End-Semester Examination
Mealy FSM using the Krypton CPLD

Dhruv Ilesh Shah — 150070016

April 26, 2017

Overview

In this experiment, we were expected to design a sequential circuit which can implement a given Mealy FSM.

The code was compiled on Quartus Prime, and simulated using ModelSim. GHDL was also used for simulation purposes, at a low level. This was then uploaded to the *Krypton v1.1* 5M1270ZT144C5 CPLD-based board.

The algorithm and setup has been covered in section 1. The VHDL codes have been kept modular and as generic as possible, for reusability and code clarity. Section 2 presents the simulation observations and scan-chain test results.

1 Setup & Algorithm

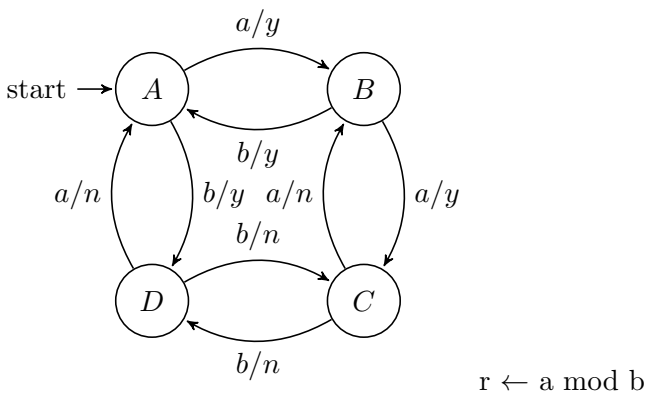


Figure 1: Automata Representation for *myFSM*

The state encoding is given in table 1. The encoding for input and output is as given in the question paper. The entity looks like this.

```
entity MyFSM is
        port (reset, x1, x2, clk: in std_ulogic;
                y1, y2: out std_ulogic);
end entity;
```

Since we were only expected to use *nor*2 gates, I created the other gates in terms of nor2 as follows.

State	q_1	q_0
A	0	0
B	0	1
C	1	0
D	1	1

Table 1: State Encoding for $myFSM$

$$\begin{aligned} \neg A &= NAND2(A, A) \\ A \cdot B &= \neg NAN2(A, B) \\ A + B &= \neg NAND2(\neg A, \neg B) \end{aligned} \tag{1}$$

The other combinational logic blocks can be easily built using the above elements. The inputs are deciphered as follows.

$$\begin{aligned} r &= \overline{reset} \cdot \overline{x_1} \cdot \overline{x_2} \\ a &= \overline{reset} \cdot x_1 \overline{x_2} \\ b &= \overline{reset} \cdot \overline{x_1} \cdot x_2 \end{aligned} \tag{2}$$

State Transitions

$$\begin{aligned} nq_0 &= A \cdot a + C \cdot a + A \cdot b + C \cdot b \\ nq_1 &= A \cdot b + C \cdot b + B \cdot a + D \cdot B \end{aligned} \tag{3}$$

Assigning Outputs

$$\begin{aligned} y_1 &= (A + B) \cdot (a + b) \\ y_2 &= (C + D) \cdot (a + b) \end{aligned} \tag{4}$$

2 Observations

Here, I present the results of running RTL & Gate-Level Simulation on the design. Scan-Chain based tests were also performed.

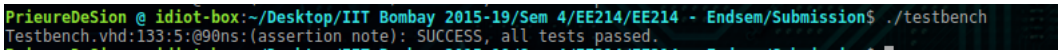


Figure 2: Results of Simulation on GHDL

Figures 3 & 4 show the outputs of the code after running gate-level simulation on Quartus/ModelSim and the Scan-Chain tests.

Conclusion

Starting from the very scratch, in this report, I have presented the logic and code for a sequential implementation of a simple mealy FSM. The logic was tested using RTL simulation, followed by the gate-level simulation for delay analysis and emulating the CPLD. This was followed by an actual rigorous test on the CPLD board after burning the code on it, using the *TIVA-C* microcontroller.

All the cases passed successfully at all stages and hence the complete FSM can be used in hardware, as required.

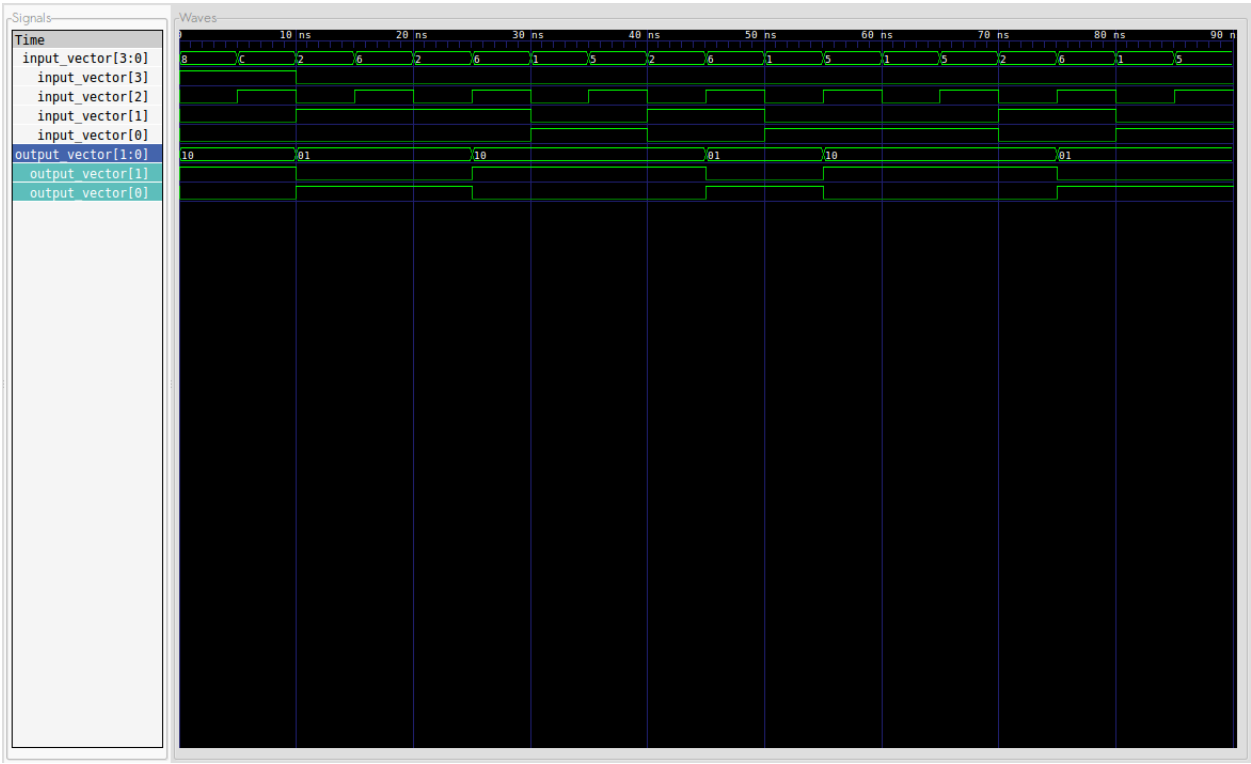
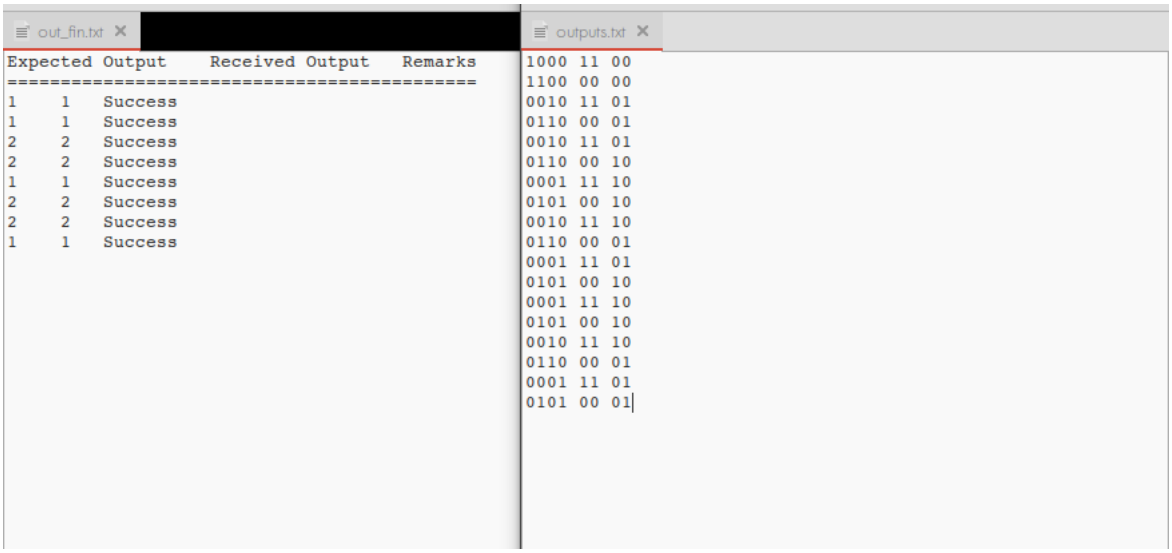


Figure 3: Snapshot of the waveforms.



out_fin.txt

```
Expected Output     Received Output    Remarks
==========================================
1     1    Success
1     1    Success
2     2    Success
2     2    Success
1     1    Success
2     2    Success
2     2    Success
1     1    Success
```

outputs.txt

```
1000 11 00
1100 00 00
0010 11 01
0110 00 01
0010 11 01
0110 00 10
0001 11 10
0101 00 10
0010 11 10
0110 00 01
0001 11 01
0101 00 10
0001 11 10
0101 00 10
0010 11 10
0110 00 01
0001 11 01
0101 00 01
```

Figure 4: Outputs of the Gate-Level Simulation(right) and Scan-Chain test(left) on $myFSM$