Computer Architecture 2024-25

Assessed Exercise 1

— State Machine Circuit —

Introduction

The purposes of this exercise are to learn how to design a simple synchronous digital circuit that implements a state machine, how to specify it in a hardware description language, and how to test it via simulation. The specification and simulation will use Hydra, a functional computer hardware description language.

The exercise

Design and implement

- Two circuits, as described below, which should all be in a file *TrafficLight.hs*.
- A simulation driver for each circuit; both of these should be in a file named *TrafficLightRun.hs*
- Suitable test data that demonstrates the correct functioning of each circuit, which should be included in *TrafficLightRun.hs*.
- There should be a main function in *TrafficLightRun.hs* that runs all of your test cases.

Informal specification of the circuit

The circuits are traffic light controllers. There are two versions.

Version 1

The circuit *controller1* has one input, a bit called reset. This would be connected to a pushbutton. The reset button should be pushed once to start the circuit, and then in normal use it would never be pressed again. We model this by defining the value of the reset input bit to be 1 during the first clock cycle, and 0 thereafter.

There are three outputs, each of which is a bit. The outputs correspond to green, amber, and red, and they determine whether the corresponding traffic light is on. At all times (after reset has been pressed and the circuit is running) one of the three output bits should be 1, indicating that the corresponding traffic light should be on, and the other two bits should be 0. The outputs should run through a fixed sequence: green, green, green, amber, red, red, red, red, amber, and then it repeats.

Version 2

The second version, *controller2*, is intended for a pedestrian crossing. There are three lights for traffic (green, amber, and red) and also two lights for the pedestrian (wait and walk). There are two input bits: a reset pushbutton, and a walkRequest which is 1 when a pedestrian presses the Walk pushbutton.

Normally the outputs indicate green/wait. However, when the walkRequest button is pressed, the traffic light changes to amber, and then the traffic light changes to red and the pedestrian light turns to walk for three clock cycles. Then the system displays amber/wait and then returns to its normal state green/wait.

Furthermore, the traffic engineers want to know how often the walkRequest button is pressed. To measure this, there is a 16-bit counter walkCount. When the Reset button is pressed, the value of WalkCount is set to 0 at the next clock tick. When the walkRequest button is pressed, walkCount should be incremented at the next clock tick.

To summarise, the inputs are: reset (a pushbutton) and walkRequest (a pushbutton). The outputs are: green, amber, red, wait, walk (each is 1 bit), and walkCount (a 16-bit binary integer).

You may notice that the specification of the problem doesn't say what to do in a few subtle situations. For example, what should happen if the walkRequest button is pressed when the system is in the red/walk state? You may adopt any reasonable policy you like for these situations. In the real world, you (the engineer) might go back to the customer to discuss what to do. It is common for the requirements of a project to evolve during the implementation. Actually, that's one of the reasons that good engineering includes making your solution maintainable as well as correct.

The point of this exercise is to gain some experience with a simple control circuit, and to use some of the building block circuits. The exercise as described doesn't account for some real world issues, which you can ignore:

- In the real world, the reset button and the walkCount display would be hidden inside the box containing the electronics, while the walkRequest button would be out where a pedestrian could see and press it, and the various light outputs would control the actual light bulbs.
- In the real world the clock cycles would be too short, so each light would be held for a larger number of clock cycles. For example, the amber phase might last for a million cycles and green/red would be correspondingly longer.

Work individually

This assessed exercise should be carried out individually. You should submit the exercise on Moodle.

Handin

Submit your solution via the Moodle page; see the page for detailed instructions on submission.

Your submission should consist of a single file named either *Exercise1.tgz* or *Exercise1.zip*. Other formats, such as rar, are not acceptable. This file defines a directory named Exercise1. It is good practice that unpacking your file should produce a *directory* — it is unprofessional and bad style just to dump a lot of files into the user's directory. The directory Exercise1 should contain the following files:

- *StatusReport.txt* — a text file that gives your name and matriculation number, as well as your status report
- *TrafficLight.hs* — the circuit definition file, containing the two circuit specifications
- *TrafficLightRun.hs* — definition of the simulation drivers as well as suitable test data. This should define *main :: IO ()* so that running main will execute your tests for both circuits.

The status report, *StatusReport.txt*, should be a plain text file. The opening lines should follow a key-value format, where each keyword is followed by a colon : and a space, and then the value is just text. You should define the keys that are illustrated in the following example of a status report. It should include the author line that contains your name, followed by a comma, followed by your matriculation number. Here's an example of the format:

```
course: CA
exercise: 1
date: 2024-12-11
author: Jane Doe, 9875434
```

After these opening lines, the status report should describe the status of your solution. It should say whether each part of the exercise has been completed. Does it compile? Does it appear to work correctly? How did you test it? Are there any aspects that appear to be not quite right? Are there any especially good aspects you would like to highlight?

If your status report contains several paragraphs, separate them with a blank line. You can explain your approach to solving the problem in the status report, or you can include your documentation as comments in the source program.

Assessment

The exercise will be marked out of 100 marks, and it counts for 10% of the CA assessment. The two exercises together are worth 20% of the assessment, while the examination is the other 80%.

On this exercise, Version 1 and Version 2 have equal weight (each is worth 50 marks). Assessment will be based on (1) handing something in; (2) a reasonable approach to the problem; (3) a correct solution; (4) good style in the circuit design; (5) a working simulation driver; (6) suitable test data.