<table>
<tr><td colspan="3">SPRAWOZDANIE</td><td colspan="7">PROSZĘ PODAĆ NR GRUPY: 1</td></tr>
<tr><td rowspan="2">NAME</td><td rowspan="2">LAST NAME</td><td>Temat ćwiczenia zgodny z wykazem tematów:</td><td colspan="4">PONIŻEJ PROSZĘ PODAĆ TERMIN ZAJĘĆ:</td><td colspan="3">ROK: 2025 r.</td></tr>
<tr><td rowspan="3">LOGICAL FUNCTIONS III
Karnaugh Tables (Maps)
Implicants and Implicents</td><td>PN</td><td>WT</td><td>SR</td><td>CZ</td><td>PT</td><td>SB</td><td>ND</td></tr>
<tr><td rowspan="2">Volha</td><td rowspan="2">Hryshkevich</td><td colspan="7"></td></tr>
<tr><td colspan="5">GODZINA ROZPOCZĘCIA ZAJĘĆ:</td><td colspan="2">9 :45</td></tr>
<tr><td colspan="10">UWAGA !!! Wypełniamy tylko białe pola. W punkcie 1, proszę zakreślić odpowiednie pola i podać godzinę w której odbywają się zajęcia, zgodnie z planem zajęć.</td></tr>
</table>

Theoretical introduction (2500 characters):

Logical Functions III: Karnaugh Maps, Implicants, and Implicates

Boolean algebra forms the mathematical foundation of digital logic design. Each logical function describes the dependence of an output variable Yon one or more input variables A, B, C, D, ..., where each variable can take only two values: 0 (false) or 1 (true). The primary goal of Boolean function analysis is simplification — finding an equivalent expression that uses the fewest possible logical operations. This reduction directly translates into simpler, faster, and more cost-effective digital circuits.

1. Canonical Forms of Boolean Functions

Any Boolean function can be represented in two canonical forms:

1. **Sum of Products (SOP)**, also called the **Disjunctive Normal Form (DNF)**:

$$Y = \sum m(i_1, i_2, i_3, \dots)$$

Each term $m(i)$is a **minterm**, a conjunction (AND) of all variables or their negations.

Example:

$$Y(A, B, C) = A\bar{B}C + AB\bar{C} + \bar{A}BC$$

2. **Product of Sums (POS)**, also called the **Conjunctive Normal Form (CNF)**:

$$Y = \prod M(j_1, j_2, j_3, \dots)$$

Each term $M(j)$is a **maxterm**, a disjunction (OR) of all variables or their complements.

Both representations are complete and can be transformed into each other using De Morgan's laws and distributive properties.

2. The Karnaugh Map

The **Karnaugh map (K-map)** provides a graphical way to simplify Boolean expressions. It is structured as a grid where each cell corresponds to a unique binary combination of the input variables. The order of cells follows **Gray code**, ensuring that adjacent cells differ by only one variable.

For example, the K-map for four variables A, B, C, Dcontains 16 cells:

AB\CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Each cell holds either 1 or 0, depending on whether the function is true or false for that particular combination.

The main advantage of the K-map is that **adjacent 1s can be grouped** to remove variables that differ within the group. Each group must contain a number of cells equal to 2^n, where $n$is an integer (1, 2, 4, 8, …).

3. Implicants and Prime Implicants

An **implicant** is any product term that corresponds to one or more cells where the function has the value 1. If an implicant covers a single 1-cell, it is called a **minterm**.

If it covers several adjacent cells, it represents a simplified product term.

For example, if two adjacent cells correspond to:

$$A\bar{B}C\bar{D} \text{ and } A\bar{B}CD,$$

then their combination eliminates D:

$$A\bar{B}C$$

This simplified expression is an **implicant**.

A **prime implicant** is an implicant that cannot be combined further with others to cover more 1s without including any 0s.

In the K-map, prime implicants correspond to the **largest possible rectangular groupings** of 1s.

There is also a subset known as **essential prime implicants**, which are prime implicants covering at least one minterm not covered by any other implicant. These must be included in the final simplified expression.

4. Implicates and Prime Implicates

The dual concept applies to 0s in the function.

An **implicate** is a sum term that includes all combinations for which the function equals 0.

Implicates are relevant when simplifying the function in **Product of Sums (POS)** form.

Grouping adjacent 0s in the K-map produces implicates.

A **prime implicate** is the largest possible group of 0s that can be combined without including any 1s.

For instance, if two 0-cells correspond to:

$$(A + \bar{B} + C + D)\text{and } (A + \bar{B} + \bar{C} + D),$$

they can be combined into:

$$(A + \bar{B} + D)$$

5. Simplification Example

Consider the function $Y(A, B, C, D)$with the following minterms where $Y = 1$:

$$m_0, m_2, m_3, m_4, m_8, m_{10}, m_{12}, m_{15}$$

The corresponding SOP form is:

$$Y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + ABCD$$

By grouping adjacent 1s in the Karnaugh map, several variables can be eliminated, yielding a simplified expression such as:

$$Y = \bar{B}\bar{D} + AC\bar{D} + \bar{A}C$$

This minimized form uses fewer terms and logical operations, demonstrating the efficiency of graphical simplification.

6. Practical Notes

- Groups must always be rectangular and contain 2^ncells.
- Each 1 must be covered by at least one group.
- Overlapping groups are allowed and often necessary.
- “Don’t care” conditions (marked as X) can be included in groups to further simplify the result.
- For functions with more than four variables, multi-dimensional maps or algorithmic approaches like **Quine–McCluskey** are used.

Exercise 1 Complete both tables.

TWO-DIGIT NATURAL CODE

SYSTEM Decimal	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1

6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

GRAY'S Code

SYSTEM Decimal	A	B	C	D
0	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**1**
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1

14	1	0	0	1
15	1	0	0	0

EXERCISE 2

USING KARNAUGH MAPS, DESIGN A DECODER.

THE BINARY CODES A-MSB AND C-LSB APPEAR ON THE THREE INPUTS. THE OUTPUT SHOULD SHOW ONES FOR THE FUNCTION.

FUNCTION PERFORMED

0,2,3,6 =1

1,4,5,7 = 0

2^2 2^1 2^0

	A	B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Implicants

C / AB	0	1
00	1	0
01	1	1
11	1	0
10	0	0

IMPLICIT

C / AB	0	1
00	1	0
01	1	1
11	1	0
10	0	0

PLEASE RECORD THE FUNCTIONS:

Implicants without reduction:

$$Y = (\bar{A} * \bar{B} * \bar{C}) + (\bar{A} * B * \bar{C}) + (\bar{A} * B * C) + (A * B * \bar{C})$$

Implicits without reduction:

$$Y = (A + B + \overline{C}) * (\overline{A} + \overline{B} + \overline{C}) * (\overline{A} + B + C) * (\overline{A} + B + \overline{C})$$

Implicants after reduction:

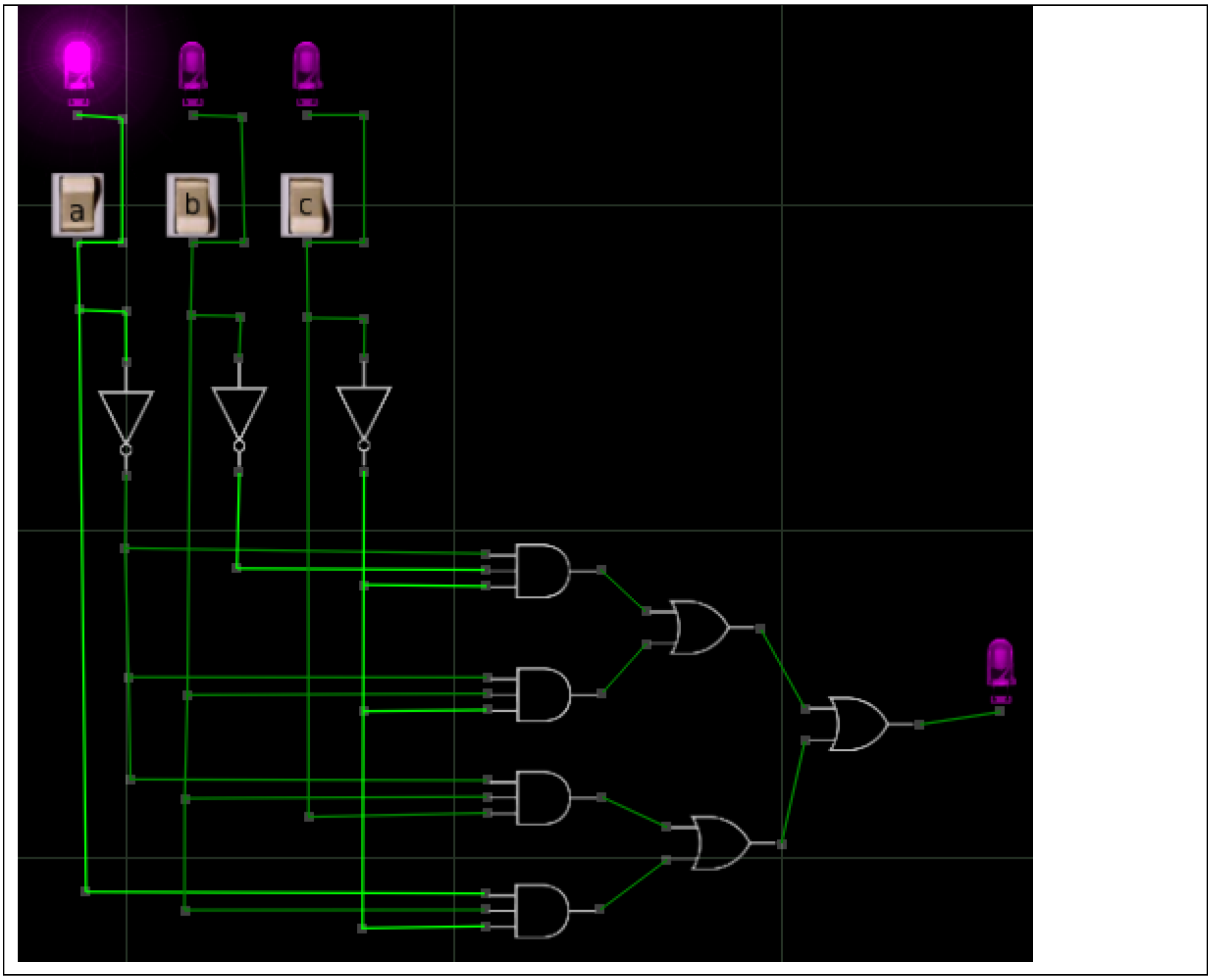
$$Y = \overline{A}\overline{C} + B\overline{C} + \overline{A}B$$

Implicits after reduction:

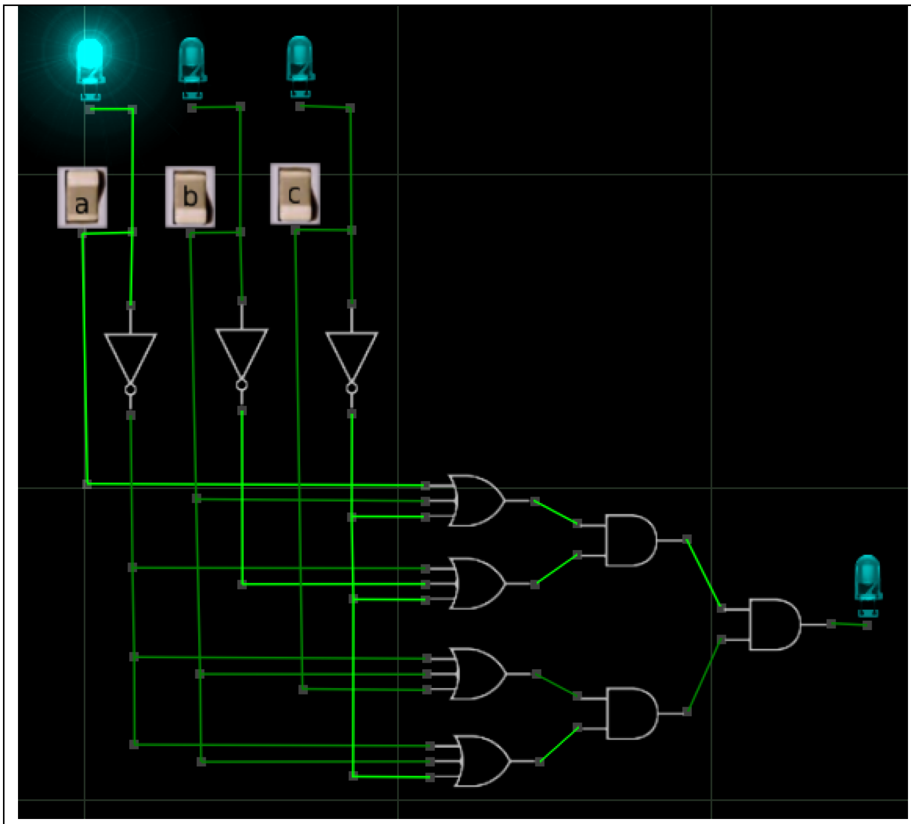
$$Y = (\overline{C} + B) * (\overline{C} + \overline{A}) * (\overline{A} + B)$$

PLEASE PERFORM SIMULATIONS FOR ALL CASES IN THE ATANUA PROGRAM. PLEASE POST SCREENSHOTS BELOW.

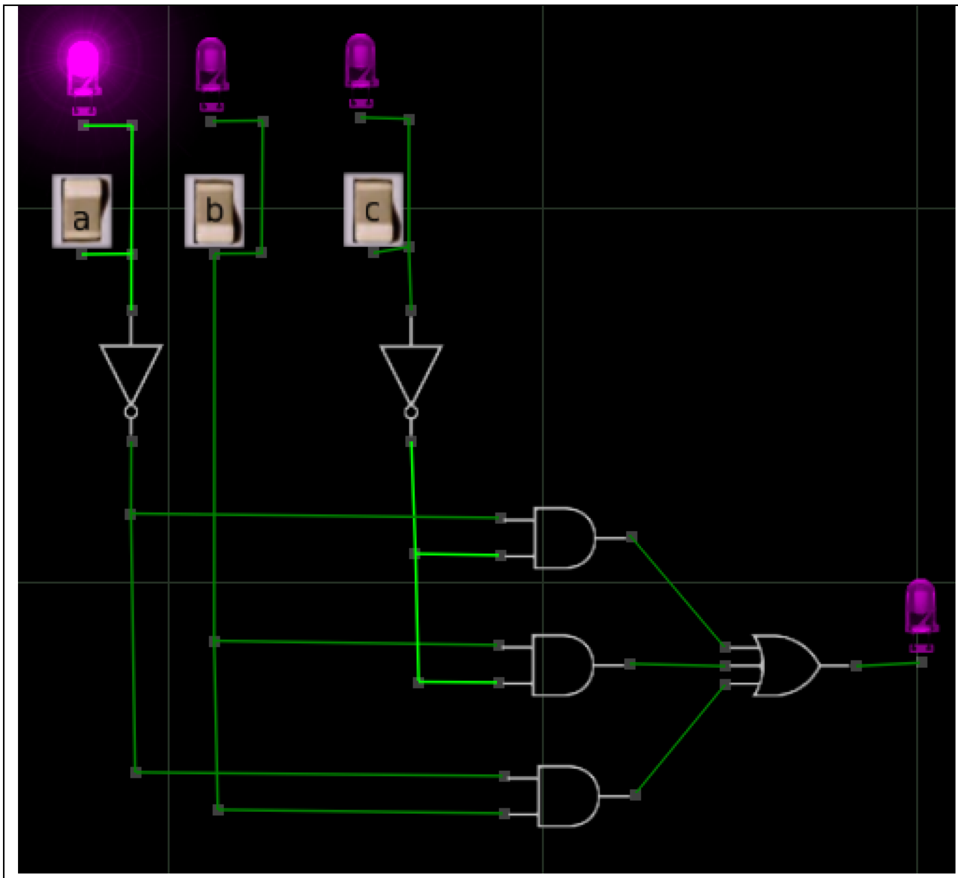
IMPLICANTS (without reduction)



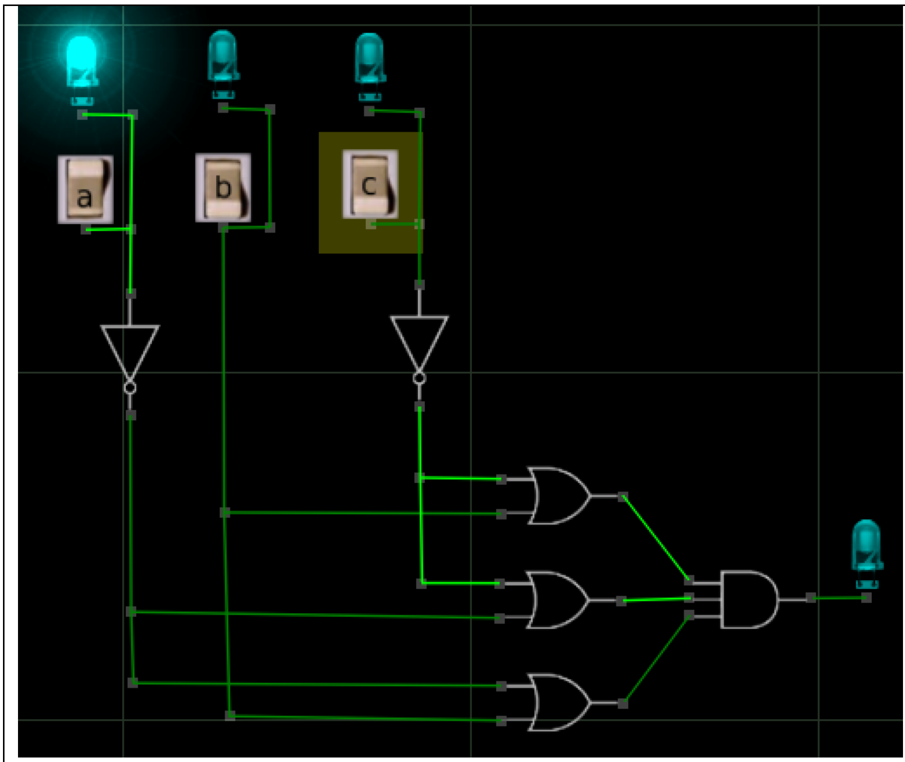
IMPLICENTS (without reduction)



PLEASE POST SCREENSHOTS BELOW. IMPLICANTS (after reduction)



IMPLICENTS (after reduction)



EXERCISE 3

Please complete the NATURAL binary code.

SYSTEM Decimal	A	B	C	D		Y
0	0	0	0	0	→	1
1	0	0	0	1	→	0
2	0	0	1	0	→	1
3	0	0	1	1	→	1
4	0	1	0	0	→	1
5	0	1	0	1	→	0
6	0	1	1	0	→	0
7	0	1	1	1	→	0
8	1	0	0	0	→	0

9	1	0	0	1	→	**0**
10	1	0	1	0	→	**1**
11	1	0	1	1	→	**0**
12	1	1	0	0	→	**1**
13	1	1	0	1	→	**0**
14	1	1	1	0	→	**0**
15	1	1	1	1	→	**1**

Please perform reduction using Karaguth tables with implicants and implicands.

IMPLICANTS EXERCISE 3A

CD / AB	00	01	11	10
00	1	0	1	1
01	1	0	0	0
11	1	0	1	0
10	0	0	0	1

IMPLICENTS EXERCISE 3B

CD / AB	00	01	11	10
00	1	0	1	1
01	1	0	0	0
11	1	0	1	0
10	0	0	0	1

PLEASE RECORD THE FUNCTIONS:

Implicants (NO REDUCTION):

$$Y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} + ABCD + A\bar{B}C\bar{D}$$

Implicents (NO REDUCTION):

$$Y = (A + B + C + \bar{D}) * (A + \bar{B} + C + \bar{D}) * (A + \bar{B} + \bar{C} + \bar{D}) * (A + \bar{B} + \bar{C} + D) * (\bar{A} + \bar{B} + C + \bar{D}) * (\bar{A} + \bar{B} + \bar{C} + D) * (\bar{A} + B + C + D) * (\bar{A} + B + C + \bar{D}) * (\bar{A} + B + \bar{C} + \bar{D})$$

Implicants (AFTER REDUCTION):

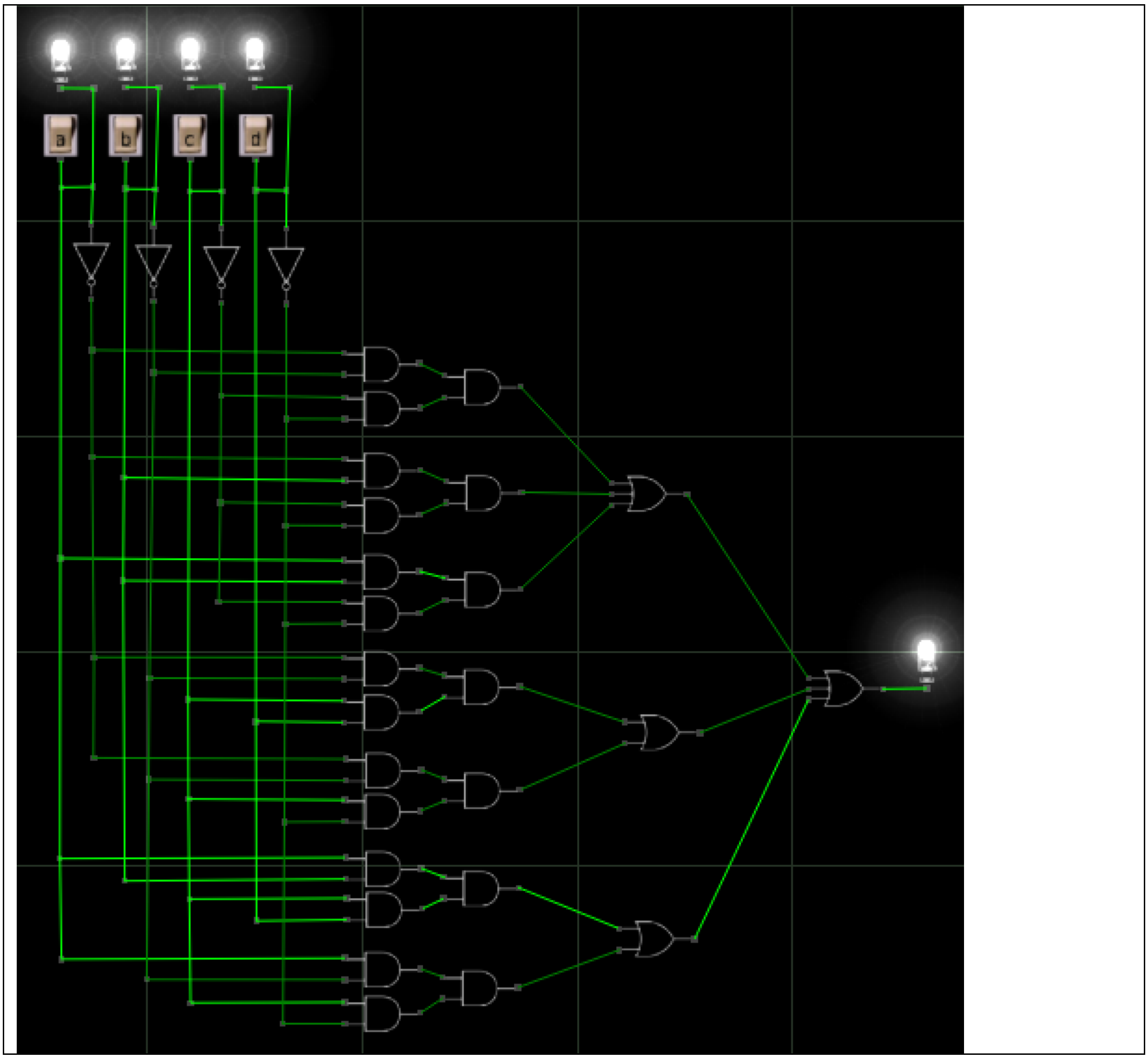
$$Y = ABCD + \bar{A}\bar{C}\bar{D} + B\bar{C}\bar{D} + \bar{A}\bar{B}C + \bar{B}C\bar{D}$$

Implicents (AFTER REDUCTION):

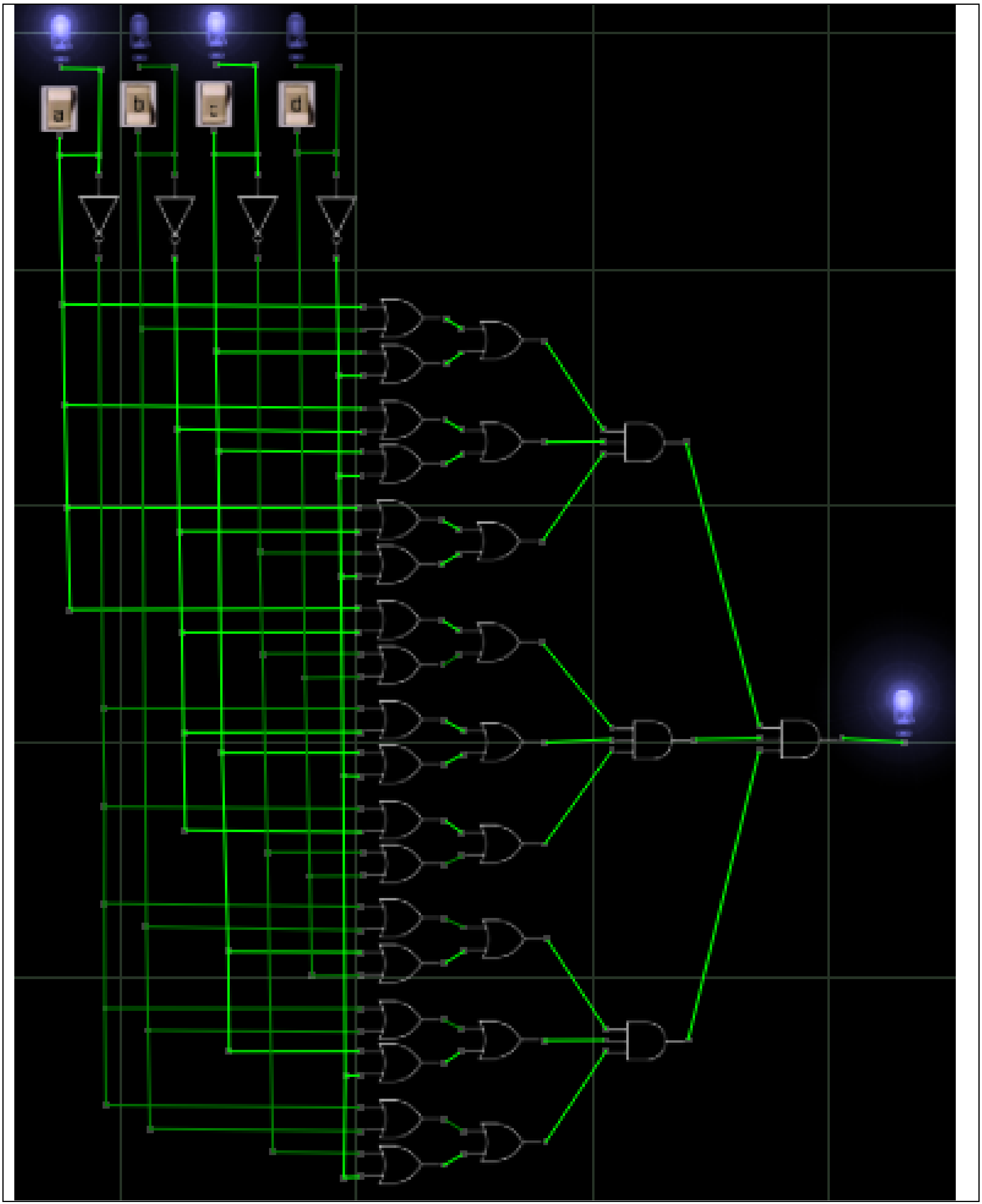
$$Y = (C + \bar{D}) * (A + \bar{B} + \bar{C}) * (\bar{B} + \bar{C} + D) * (\bar{A} + B + C) * (\bar{A} + B + \bar{D})$$

Please complete all exercises based on the integrated circuits available in the ATANUA program.

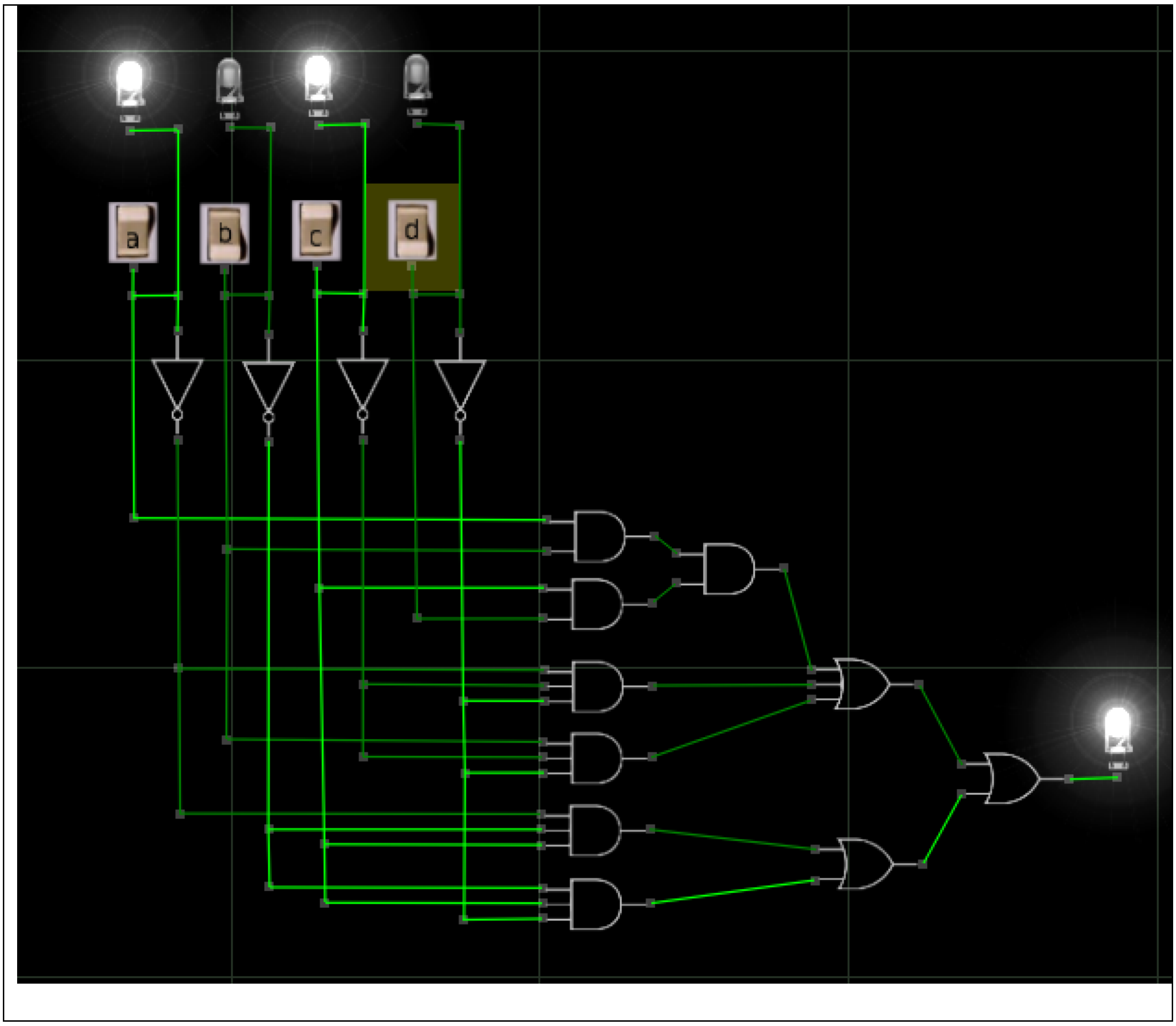
EXERCISE 2A SIMULATION OF IMPLICANTS (WITHOUT REDUCTION)



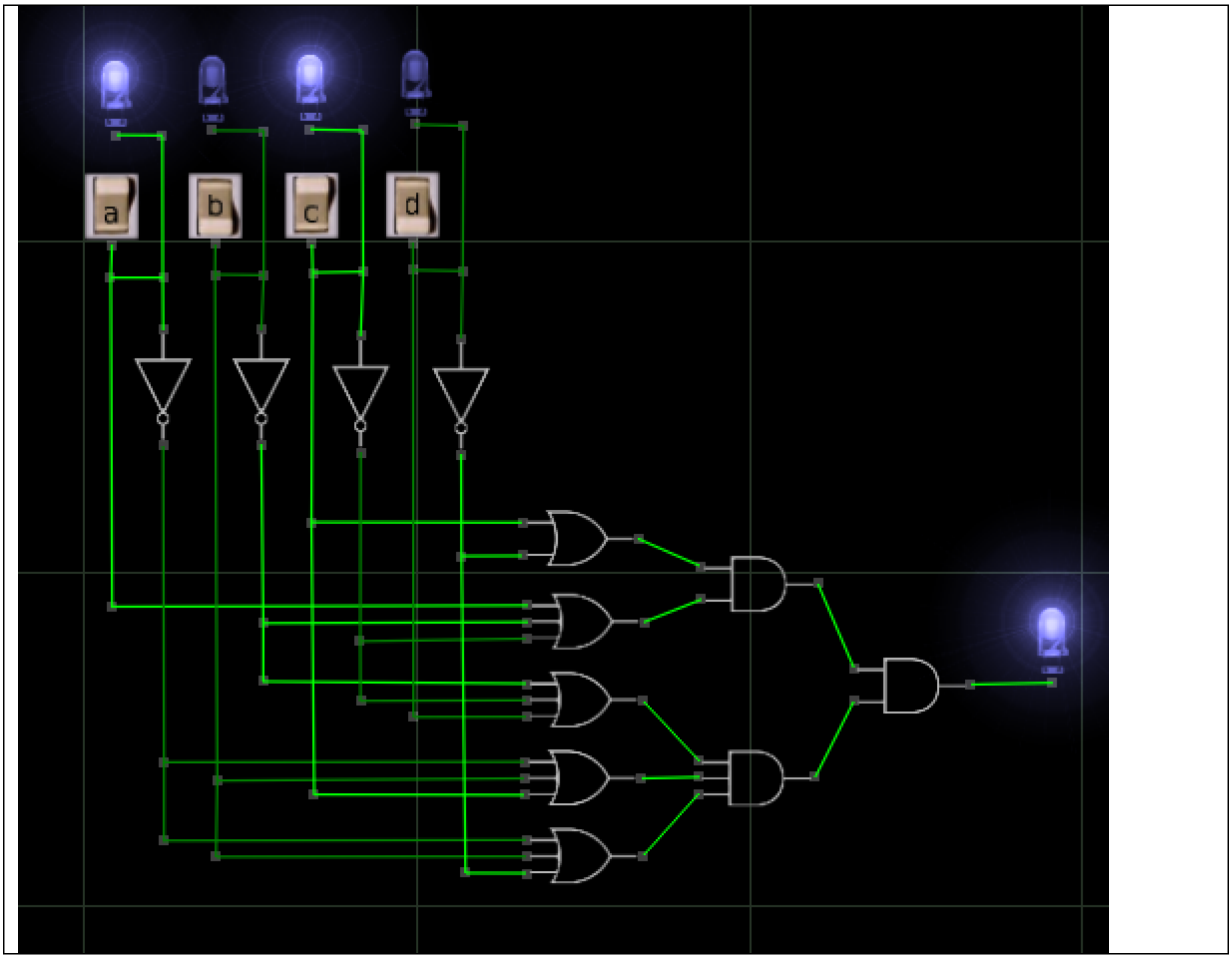
EXERCISE 2B IMPLICENT SIMULATION (WITHOUT REDUCTION)



EXERCISE 2A SIMULATION OF IMPLICANTS (AFTER REDUCTION)



EXERCISE 2B IMPLICENT SIMULATION (AFTER REDUCTION)



Conclusions (Write which reduction is more beneficial where the fewest goals were used)
From exercise 2 there is no difference in reduction of the implicents and reduction of the implicants.
From exercise 2b Reduction of the implicents is more beneficial as it uses a little bit smaller number of operations then implicants.