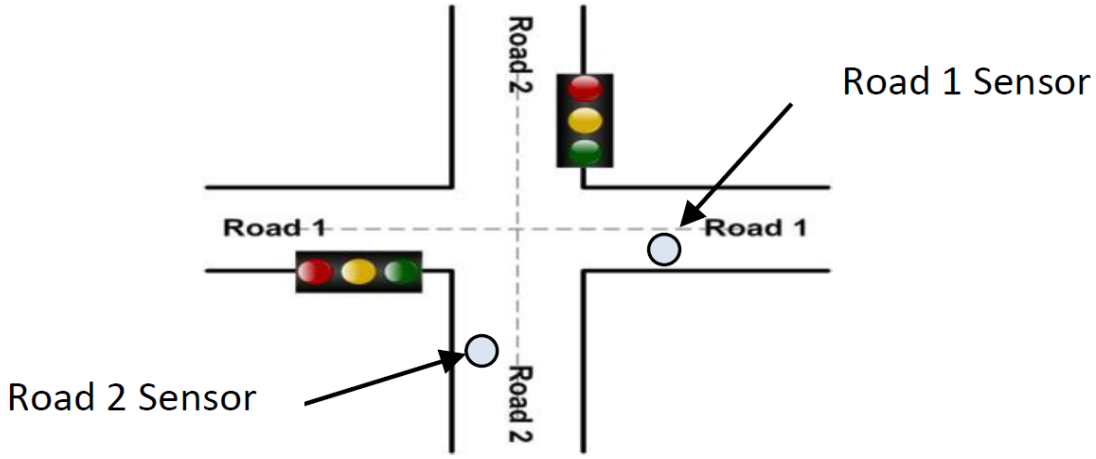
EG3204/5 FPGA Assignment 2
2-Road Junction Traffic Light Controller

1 Introduction

For the second assessment, you will design, implement, and simulate a Traffic Light Controller for a junction of two roads (Road 1 and Road 2) with two traffic sets of lights as shown in Figure 1. Each road has a corresponding vehicle sensor (Road 1/2 Sensor) that needs to be triggered (active high for each vehicle detected) before the traffic lights for that road change state (otherwise they will stay at RED). You should use a Moore or Mealy finite state machine (FSM) for your design.

Figure 1: Traffic lights at a 2-way junction.



2 Individual exercise

This is an assessed exercise. Your solution will constitute 25% of your overall mark for EG3204. You may work on your solution in the lab hours, and you may also work on it in your own time.

While you are encouraged to discuss aspects of the design with others, you must attempt the writing of the VHDL as an individual exercise. You may not share your VHDL solution, or part of your solution, with other students or report. Any submissions (or parts of solutions) that are found to be shared between students will be dealt with under the University policy for collusion and/or plagiarism, and may result in a module mark of zero and/or loss of credit for the module or even downgrading of your final degree. Details of what you are required to submit, how you should submit your work, and deadlines for submitting your work are given at the end of this sheet.

During the laboratory sessions reduced demonstrator support will be available, but because this is an individual exercise demonstrators will not be debugging code.

3 Design of the traffic light controller

The Traffic Light Controller is a sequential circuit. Each state represents a particular configuration of the traffic lights ON/OFF values. State changes occur on the rising edge of the clock (clk_div) either after a 2 second interval (whenever the AMBER light is active) or a 10 second interval (whenever RED and GREEN lights are active together), but state changes only occur when the Enable switch is set to active LOW/OFF and the corresponding vehicle sensor has been activated at least once since the last GREEN light was active on that road (see below). If the Enable is inactive (HIGH) the controller should stay in the current state regardless of input. Each traffic light sequences through the standard UK configurations RED, RED+AMBER, GREEN, AMBER, and then back to RED), and it is the job of the controller to co-ordinate the sequences of each of the two traffic lights safely, such that both lights can never be at GREEN at the same time. The part of the sequence RED, RED +AMBER, GREEN is only triggered after the corresponding road sensor is active at least once since the last GREEN light was active on that road). When the road sensor has not been activated the state of the lights should be frozen. The starting state is Road 1: RED + AMBER, Road 2: RED which is held for 2 seconds.

The cycle time of the entire sequence is 28 seconds, assuming that the road sensors on both roads have been triggered by the user at the appropriate times to allow state changes. Therefore the controller contains a MOD-28 counter which increments on positive edges from the clock divider. The controller state will be changed synchronously on the positive edges of clk_div. The controller can also be reset to the starting state when the Reset button is pressed. The block diagram for the traffic light controller is given in Figure 2. The block diagram consists of three blocks of digital logic, with various external and internal signals.

• Clock Divider : Divides the 50 Mhz clock to allow operation on the board or during simulation.. You are supplied with the VHDL code that does this. Change the LIMIT value inside the clock divider block to slow down or speed up the system. LIMIT needs to be adjusted to drive the counter at one second intervals on the board and LIMIT=2 to reduce simulation steps within ModelSim.

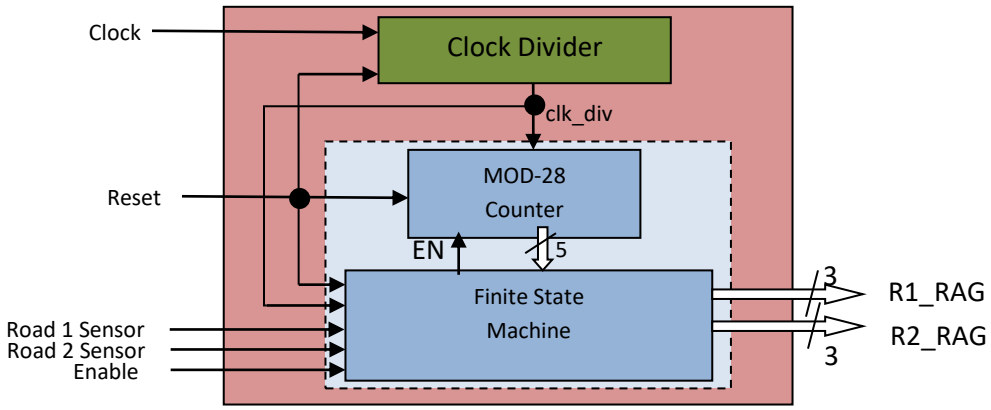


Figure 2: Block diagram

• MOD-28 Counter : The modulo counter logic. You are required to implement this block.

• Finite state machine : This is the heart of the Traffic Light Controller. This block contains all the state transition logic of a traffic light finite state machine, including state register, next state logic and output logic. The traffic light outputs are generated from this block. You are required to implement this block.

1. Clock : A standard 1-bit 50 MHz clock signal.
2. clk_div : A second clock signal that pulses once every second (1 Hz) by adjusting the LIMIT value.
3. Reset : A standard 1-bit reset line, that puts the traffic light controller back into the starting state Road 1: RED+AMBER, Road 2: RED. You should decide whether a synchronous or asynchronous reset is appropriate.
4. Enable : The signal that causes the traffic controller to stay in active (low) or inactive (high) states.

5. Road 1 Sensor: Input that is triggered active (high) when a vehicle passes and inactive low (otherwise). Must be triggered at least once after each transition to GREEN in order for the traffic lights for the corresponding road to progress from RED to RED + AMBER, back to GREEN …

6. Road 2 Sensor: Input that is triggered active (high) when a vehicle passes and inactive low (otherwise). Must be triggered at least once after each transition to GREEN in order for the traffic lights for the corresponding road to progress from RED to RED + AMBER, back to GREEN …

7. EN : A 1-bit control signal, that enables the MOD-28 counter to count UP. When EN is active (low) the count is enabled, otherwise the current count value and state of the lights are frozen.

8. R1_RAG : A 3-bit bus (STD_LOGIC_VECTOR) that outputs the GREEN, AMBER + RED values for Road_1. LSB and MSB are GREEN and RED respectively.

9. R2_RAG : A 3-bit bus (STD_LOGIC_VECTOR) that outputs the GREEN, AMBER + RED values for the Road_2. LSB and MSB are GREEN and RED respectively.

4 Building a test bench

You will need to build and implement a test bench so that you can test your design in simulation. You should consider all scenarios of input that you need to test so that you can ensure that your design is functionally correct. The testbench should use a combination of functional and regressive testing strategies to achieve an exhaustive testing of your digital system. When you are confident that your design works correctly, you can load your design onto the development board.

5 Using the development Board

Your solution should download and run on the development boards that we have been using in the lab. Your logic will connect to the peripherals on the board that we have been using in this module. You are given the pin configuration file that does this (on Blackboard), and can use this file in the same way as previous exercises. The assignment of the various input/outputs of your design to the FPGA board peripherals should be as follows:

1. Clock : connects to the clock divider logic.

2. Reset : connects to BUTTON0

3. Road 1 Sensor: connects to BUTTON1

4. Road 2 Sensor: connects to BUTTON2

3. Enable : connects to SW0

4. R1_RAG : connects to the lower side of LED array in the order RED (LEDG0)), AMBER (LEDG1), GREEN (LEDG2).

5. R2_RAG : connects to the higher side of LED array in the order RED (LEDG7), AMBER (LEDG8), GREEN (LEDG9).

6 Documenting and reporting on your design

You are required to produce a short report (normally a few pages in length).

Your report should first give a state diagram for your design and a state transition table. Consider if you can reduce the state table by merging states to make a simpler state machine.

Your report should include timing diagrams to show the different scenarios of testing – draw arrows onto the timing diagrams to indicate key events in the circuit behaviour (and whether correct or incorrect).

If there are errors in your design that you have found in simulation but not been able to debug, you can take the opportunity to explain them in your report. Credit will be awarded if you are able to demonstrate that your test bench either fully demonstrates that the design works, or that you have managed to expose an error and have some understanding of what the error is and why it may be arising.

7 Assessment submission requirements

Your submission will consist of 3 items:

1. Your VHDL source code for your design. If the board and simulation codes are in different files please submit both.
2. Your complete VHDL test bench;
3. Your report.

You are not required to submit compiled files such as bit files, or any of the files that you have been supplied with for this assignment. Your report should include timing diagrams generated from the behavioural simulation in ModelSim.

You should submit your assignment electronically, via the blackboard site for this module (in the Assessment section). Late submissions will be adjusted according to the university regulations. Incomplete submission will only earn credit for those components that have been submitted.

8 Marking Breakdown

Grades for this assignment will depend equally on the quality of your state diagram solution, VHDL code, technical achievement in the completeness of your test bench and its ability to exhaustively test the design, and the understanding that you demonstrate in your report/demonstration.

TCP