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.

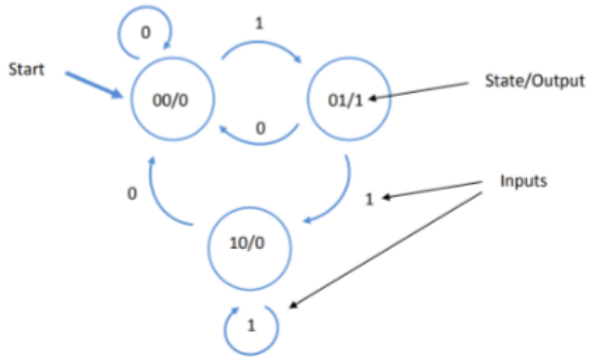

Figure 1: Moore Machine

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	Output
00	0
01	1
10	0
11	x

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

Table 2: Truth Tables

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

$$Output = S_1 \,.\, \bar{S_0} \tag{1}$$

$$S_1^{'} = \bar{S_1} S_0 w + S_1 \bar{S_0} w$$

$$S_0^{'} = \bar{S_1} \bar{S_0} w$$

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 2N7000 Mosfet Transistor. We used it to implement an inverter with Open-Collector. By connecting the output to a pull-up network with the voltage we want, we can convert the level with no problems, form 5(v) to 3.3(v), and vice versa.
For the Pull-Up network, we decided to use a resistor $R = 1\,(k\Omega)$ so that when it's conected to ground, te current flowing through the inverter is less than $10\,(mA)$, and when the inverter produces a High Z, the resistor its 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.

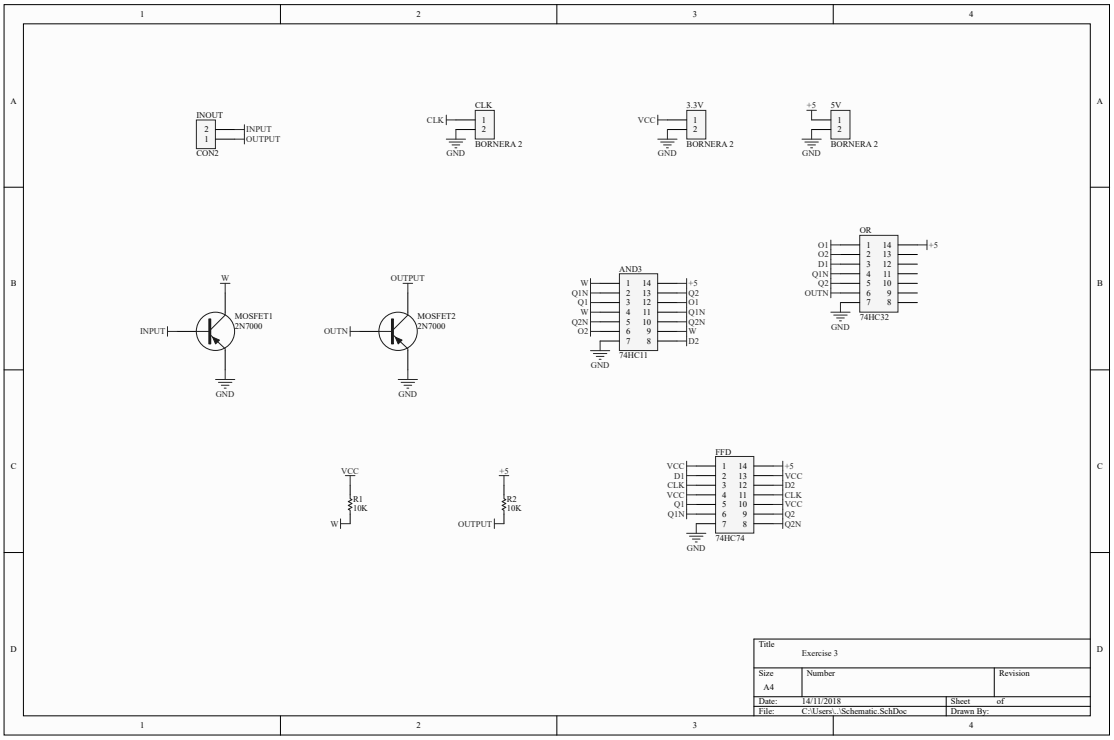

Figure 2: Schematic

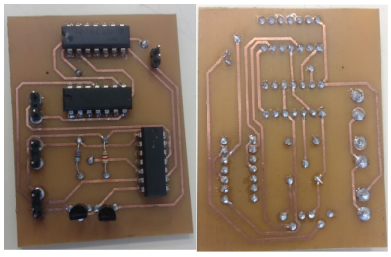

Figure 3: Printed Board

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, mantains its form and formula as the moore state machine, however, the output truth table, and by consequence, its formula , change ash shown on Table 5.

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

We can now easily see that in Table 3, states 01 and 10 are equal, so we can simplify those states into only one state. The final Table is shown on Table 4.

State	W=0	W=1
0	0/0	1/1
1	0/0	1/0

Table 4: Mealy Machine simplified

State	W	Output
0	0	0
0	1	1
1	0	0
1	1	0

Table 5: Output Truth Table

Finally, the formulas for the output and the next states are the following:

$$Output = \bar{St}.W$$

$$State = W$$

Implementing this would become someting like shown on Figure 4.

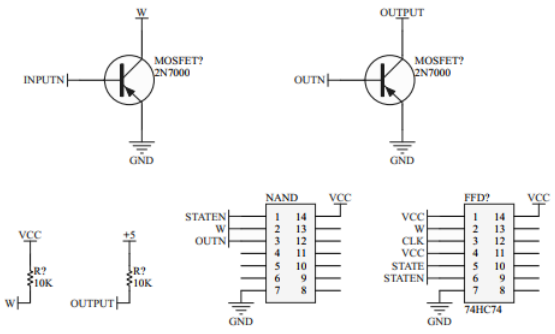

Figure 4: Schematic Mealy Machine

We builded this schematic in a breadboard, as shon on Figure 5.

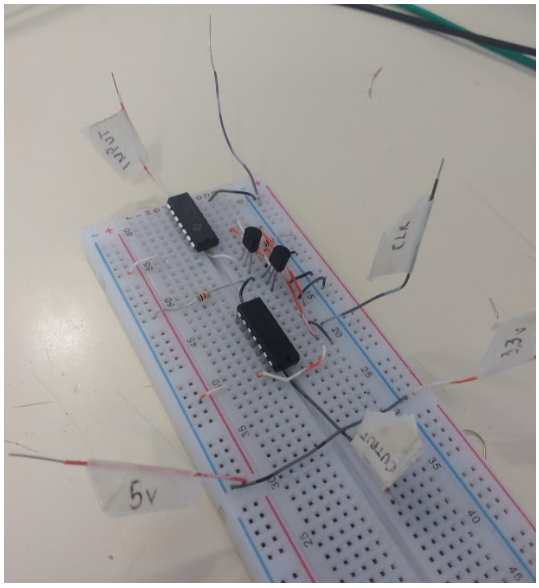

Figure 5: Mealy Implementation

1.5 Conclusions

By analizing the behaviour of both circuits, how they behave in the same test conditions, and comparing the amount of components used in both PCBs, we conclude that in this case, Mealy's implementation is more efficient because it utilizes less components and realizes the same operation. However, we need to clarify that not always Mealy's implementation is more efficient than Moore's, this depends on each state machine to implement.