# Report from Lab 1

# Karnaugh Map and Multiplexers

I declare that this piece of work, which is the basis for recognition of achieving learning outcomes in the Digital Circuits course, was completed on my own.

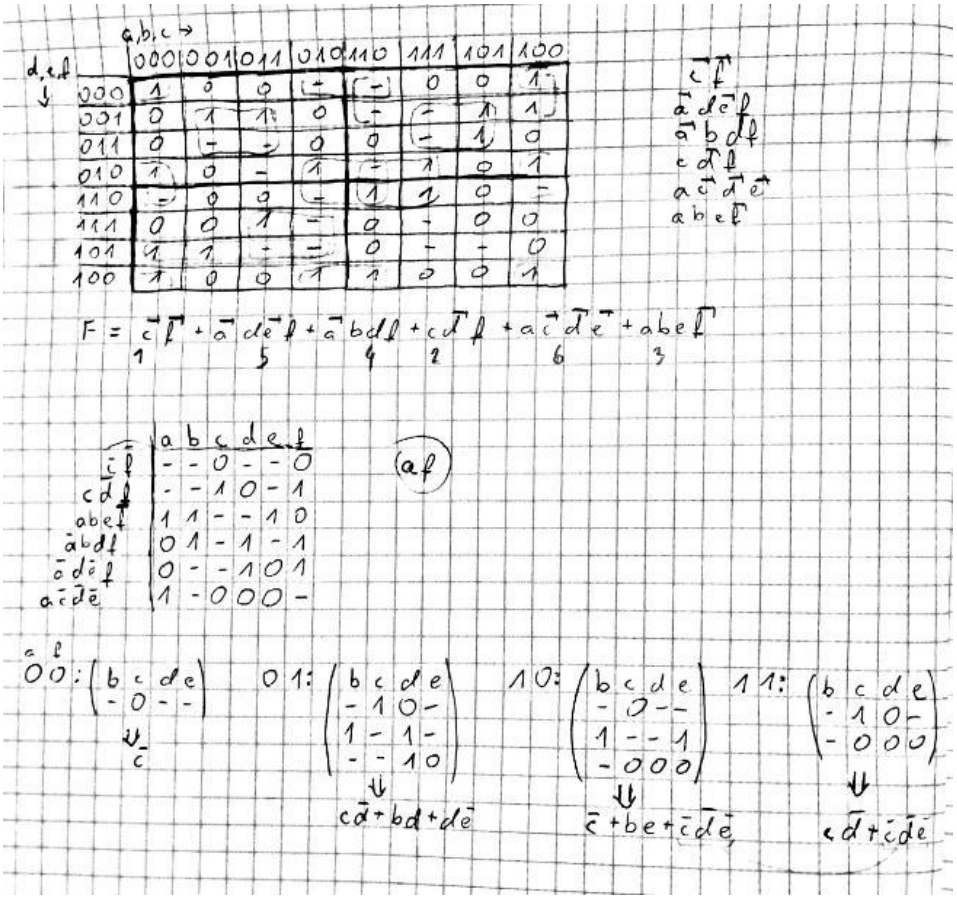
First and last name: Michał Łezka

Student record book number (Student ID number): 303873

Date: 25.11.2020

## Theoretical design on paper

I have started my task from finding prime implicants from the Karnaugh map that was sent to me before the labs the way it was shown to us at the start of the laboratory, by making groups containing 1's as big as possible. Then I also computed how the same map should look like for a 2-bit multiplexer and I did it the way shown during laboratory, first I chose two most common-appearing inputs (it was 'f' with 5 appearances and there were two with 4 appearances: 'a' and 'd', but I chose 'a'). I chose 'a' as most significant digit and 'f' as least significant digit Then computed it as shown below:

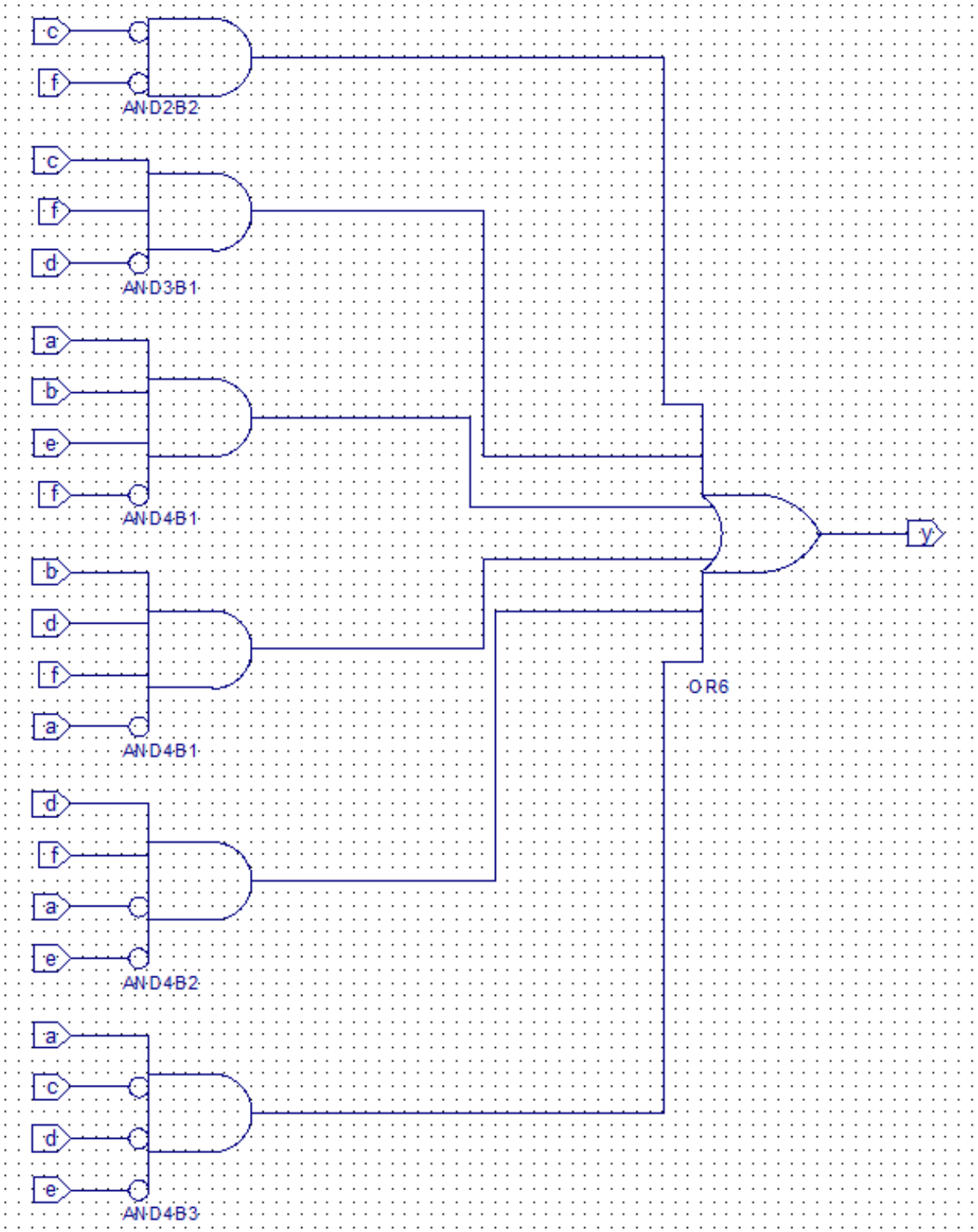


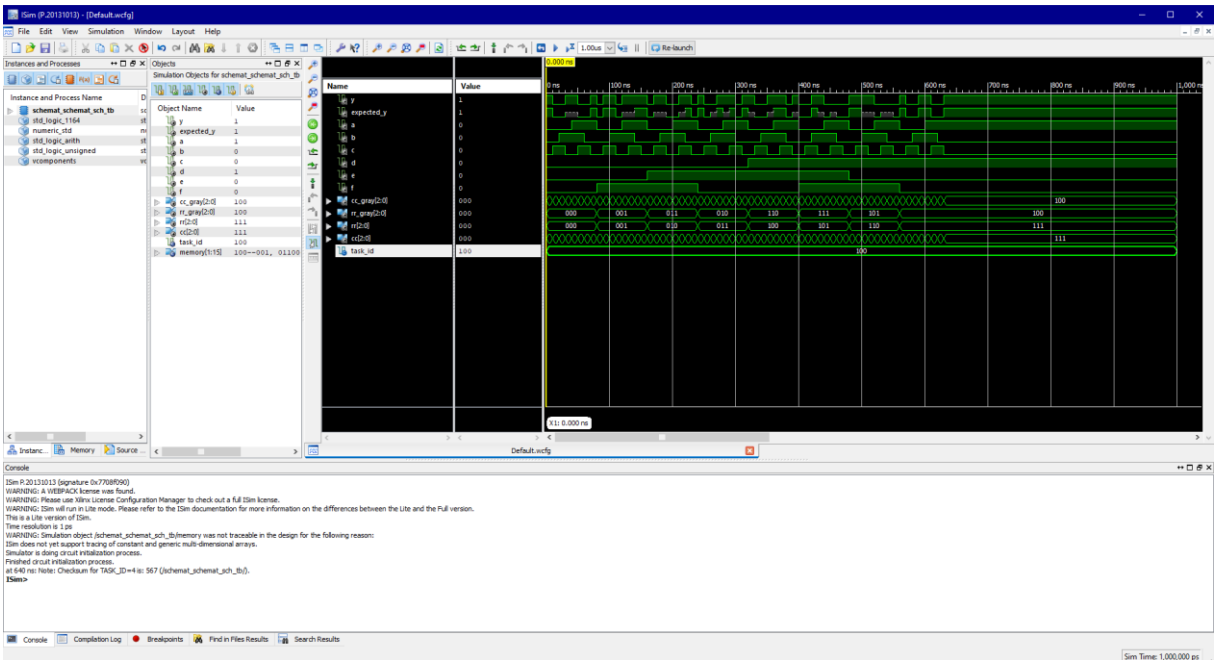

Note: I have numbered the prime implicants (numbers are underneath them) from the shortest to the longest to increase clarity on the schematic in Xilinix.

## Schematics and test results

- **K-Map**

Below is the schematic done according to the theoretical design in the earlier paragraph in the order I have mentioned in the note below the scan. Each prime implicant is computed in an AND gate, but each AND gate had different size and some inputs had negation before it (the circle was showing which inputs had the negation).

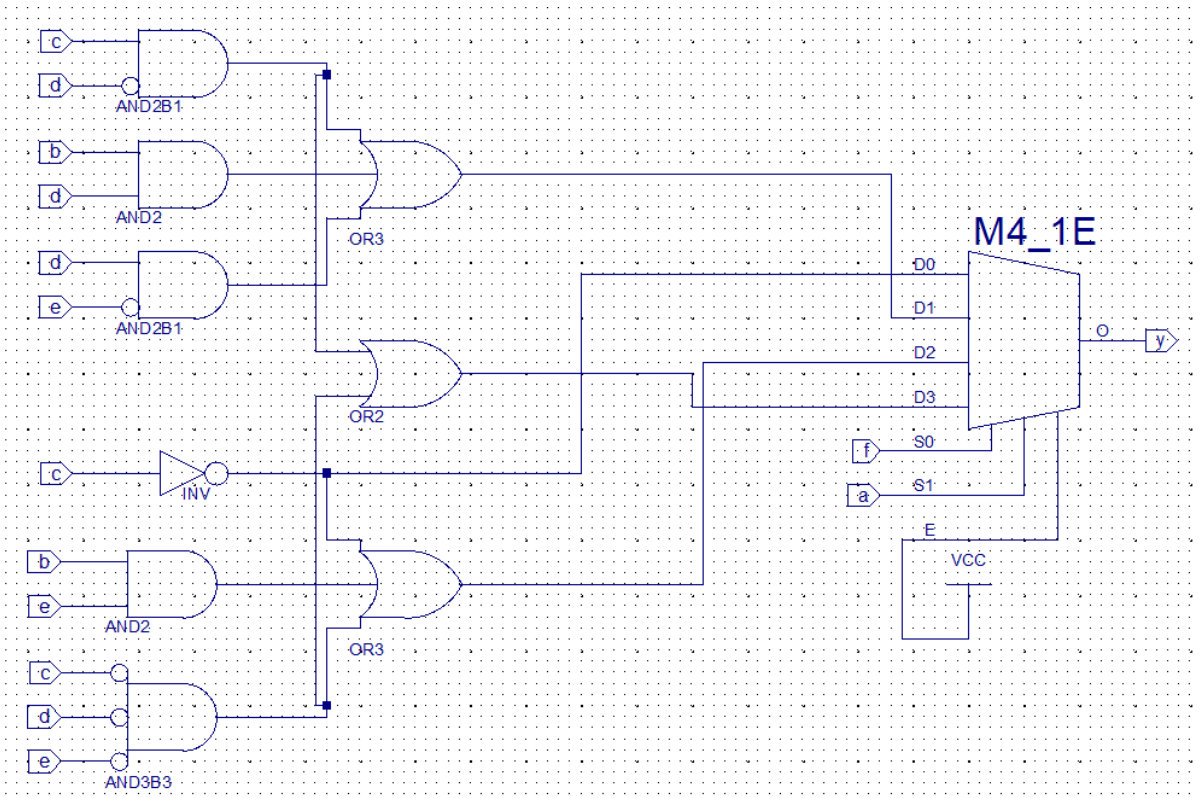


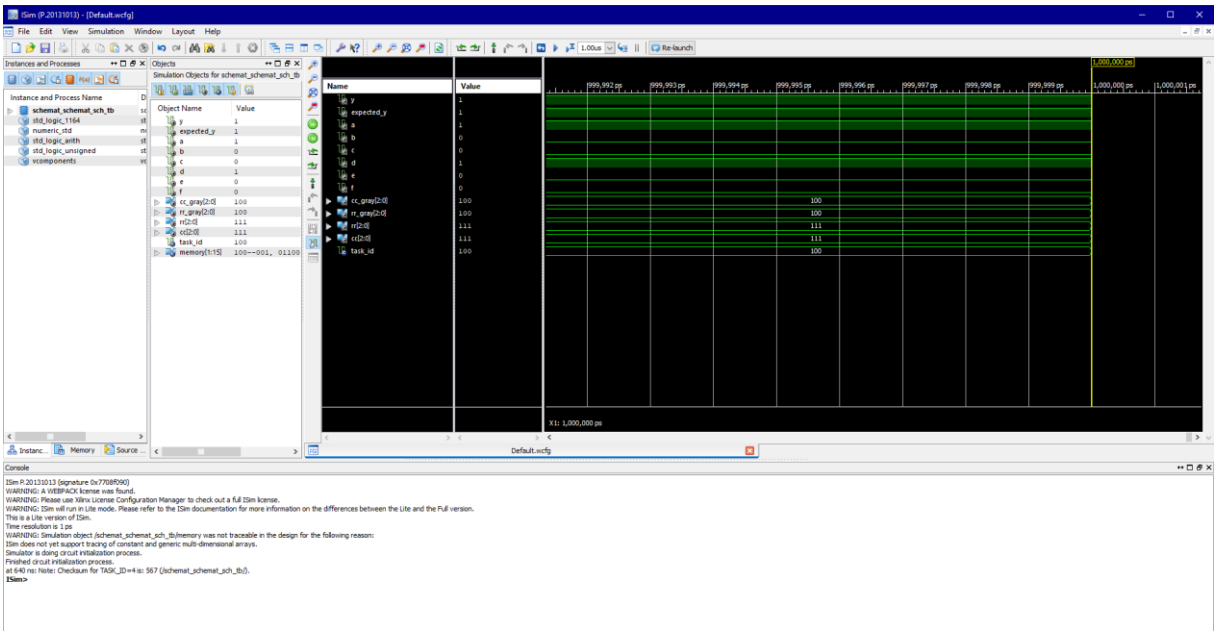


Any errors that could appear would do so in the “Console” tab, but as we can see, there are no errors visible. This means that the schematic works correctly.

- **2-bit multiplexer (MUX4)**

Below we can see a 2-bit multiplexer and multiple gates at its input. Each input fulfils what I have computed in the theoretical design and it was properly optimised so that there were no repeating gates. The most significant digit was ‘a’ so I put in in S1 input and least significant was ‘f’, so I put it in S0 input. This creates 4 situations: 00, 01, 10, 11. Depending on the result, it chose D0, D1, D2, D3 accordingly.

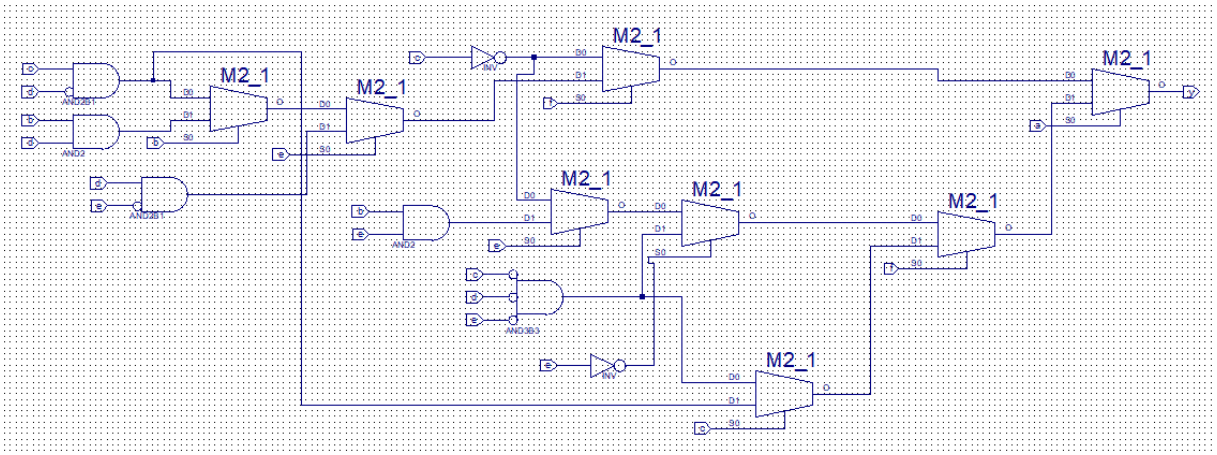




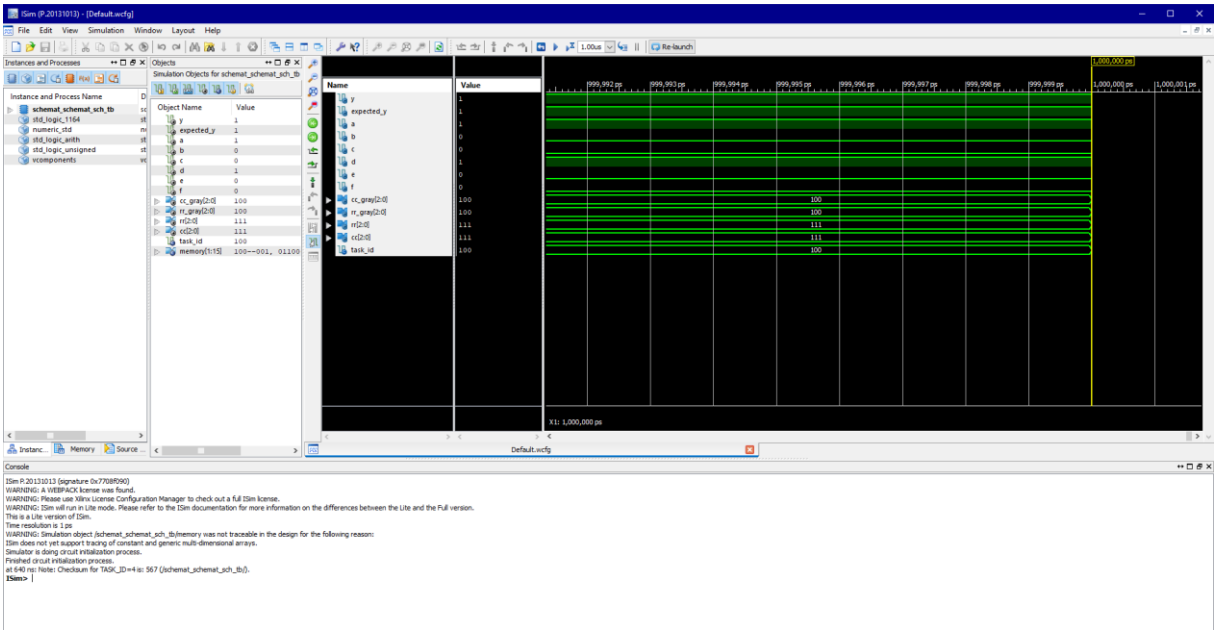
As we can see, there are once again no errors meaning all calculations are correct and this schematic works.

- **1-bit multiplexers (MUX2)**

This task was non-obligatory, but I still managed to do it. It required of me using multiple (8 MUX2 in total) multiplexers and connecting them wasn't as easy. What I did was to change all of the OR gates with two inputs into MUX2 and if it had 3 inputs, it required 2 1-bit multiplexers. I carefully analysed each input and deduced which one of the initial inputs (a, b, c, d, e, f) should be used as select. For example in the gate cd + bd, to change OR into a MUX2, I put cd into one input, bd into the other and by analysing them for a moment I noticed that if d = 0, output will always be 0. So now I had to choose either b or c as select. I chose b and now if b =1 it checks the gate with bd, if it's equal to 0, it checks the gate with cd. Because they share 'd' I can do that. And I did such analysis for every gate.
Below you can see the schematic of it:



And the test:



As we can see, there is not a single error meaning that everything works as it should.

## Conclusions

To conclude, a K-map can be easily transferred into a schematic, and it can be done in many different ways. Some are more optimal, some are less optimal, just like the one with the use of MUX2 was much bulkier than the other two.