Lab Report 6

Testing the FSM from Lab 4

Sven Pfiffner and David Zollikofer

Modifications

There was a problem we faced: Our computer could never simulate multiple seconds, we just don't have the power for that. In order to solve this, we have had multiple possible solutions:

- directly pass the new clock to the FSM and do not use the clock divider.
  - We didn't do this since this would have resulted in changing the FSM's inputs which is something we didn't want to do since we wanted to test it exactly as it has been on the board.
- Wait for a couple of days for the simulation to complete.
  - not practical
- Rent a cloud based computer (e.g. Azure or AWS) and run the simulation there
  - Wouldn't change a thing since simulation primarily depends on single core performance, see https://forums.xilinx.com/t5/Simulation-and-Verification/How-to-change-Vivado-simulator-settings-to-use-multiple-cores/td-p/907661
- Change the clock divider
  - This is the solution we have used, it is a very small modification and doesn't change the internal workings of our FSM. (The FSM uses the clock divider as a "black box" therefore as long as the "black box" works, we will be able to simulate the FSM).

This is the change we have made:

```verilog
module clk_div(input clock, input reset, output clock_en);
    reg [24:0] clk_count;
    always @ (posedge clock)
        begin
            if(reset)
                clk_count <= 0;
            else
                clk_count <= clk_count+1;
        end
        assign clock_en = &clk_count;
endmodule
```

to

```
module clk_div(input clock, input reset, output clock_en);
    reg [4:0] clk_count;
    always @ (posedge clock)
        begin
            if(reset)
                clk_count <= 0;
            else
                clk_count <= clk_count+1;
        end
        assign clock_en = &clk_count;
endmodule
```

This allows us to make the simulation go faster and actually see what is going on in the simulation window with our own eyes. However we would like to point out that this does not change the outcome of the simulation in any significant way (except for the time it takes to transition from state to state) since the FSM uses the clock divider as a black box.

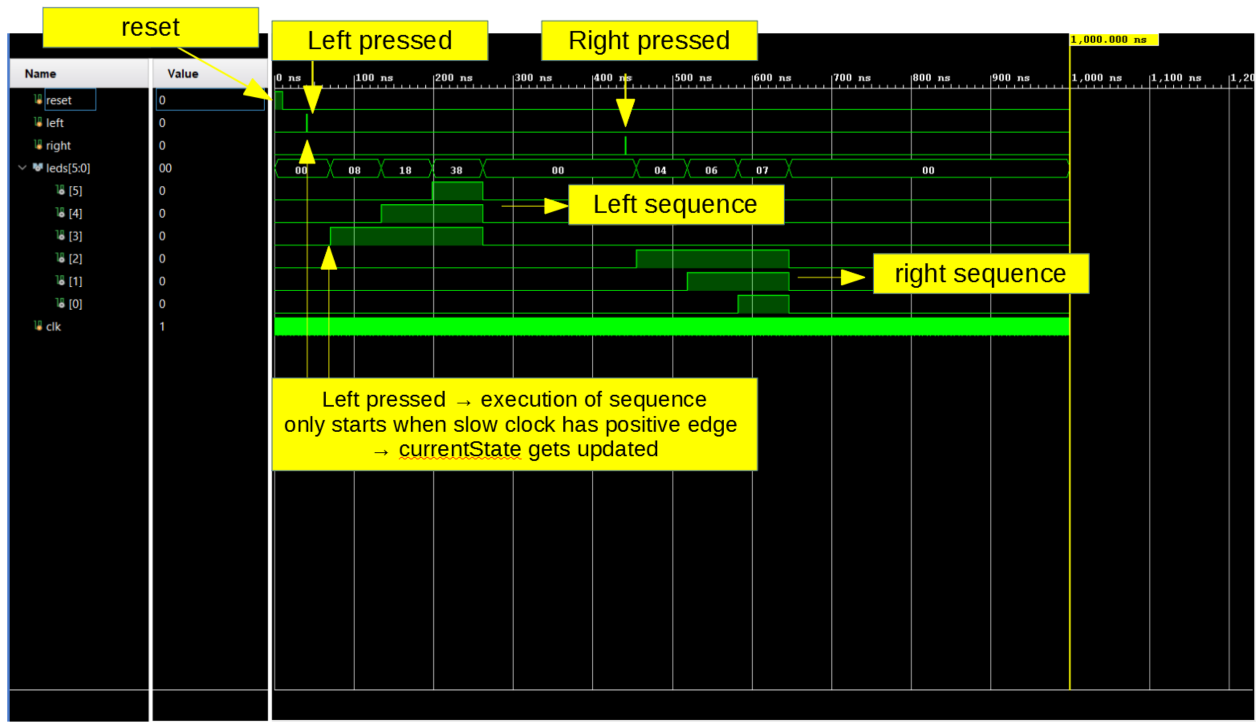
Testbench

We have taken the testbench code from lab 6 but we have heavily adapted it. One one hand, this report does not require that we automatically check the state transitions with testvectors, but instead want's us to focus on the outputs of the waveform diagram.

See next page:

Results:

When we run the testbench we get the following output: (a copy of the output that is not annotated can also be found in this folder):



First we issue the reset. This doesn't really have any visible effects on the output, but resets the FSM to its initial state.

We the press the left button for a nanosecond. This doesn't cause the FSM to immediately enter the sequence to the left. Instead, internally, the FSM's nextState will be set to the first state to the left and only after the slow clock from the clock divider has a rising edge the left sequence starts (this explains the gap between the moment when left is pressed and the sequence starts).

After the left sequence has finished we wait a bit and do the same to the right.

For completeness sake we have also added a picture showing only the right sequence (since this is what you are asking for in the report.)

Feedback

In the lab report you write "Enable and disable the reset signal during the initialization phase. You also need to pass the clock signal to the FSM module.", this can be understood in two ways. Either just pass the fast clock to the FSM as we have done it in the Lab or to not use a clock divider and just use one clock. We have used the former, but from reading the sentence it was not clear to use which one you meant.

We have worked very hard on this lab and this report. If you have any questions feel free to send us an email to svenp@student.ethz.ch or zdavid@student.ethz.ch .