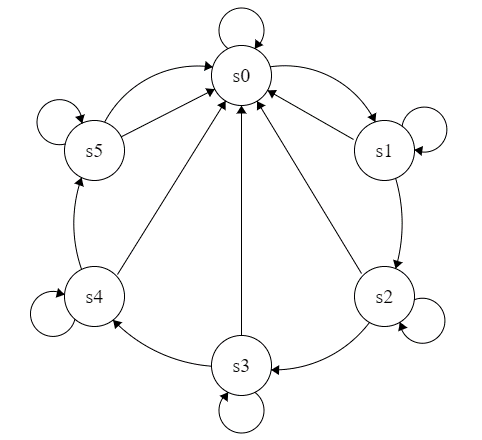
[3 Conclusion 6](#_Toc2297784)

# My Implementation

The assignment asked for a traffic light controller which controlled a junction of two roads, using a finite state machine, mod-28 counter and a clock divider. The clock divider was already implemented and the counter and finite state machine had to be coded.



## State Diagram



reset = 0 or enable = 1

enable = 1 or road\_sensor\_2 not pressed

enable = 1

enable = 1

enable = 1 or road\_sensor\_1 not pressed

enable = 1

enable = 0 and count > 1

enable = 0, road\_sensor\_2 has been pressed and count > 11

enable = 0 and count > 13

enable = 0 and count > 15

enable = 0 and count > 27

reset = 0

reset = 0

reset = 0

reset = 0

enable = 0, road\_sensor\_1 has been pressed and count > 25

R1\_RAG = 011

R2\_RAG = 001

R1\_RAG = 100

R2\_RAG = 001

R1\_RAG = 010

R2\_RAG = 001

R1\_RAG = 001

R2\_RAG = 011

R1\_RAG = 001

R2\_RAG = 100

R1\_RAG = 001

R2\_RAG = 010

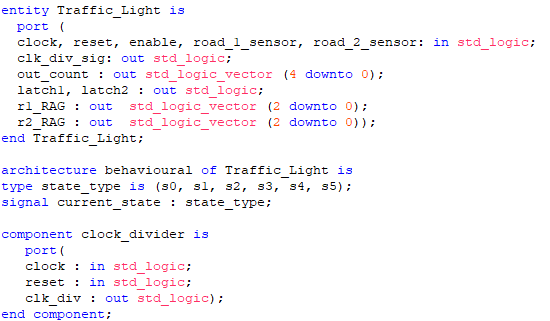
Figure 3 - Finite State Machine

## State Transition Table

Table 1 - State Transition Table

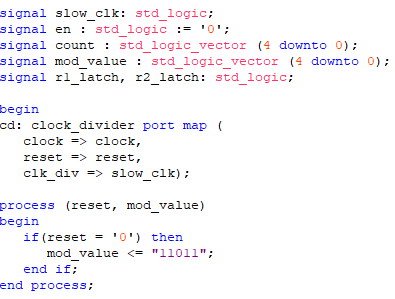


## Explanation of Code



The clock divider provided is instantiated as a component in the counter architecture, which takes a clock signal and outputs a slowed down clock signal. A new state type is also made for the finite state machine.

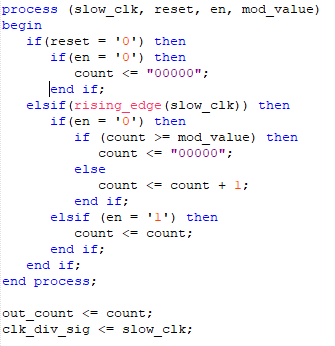
First, the entity of the traffic controller was set up with out\_count being the 4-bit output of the counter and clk\_div\_sig was used to see how the clock divider signal was working, it was used to help understand the test bench. r1\_RAG and r2\_RAG are the traffic light outputs. latch1 and latch2 are output merely for checking if the two latches were working appropriately. The other signals are used to control the counter behaviour according to Figure 2.



Six other signals are set up: the clock divider output is fed into the counter to which refers to slow\_clk, the count is used to hold the temporary value of the count e.g. if it is “1001” etc and, mod\_value stores the set maximum value that the counter will count up to, en (which controls the continuity of the count) is controlled by the enable and the latches. It’s initialised to 0 to make sure the counter immediately counts when a simulation is run. r1\_latch and r2\_latch are the signals which store the latch values.

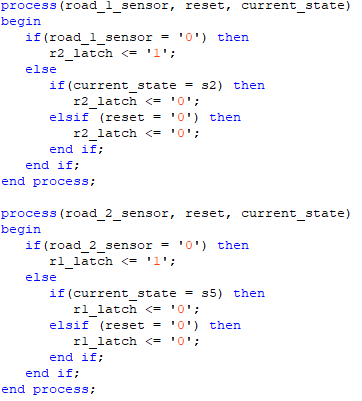
The port map of the clock divider is set up which assigns which of the inputs and outputs are connected to the inputs or outputs or signals of the counter.

For the modulo register, an asynchronous reset has been used this doesn’t require a clock signal to set the modulo value, which has been set to 27.



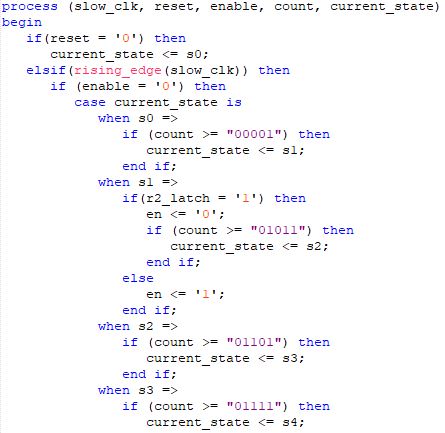
If the reset is 0 and the signal en is enabled, then the counter is set to “00000”. Otherwise, if there is a rising clock edge and en is enabled, then the counter increments up to the mod-value, in this case 27. Otherwise if the signal en is disabled then the count remains constant. This is using an asynchronous reset.

Again, these signals have been used as outputs purely for understanding the behaviour in the test bench.

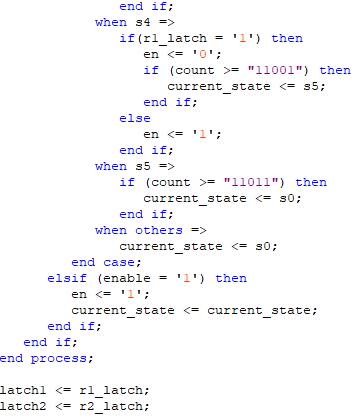


These are the latches which will store the value when a road sensor has been triggered. So, for road\_sensor\_1, if it is triggered then the value is stored until it reaches state s2, otherwise the traffic lights remain in state s1. Or if the reset button is pressed in which case the latch value is reset to 0.

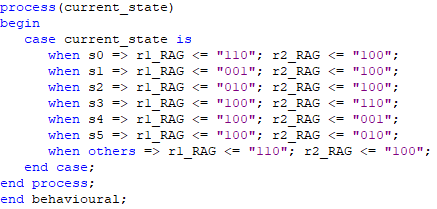
For road\_sensor\_2, if it is triggered then the value is stored until it reaches state s5, otherwise the traffic lights remain in state s4. Or if the reset button is pressed in which case the latch value is reset to 0.



The finite state machine first checks if reset has been pressed, in which case we return to s0. If there is a rising edge of the clock and enable is set to 0, then we look at if the counter value is beyond a certain value. If it goes beyond this value, we move to state s1. If road sensor 1 isn’t triggered the state remains in s1 and the counter is stopped at the latest value. If it is triggered, then we continue to state s2 after 10 seconds, then to s3 after another 2 seconds and then to s4 after another 2 seconds. Again, if road sensor 2 isn’t triggered the state remains in s4 and is stopped at the latest value. If it is triggered, then we continue to state s5 until the cycle restarts. If at any point in the cycle enable is set to 1 then we stop the counter and stay in the current state.

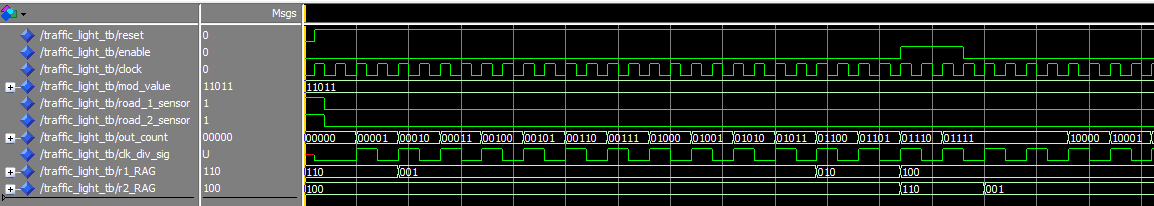


These latch values are used as outputs to see the behaviour in the test bench.



This process sets what the outputs of each traffic light should be for each state, where MSB is RED and LSB is GREEN.

# Test Bench

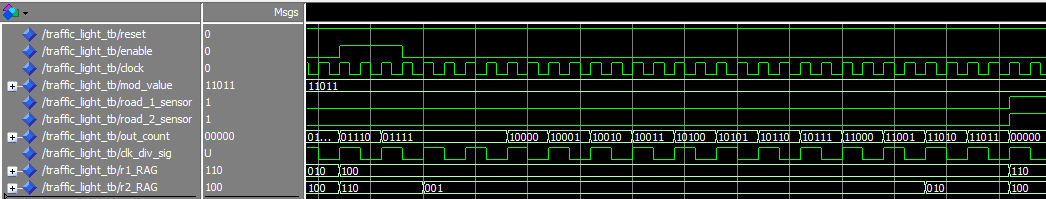


State s2

State s1

State s0

Figure 4 - Timing diagram showing state changes from s0 to s3



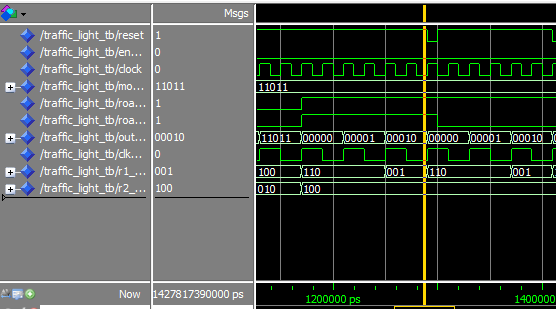
State s4

State s5

State s3

Enable is switched off, however there is a delay of one clock cycle

Figure 5 - Timing diagram showing state changes from s3 to s5 and enable function

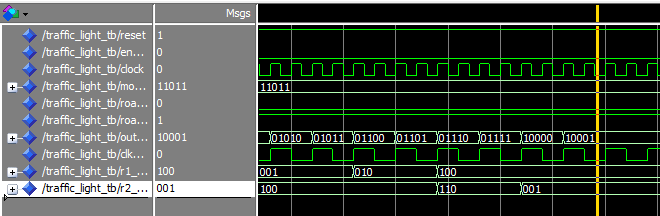


Road sensor isn’t triggered so counter stops counting.

State s0

Reset

Figure 6 - Figure showing reset of counter



Stays in s4

Road sensor isn’t triggered so counter stops counting.

Figure 7 - Figure showing counting stops in state 4

# Conclusion

These are the various tests run which show that the design with the asynchronous reset is working correctly on the test bench and the board. I have implemented a mealy machine which is why the enable is slightly delaying the output thus the output is asynchronous.