Digital Circuits Theory - Laboratory						
Academic year	Laboratory exercises on	Mode of studies	Field of studies	Supervisor	Group	Section
2024/2025	Wednesday	SSI	Informatics	KP	1	1
	11:45 – 13:15					

Report from Exercise No 1

Performed on: 23.10.2024

Exercise Topic: Combinational Circuits

Performed by:

Piotr Copek
Zuzanna Micorek

Introduction

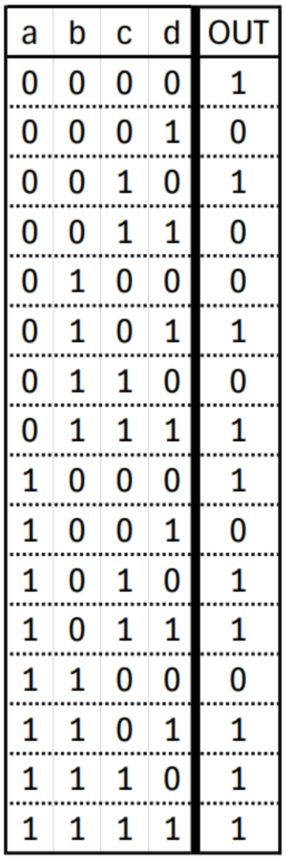
These circuit design tasks aimed to deepen our understanding of digital logic, Boolean algebra, and practical circuit implementation. We focused on:

- Creating and analyzing truth tables, Karnaugh maps, and Boolean expressions.
- Building and observing circuits in a lab environment to bridge theory and practice.
- Troubleshooting issues encountered during circuit construction.
- Utilizing TTL Technology.

Task 2

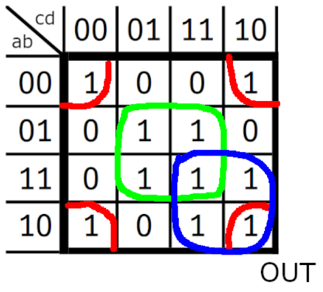
Design a circuit that adds two 2-bit binary numbers (given in the natural binary code). When the sum is a power of 3, the output should equal 0, and 1 otherwise.

- First, we enumerated all possible input combinations for two 2-bit binary numbers. We listed each possible binary combination for these bits in a truth table.
- Next, we examined the sum of the inputs for each row in the truth table to determine whether the sum is a power of 3 (1, 3). According to the task, if the sum is a power of 3, the output should be 0; otherwise, it should be 1, based on those conditions we assigned them properly in the truth table.



a	b	c	d	OUT
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

- Then we then transferred the values from the truth table to a Karnaugh map.



ab \ cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	1
10	1	0	1	1

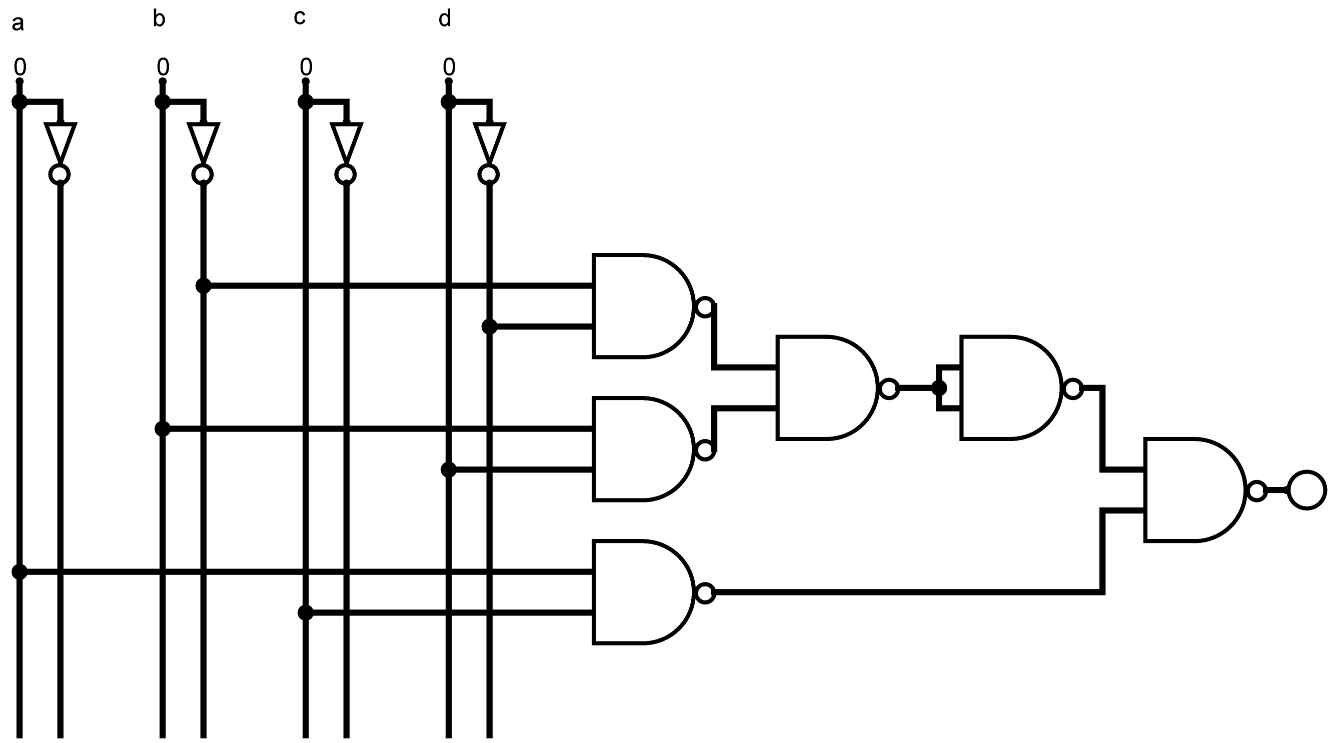
OUT

- Marking the biggest possible groups we obtained boolean expression from the K-map:

$$OUT = \overline{b} \cdot \overline{d} + b \cdot d + a \cdot c$$

- At the end of task-solving process we applied De Morgan's laws to transform the simplified Boolean expression into a form that could be implemented with NAND gates:

$$OUT = \overline{\overline{\overline{\overline{\overline{b} \cdot \overline{d}} \cdot \overline{b \cdot d}}} \cdot \overline{a \cdot c}}$$



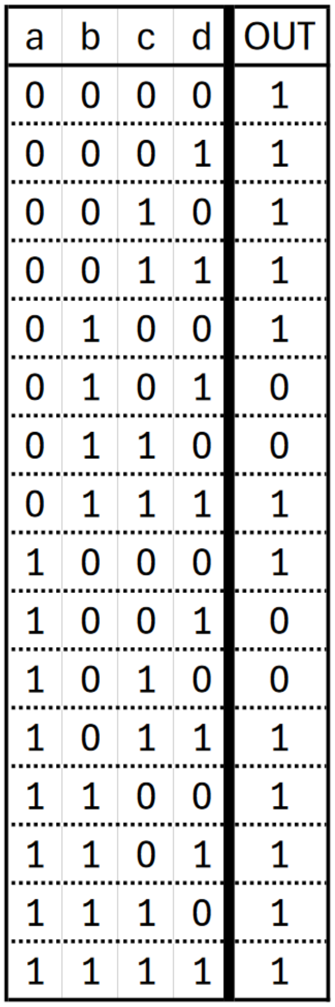
[Figure 1] - Implementation of task number 2 on NAND gates.

With this solution, we successfully designed a circuit that adds two 2-bit binary numbers and gives the correct output based on the given conditions. All gates were working properly, so we didn't encounter any difficulties during implementation of this task.

Task 4

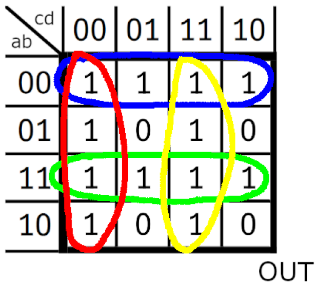
Design a circuit that multiplies two 2-bit binary numbers (given in the natural binary code). When the product is a power of 2, the output should equal 0, and 1 otherwise.

- We began by listing all possible input combinations for two 2-bit binary numbers.
- Then we examined each product to determine whether it was a power of 2 (1, 2, 4). According to the task, the output should be 0 if the product is a power of 2, and 1 otherwise, based on that we assigned proper outputs in the truth table.



a	b	c	d	OUT
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

- Then we transferred the values from the truth table to a Karnaugh map.



ab \ cd	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	1	1	1	1
10	1	0	1	0

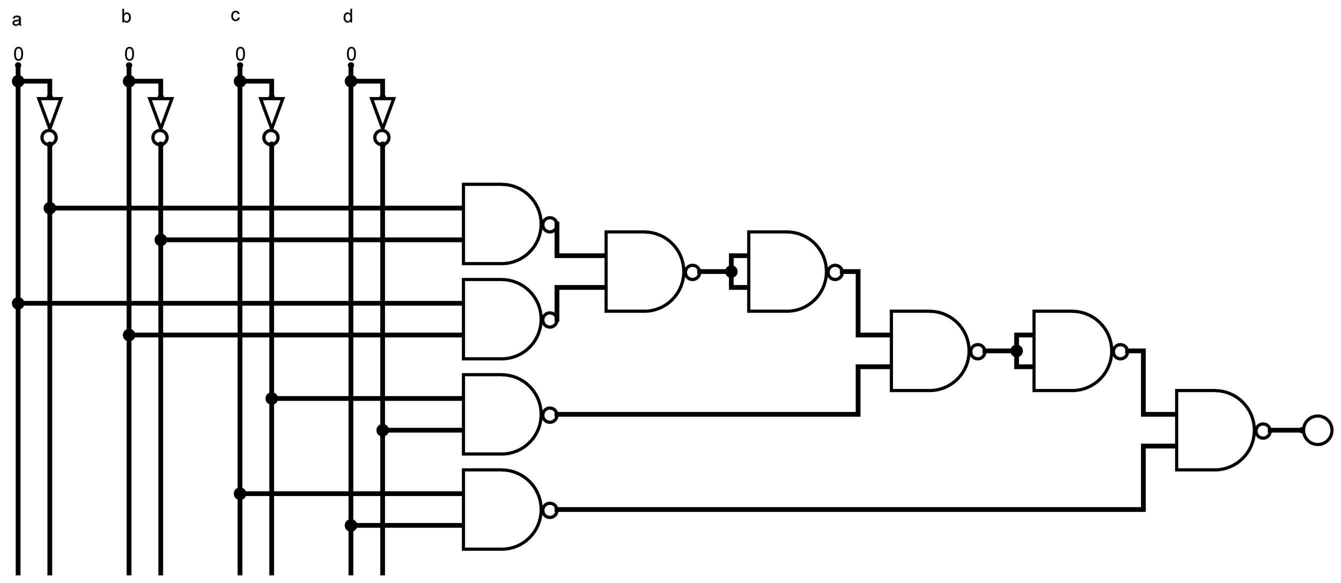
- We derived boolean expression from the K-map:

$$OUT = \overline{a} \cdot \overline{b} + a \cdot b + \overline{c} \cdot \overline{d} + c \cdot d$$

- After obtaining a Boolean expression, we applied De Morgan's laws to express it in terms of NAND gates:

$$OUT = \overline{\overline{\overline{\overline{\overline{\overline{\overline{a}\cdot\overline{b}}\cdot\overline{a\cdot b}}}\cdot\overline{\overline{c}\cdot\overline{d}}}}\cdot\overline{c\cdot d}}$$

- Due to the limited availability of NAND gates, we used 4-input NAND gates to perform negation. By connecting only one input to a 4-input NAND gate, it functioned as a NOT gate which is a feature of the TTL technology.



[Figure 2] - Implementation of task number 4 on NAND gates.

For better visualization all 4-input NAND gates were replaced with 2-input NAND gates with one of the inputs branched and connected to two inputs.

During the laboratories implementation, we encountered a problem where one of the 4-input NAND gates did not perform the negation function as expected. This issue required adjustments to ensure the correct operation of the circuit.

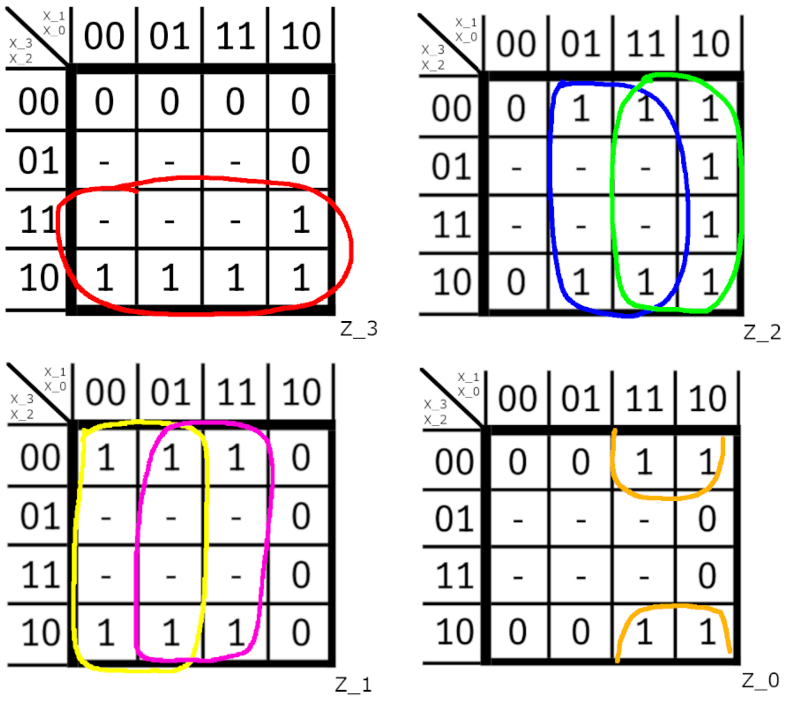
Task 6

Design 4-bit translator from Watts code into Gray+3 code.

- We began by listing all possible 4-bit combinations in Watts code. For each combination, we calculated the corresponding 4-bit Gray+3 code. We wrote these pairs in a truth table.

Watts				Gray+3			
x_3	x_2	x_1	X_0	Z_3	Z_2	Z_1	Z_0
0	0	0	0	0	0	1	0
0	0	0	1	0	1	1	0
0	0	1	1	0	1	1	1
0	0	1	0	0	1	0	1
0	1	1	0	0	1	0	0
1	1	1	0	1	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	1	1	1	1
1	0	0	1	1	1	1	0
1	0	0	0	1	0	1	0

- We transferred the values from the truth table to a Karnaugh map for each of the four output bits.



Z_3

X_3 X_2 \ X_1 X_0	00	01	11	10
00	0	0	0	0
01	-	-	-	0
11	-	-	-	1
10	1	1	1	1

Z_2

X_3 X_2 \ X_1 X_0	00	01	11	10
00	0	1	1	1
01	-	-	-	1
11	-	-	-	1
10	0	1	1	1

Z_1

X_3 X_2 \ X_1 X_0	00	01	11	10
00	1	1	1	0
01	-	-	-	0
11	-	-	-	0
10	1	1	1	0

Z_0

X_3 X_2 \ X_1 X_0	00	01	11	10
00	0	0	1	1
01	-	-	-	0
11	-	-	-	0
10	0	0	1	1

- Using the K-maps we obtained the Boolean expressions for each output bit.

$$Z_3 = X_3$$

$$Z_2 = X_0 + X_1$$

$$Z_1 = \overline{X_1} + X_0$$

$$Z_0 = \overline{X_2} \cdot X_1$$

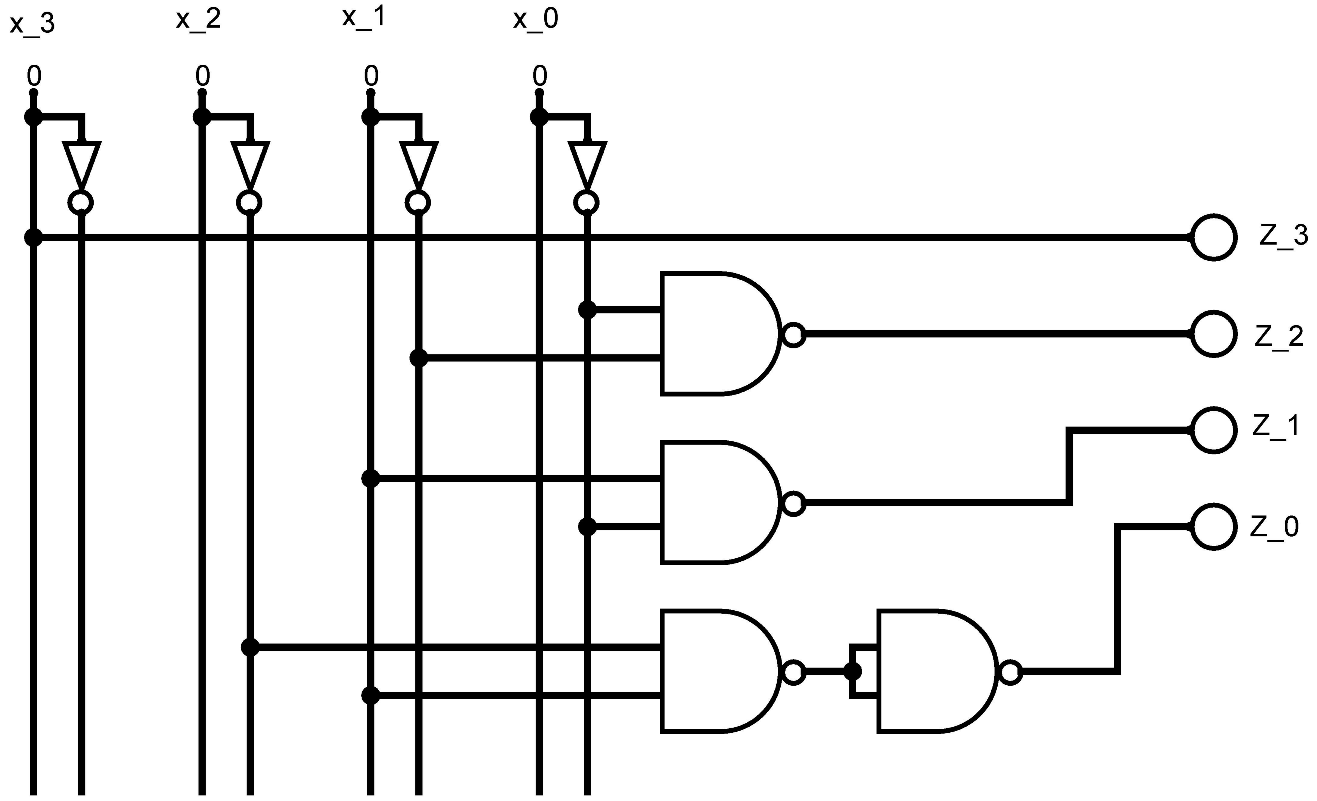
- Next for each output bit, we applied De Morgan's laws to express these functions using NAND gates.

$$Z_3 = X_3$$

$$Z_2 = \overline{\overline{X_0} \cdot \overline{X_1}}$$

$$Z_1 = \overline{X_1 \cdot \overline{X_0}}$$

$$Z_0 = \overline{\overline{\overline{X_2} \cdot X_1}}$$



[Figure 3] - Implementation of task number 6 on NAND gates.

This approach led to a final circuit design that satisfies the task requirements, translating a 4-bit Watts code into Gray+3 code using only NAND gates.

Final conclusions and observations

In these tasks, we designed and implemented circuits to achieve specific digital logic functions. We gained hands-on experience with creating truth tables, and using Karnaugh maps. We also translated theoretical designs into practical implementations using TTL technology and NAND gates. Through these exercises, we developed problem-solving skills and learned to troubleshoot real-world issues, such as detecting and dealing with malfunctioning components.