

1 Exercise 3

For this exercise we were asked to implement the Moore machine shown on Figure 1, but with one condition, everything inside our machine should work on 3.3v and inputs and outputs should work on 5v logic.

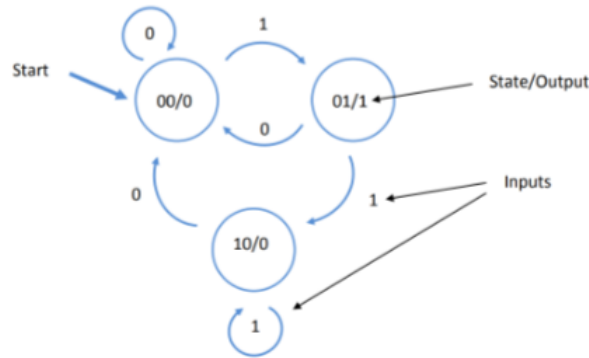


Figure 1: Moore Machine

1.1 State Machine Implementation

We proceeded to convert the diagram to a table that represents it, and we got Table 1, and by transforming each bit (Output, and next states bits) into a truth table, we could create our logic implementation for each bit. The Truth Tables are shown on Tables 2.

State	W=0	W=1	Output
00	00	01	0
01	00	10	1
10	00	10	0

Table 1: Moore Machine Table

		S1	S0	W	S1'	S0'
		0	0	0	0	0
		0	0	1	0	1
		0	1	0	0	0
		0	1	1	1	0
		1	0	0	0	0
		1	0	1	1	0
		1	1	0	x	x
		1	1	1	x	x

S1 S0	Output
00	0
01	1
10	0
11	x

Table 2: Truth Tables

So by re-writing those truth tables we have got the following equations:

$$Output = S_1 \cdot \bar{S}_0 \quad (1)$$

$$S_1' = \bar{S}_1 S_0 w + S_1 \bar{S}_0 w$$

$$S_0' = \bar{S}_1 \bar{S}_0 w$$

So, by implementing those formulas into a synchronized logic circuit, we have got the Moore machine.

1.2 Level Converter Implementation

To convert the voltage levels on the PCB for compatibility, we decided to use a 74LS05 Integrated Circuit. It is an inverter with Open-Collector outputs, that has a $V_{IH_{min}} = 2(V)$, which is ideal for our implementation. By connecting the output to a pull-up network with the voltage we want, we can convert the level with no problems. For the Pull-Up network, we decided to use a resistor $R = 10(k\Omega)$ so that when it's connected to ground, the current flowing through the inverter is less than $10(mA)$, and when the inverter produces a High Z, the resistor is not enough big to produce a High Z to the output, and set the output to a logic 1.

1.3 PCB Implementation

We proceeded to implement the PCB using Altium Designer, the Schematic for this Finite State Machine is shown on Figure 2. The Top and Bottom Layers are shown on Figure 3.

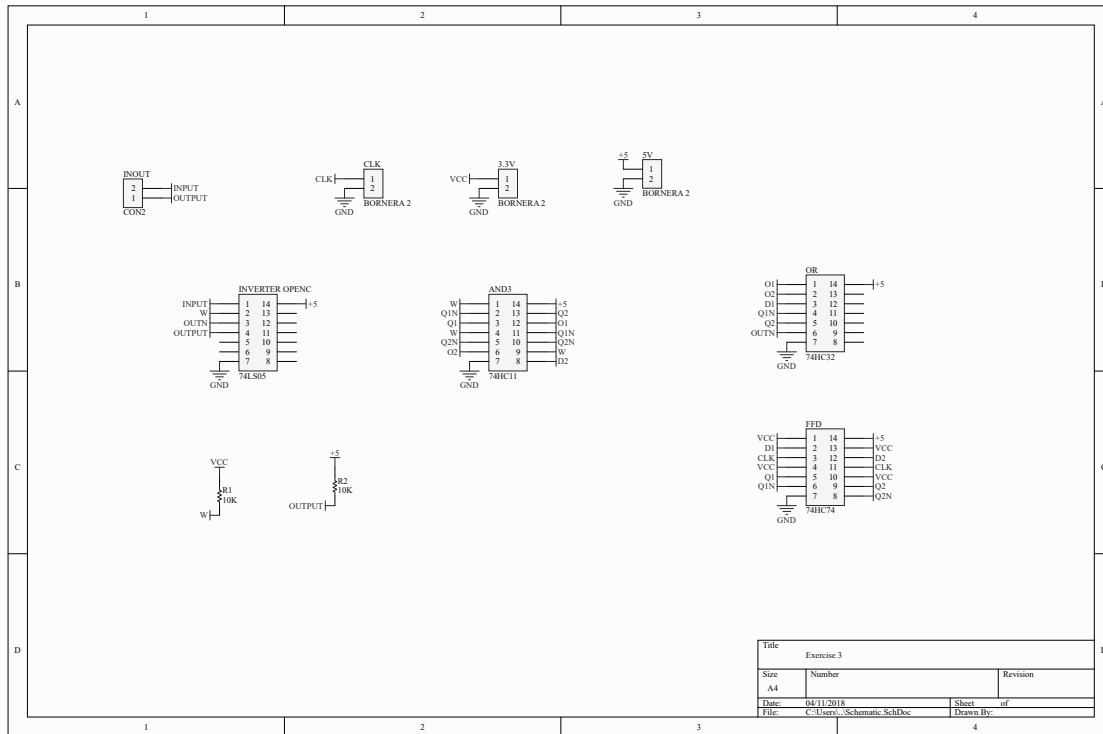


Figure 2: Schematic

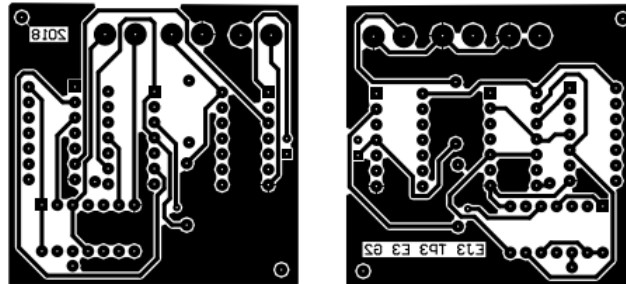


Figure 3: Top and Bottom Layers

1.4 Mealy State Machine Re-Implementation

Finally, we were asked to re-implement the Moore State machine described in the previous Subsections, into a Mealy State Machine. As we know, Mealy Machines Outputs depend on states and inputs, different from Moore that only depends on states. So the truth tables for the next state, maintains its form and formula as the moore state machine, however, the output truth table, and by consequence, its formula , change ash shown on Table 4.

State	W=0	W=1
00	00/0	01/1
01	00/0	10/0
10	00/0	10/0

Table 3: Mealy State Machine

S1	S0	W	Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	x
1	1	1	x

Table 4: Output Truth Table

So the formula for the output becomes:

$$Output = \bar{S}_1 \bar{S}_0 w \quad (2)$$

By comparing the equations (1) and (2), we notice that in this case, Moore machine is more efficient because it utilizes less lotic gates as it needs one less variable to compare.