Practical 11: Finite State Machines

This practical is all about implementing Finite State Machines (FSMs) in VHDL.

We will begin by looking at a prepared code example for a set of traffic lights (Sections 1 and 2). Following that, you will be asked to write your own VHDL code and testbench for an FSM for a hot water dispenser, based on a state diagram (Section 3).

1: Simulating the Traffic Light Controller (example code)

This first section involves working with prepared code for the traffic light controller. The functionality of the controller is described in the lecture notes.

(a) For the first part of the exercise, download the following files from MyPlace:

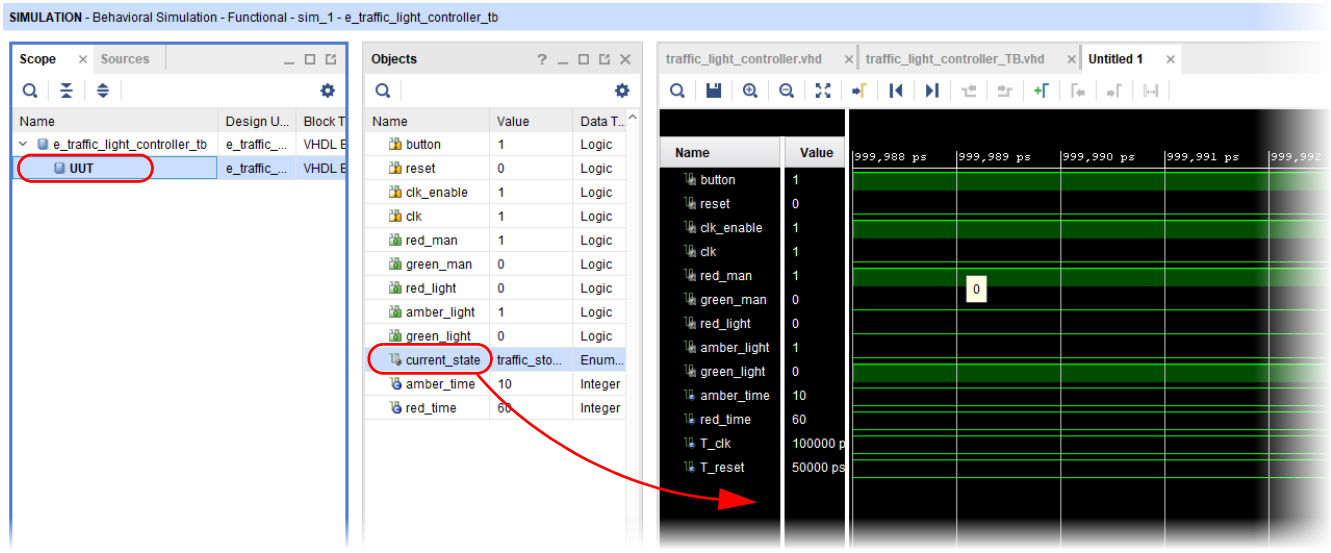
- **'traffic_light_controller.vhd'** and
- 'traffic_light_controller_TB.vhd'

(b) Create a new Vivado project, and add both files.

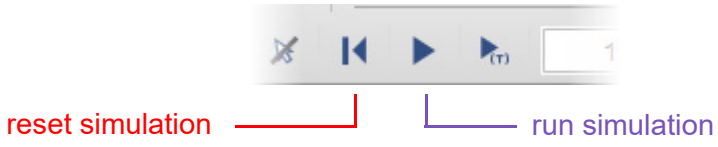
(c) Next, inspect the code, and make sure you understand how it corresponds to the state diagram provided. In doing so, consider the following questions...

- How many VHDL processes have been used in the VHDL code for the FSM, and what do they each do?
- How many states does the FSM have? Can you see any correspondence between the number of states, and the 'case' statements?
- Why do you think generics been used in the design of the FSM, and what do these control in terms of the FSM's operation?

(d) Run a simulation and view the results. It may be useful to inspect the 'current_state' signal to the waveform. As this is an internal signal (i.e. it exists within the UUT, but not within the testbench level of hierarchy), then it is not automatically added to the waveform. To add it manually: (i) high-light the UUT in the left hand panel waveform window, then (ii) drag the 'current_state' signal from the middle panel to the right hand (wave) panel.



(e) After that, reset the simulation and run it again, using the blue buttons in the toolbar:

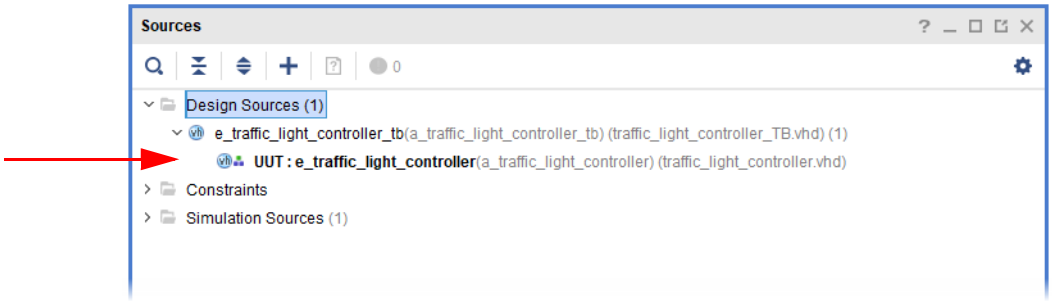


(f) You should now be able to see not just the inputs and outputs changing value, but also the 'current_state' of the FSM. Inspect the waveform and ensure the observed behaviour corresponds to the behaviour you expect, based on the VHDL code.

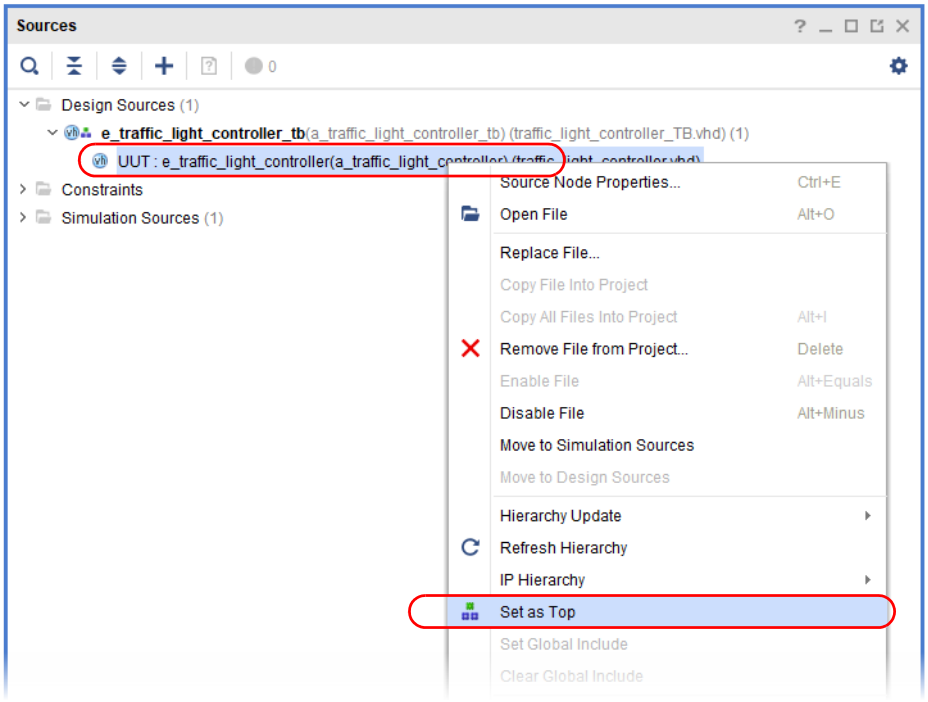
2: Synthesising the Traffic Light Controller (example code)

We will now consider synthesis of the example code for the traffic light controller FSM.

(a) First, ensure that 'traffic_light_controller' (rather than 'traffic_light_controller_TB') is defined as the top level in the *Sources* pane. This is important because it is the FSM that we wish to synthesise — not the testbench (which indeed is not intended to be synthesised!). The FSM code file should be shown in bold, with a blue and green symbol denoting it as the top level design:



(b) If it is not already denoted as the top level, right click on the UUT item and then choose the option ‘Set as Top’, as shown below. This will change the configuration for synthesis, such that the UUT becomes the top level, as desired.



(c) Next, try to run synthesis. You might find that it initially fails — why? Use the errors messages to find out what is wrong, and then correct the problem, save the file(s), and re-attempt synthesis (hint: the issue may have something to do with generics!).

(d) Once synthesis has completed, choose the option to inspect the synthesised design, and open the schematic. How many registers have been used to hold the state ‘current_state’? You may now that further registers are synthesised for the counters inferred from the design.

(e) By inspection of the schematic, can you confirm that the design implements a Moore machine? Check in particular that the outputs depend only on the current state.

3: VHDL Coding of an FSM from a State Diagram (Hot Water Dispenser)

For this section of the Practical, we will consider the VHDL coding of a Moore type FSM for a hot water dispenser. The behaviour of the FSM is completely described by the state diagram in Figure 1, along with the table of outputs for each of the states (Table 1). The inputs and outputs of the system (corresponding to the VHDL entity declaration) are confirmed by the block diagram in Figure 2.

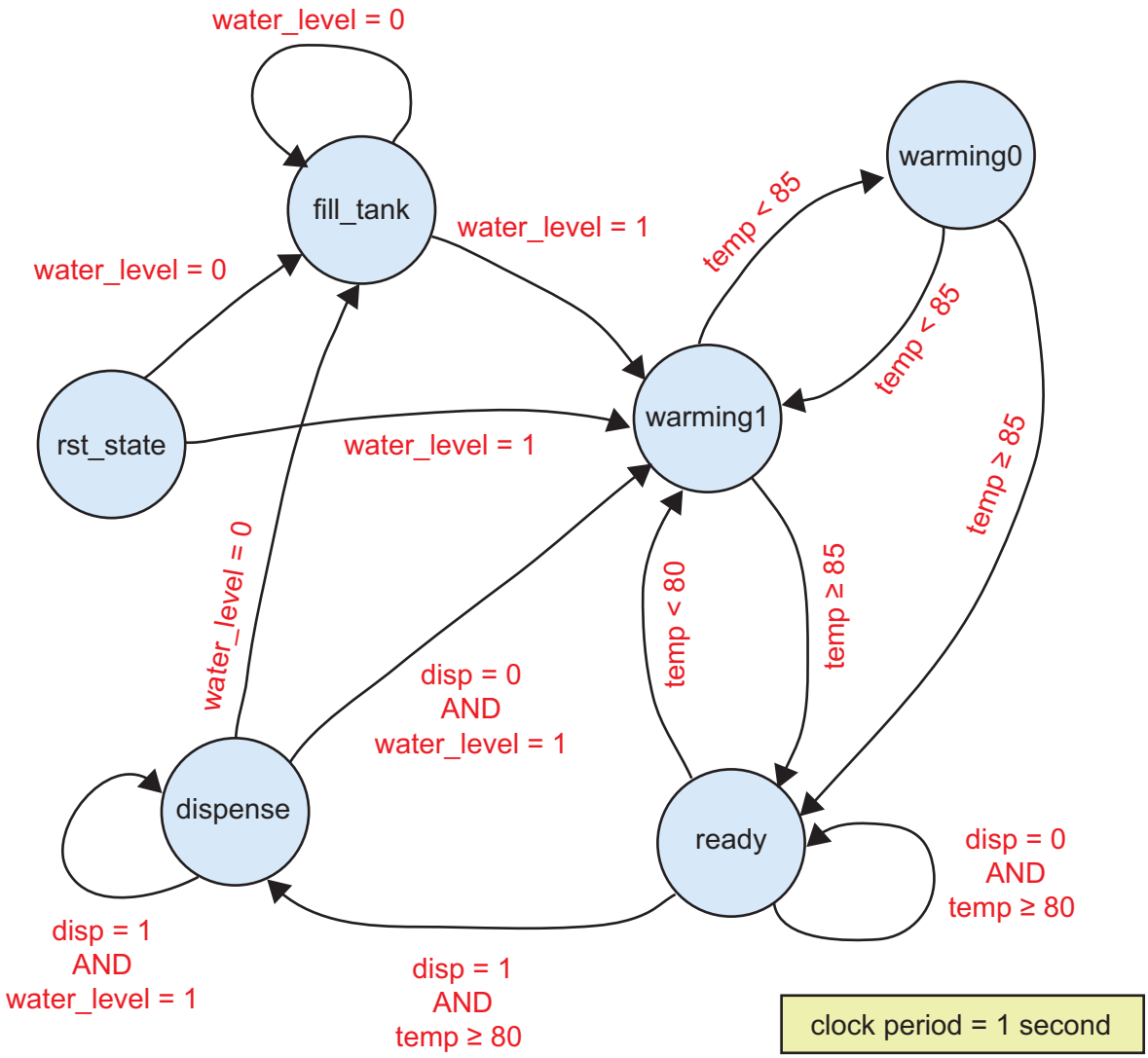


Figure 1 : State Diagram for the Hot Water Dispenser FSM

Table 1: Outputs in each of the FSM States

State	Outputs						
	LEDs(0)	LEDs(1)	LEDs(2)	LEDs(3)	heater	pump_in	pump_out
rst_state	1	1	1	1	0	0	0
fill_tank	1	0	0	0	0	1	0
warming0	0	0	0	0	1	0	0
warming1	0	1	0	0	1	0	0
ready	0	0	1	0	0	0	0
dispense	0	0	0	1	0	0	1

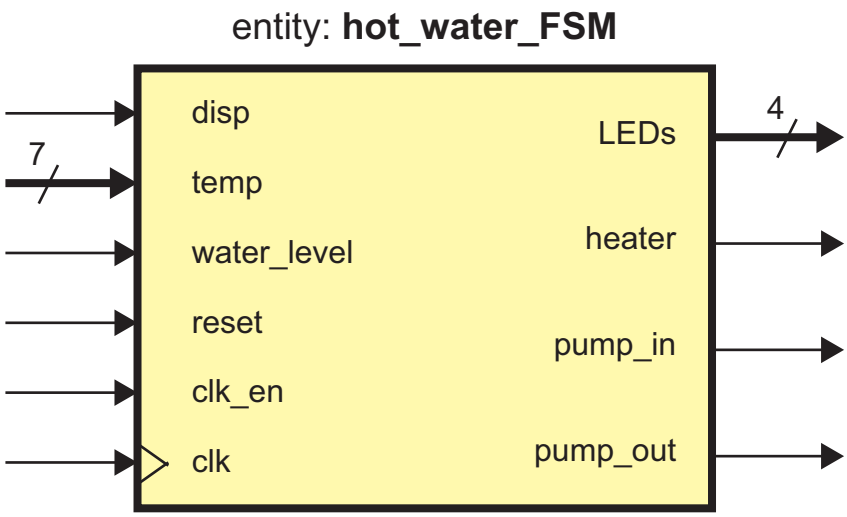


Figure 2 : Top Level Diagram for the Hot Water Dispenser FSM

(a) Consider the state diagram carefully and consider how the FSM behaves. Write down a description of operation. What do you think each of the LEDs are intended to show? And why do you think the transitions between some of the states are based on slightly different temperatures? Additionally, record any assumptions you have made.

(b) Write code for the **hot_water_FSM** design (entity and architecture), which you can base on the code for the traffic light controller.

(c) Develop VHDL for the testbench (again, you can use the traffic light example as a starting point). When writing the testbench, think carefully about how the hot water dispenser is intended to operate, and write your stimuli process appropriately. ***Hint:*** *Writing the testbench for this FSM is reasonably challenging, as the inputs it generates will be determined by an output of the FSM — think, how should the temperature of the water (an input to the FSM) change, based on the heater (an output of the FSM) being switched ON or OFF?*

(d) Make sure that all of your VHDL is appropriately commented!

(e) Once your code is complete, run a simulation and ensure that the FSM operates as expected. If necessary, make adjustments to your FSM code, or the test stimulus.

(f) It would be a good idea to copy and paste all of the VHDL code and waveforms you have generated into your logbook, along with any notes that may be useful.

(g) Now, try synthesising your circuit, and hence ensure that it can be successfully converted into hardware.

(h) Lastly, return to your design and consider how VHDL generics could be used to make the design more flexible. For instance, what if your water dispenser was required to heat water to a minimum temperature of 65 degrees, rather than 80? Make appropriate changes to your code, and repeat the processes of simulation and synthesis, to validate your design.

4: Summary

FSMs are an important component in many digital electronic designs. They are used to coordinate actions and respond to sequences of events. To do this, they hold state, i.e. a register within the FSM keeps track of the current state.

In this practical, we have looked at a prepared example, and you have also written and tested VHDL code for an FSM, based on a state diagram.