REPORT			PLEASE ENTER GROUP NUMBER: 1						
NAME	**SURNAME**	**Temat ćwiczenia zgodny z wykazem tematów:**	**PLEASE ENTER THE DATE OF THE CLASS BELOW:**					**ROK:**	
Volha	Hryshkevich	**LOGIC FUNCTIONS IV BINARY CODE DECODER TO SEVEN-SEGMENT**	**PN**	**WT**	**SR**	**CZ**	**PT**	**SB**	**ND**
			START TIME OF CLASSES:					**9 : 45**	

UWAGA !!! Wypełniamy tylko białe pola. W **punkcie 1**, proszę zakreślić odpowiednie pola i podać godzinę w której odbywają się zajęcia, zgodnie z planem zajęć.

Theoretical introduction: (2,500 characters)

In digital electronics, complex systems are built from simpler logic functions, each implemented using basic logic gates. Atanua, a real-time logic simulator, allows students and engineers to model these functions visually and interactively. One of the most common practical applications of logic functions is the binary-coded decimal (BCD) to seven-segment decoder, used for driving numeric displays. To reduce the number of gates needed in such circuits, techniques like Karnaugh maps (K-maps) are applied to simplify Boolean expressions while preserving the functional behavior of the circuit.

1. Logic Functions in Digital Circuitry

Logic functions describe how outputs depend on inputs in a digital system. Built from Boolean algebra, they are implemented using gates such as:

- **AND**: Output is 1 only when all inputs are 1.
- **OR**: Output is 1 when at least one input is 1.
- **NOT**: Inverts the input.
- **NAND, NOR, XOR, XNOR**: Variations that produce complex behavior.

In Atanua, these gates are arranged graphically, allowing immediate visualization of signal flow. Students can observe how the output changes dynamically as they toggle switches or alter input signals, making abstract Boolean concepts easier to understand.

2. Binary Code and the Purpose of Decoding

Digital systems represent numerical values using binary encoding. For example, a 4-bit binary number can represent 0–15. However, humans do not interpret binary values directly, so electronic devices often convert binary codes into a readable visual form. A common solution is the **seven-segment display**, which consists of seven individually controllable LED bars labeled A through G.

Each digit (0–9) requires a distinct combination of segments to be illuminated. A **decoder** translates binary-coded input into the correct pattern of active segments. For instance:

- Digit **0** → segments A, B, C, D, E, F
- Digit **1** → segments B, C
- Digit **7** → segments A, B, C
- Digit **8** → all segments

The decoder ensures that for any valid binary input, the correct display pattern is produced.

3. Binary-to-Seven-Segment Decoder Logic

A typical seven-segment decoder takes **four binary inputs** (B3, B2, B1, B0) and generates **seven outputs** (A through G). Each output is a Boolean function of the inputs. For example, the Boolean expression for segment A (before simplification) might look like:

$A = (\neg B3 \wedge \neg B2 \wedge \neg B1 \wedge B0)$

∨ (¬B3 ∧ B2 ∧ ¬B1 ∧ ¬B0)
∨ (B3 ∧ ¬B2 ∧ B1 ∧ B0)
∨ ...
While these expressions are accurate, they are usually **not minimal**. Using too many gates leads to inefficient circuits, increased propagation delays, and unnecessary complexity. This is where **Karnaugh maps** become essential.

4. Karnaugh Maps (K-maps) for Logic Reduction
A **Karnaugh map** is a graphical tool used to simplify Boolean expressions by grouping adjacent terms in a truth table. It works by visually identifying patterns that can be reduced through Boolean algebra rules.
Why K-maps Are Used
- Reduce the number of logic gates
- Minimize circuit complexity
- Produce faster and more reliable circuits
- Make manual simplification easier than algebraic methods alone

How a K-map Works
For a four-input system (common in seven-segment decoders), a **4-variable K-map** is used, arranged as a 4×4 grid. Each cell represents one combination from the truth table. Adjacent cells differ by only one bit, following **Gray code** ordering.
Simplification is done by grouping cells containing 1's (minterms):
- Groups must be powers of two: 1, 2, 4, 8, or 16 cells
- Groups should be as large as possible
- Wraparound is allowed at map edges

The result is a minimized Boolean expression that may require far fewer gates.

5. Applying K-maps to Seven-Segment Decoder Design
Each segment (A through G) has its own Boolean function, so **seven separate K-maps** are typically produced. Using K-maps:
1. List output values (1 = segment ON, 0 = segment OFF) for digits 0–9.
2. Mark combinations 10–15 as **don't-care (X)**, since these BCD inputs are invalid.
3. Place 1's and X's in the K-map for that segment.
4. Form the largest possible groups to simplify the equation.
5. Convert grouped terms back into minimized Boolean expressions.

This dramatically reduces gate counts. For example, segment A might simplify from an expression containing 7 minterms to one containing only 3 or 4 simplified terms.

6. Implementing the Reduced Logic in Atanua
Once simplified expressions are obtained:
- Build circuits using AND, OR, NOT gates
- Connect each reduced output function to the corresponding segment input
- Feed the 4-bit BCD input from switches or a counter
- Observe real-time updating of the seven-segment display

Atanua's real-time simulation helps verify that the K-map reductions were done correctly. If a segment lights incorrectly, users can trace signals through probes and fix errors more intuitively than on paper.

7. Educational Importance
Combining logic functions, decoders, and K-map simplification in Atanua teaches:
- Boolean algebra

- Truth tables and minterm analysis
- Logic optimization
- Practical display interface design
- Error-checking through simulation

This forms a strong foundation for later topics such as multiplexed display systems, microcontroller interfacing, and programmable logic devices.

Task 1

The task is to design a decoder for your own index number and simulate its operation in the ATANUA program. Regardless of the range of numbers in the index number, a four-bit decoder should be designed.
Decoder for digits appearing in your album number (index). If a given digit appears at the output, the LED should light up. If it does not appear, the LED at the output should not light up. At the input, you encode the digits in binary code. The order of the digits in the index number does not matter.
Next, reduce the Karanguh tables and present two designs for the IMPLICANTS and IMPLICENTS methods. The designs should be in the form of screenshots from the Atanua program and two files with ready-made decoders. Figure 1 shows a graphical representation of the decoder's operation.
Before doing the exercise, check which digits appear in your index number (album)
This number appears in your email.

Your project based on logic functions (do not use chips)

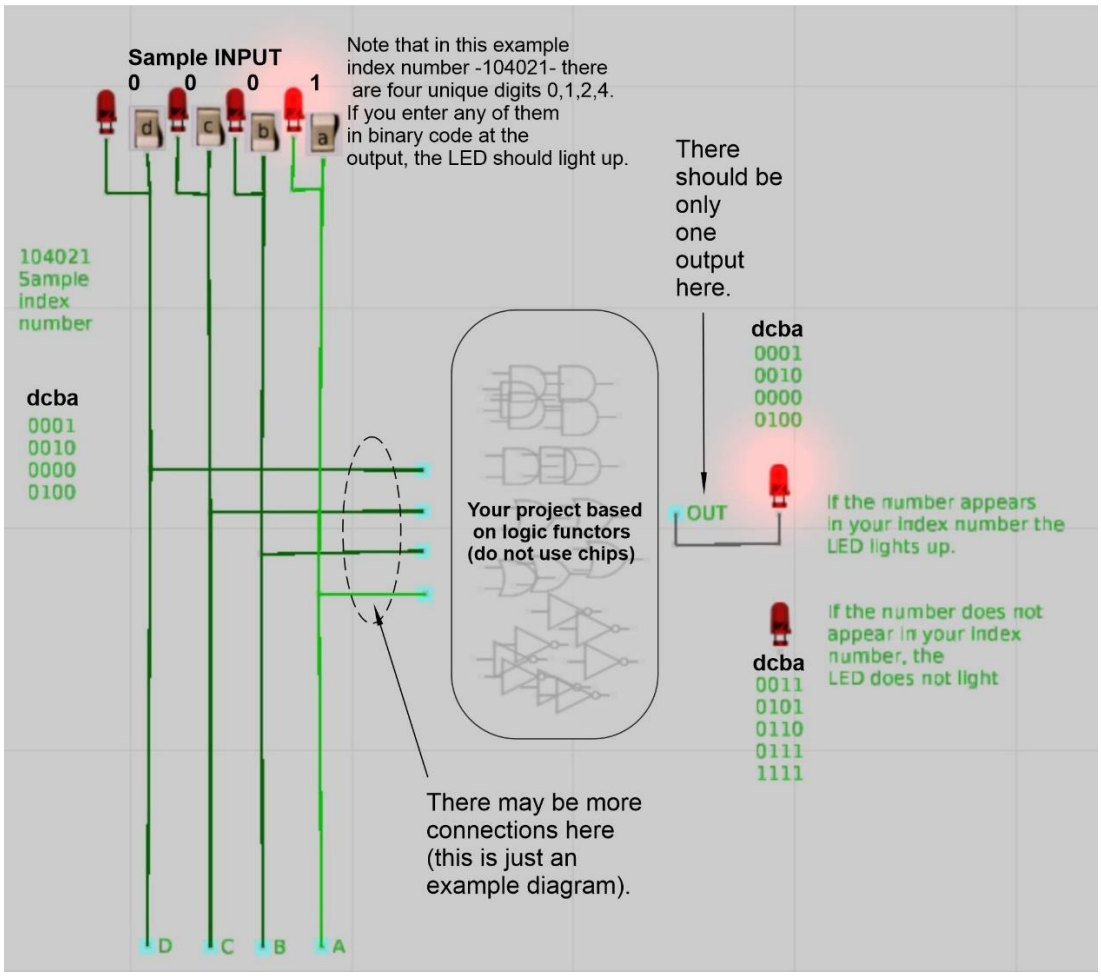


Fig. 1 Implementation of a decoder for numbers contained in the index number and those that are not contained in it.

When completed correctly, this part of the assignment will contain three files :
This completed raport
Six simulation files from the Atanua program.

Index number = 237714

D	C	B	A	Natural	Right number
0	0	0	0	0	-
0	0	0	1	1	+
0	0	1	0	2	+
0	0	1	1	3	+
0	1	0	0	4	+
0	1	0	1	5	-
0	1	1	0	6	-
0	1	1	1	7	+
1	0	0	0	8	-
1	0	0	1	9	-
1	0	1	0	10	-
1	0	1	1	11	-
1	1	0	0	12	-
1	1	0	1	13	-
1	1	1	0	14	-
1	1	1	1	15	-

Simulation without reduction

Implants

DC \ BA	00	01	11	10
00	0	1	1	1
01	1	0	1	0
11	0	0	0	0
10	0	0	0	0

Please write a function

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

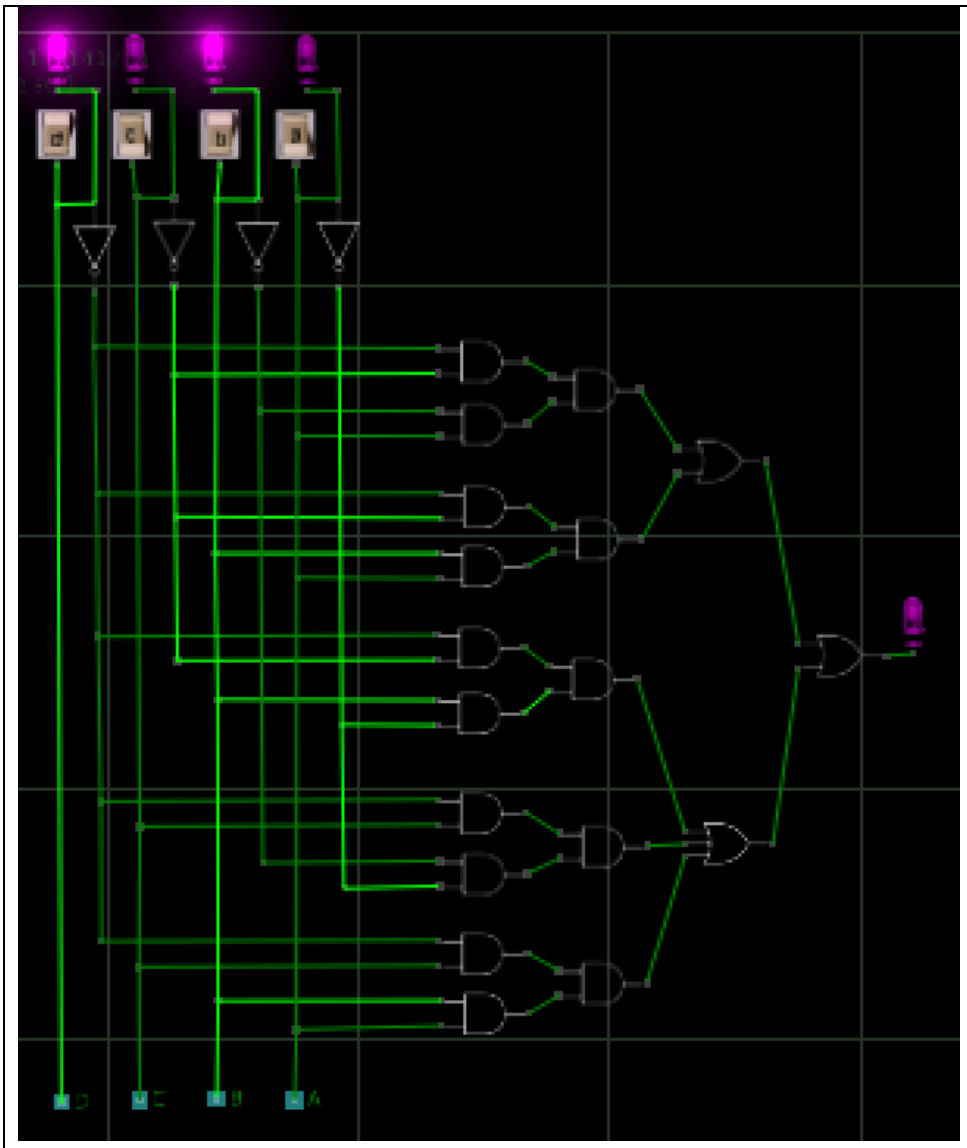
Implicents

DC \ BA	00	01	11	10
00	0	1	1	1
01	1	0	1	0
11	0	0	0	0
10	0	0	0	0

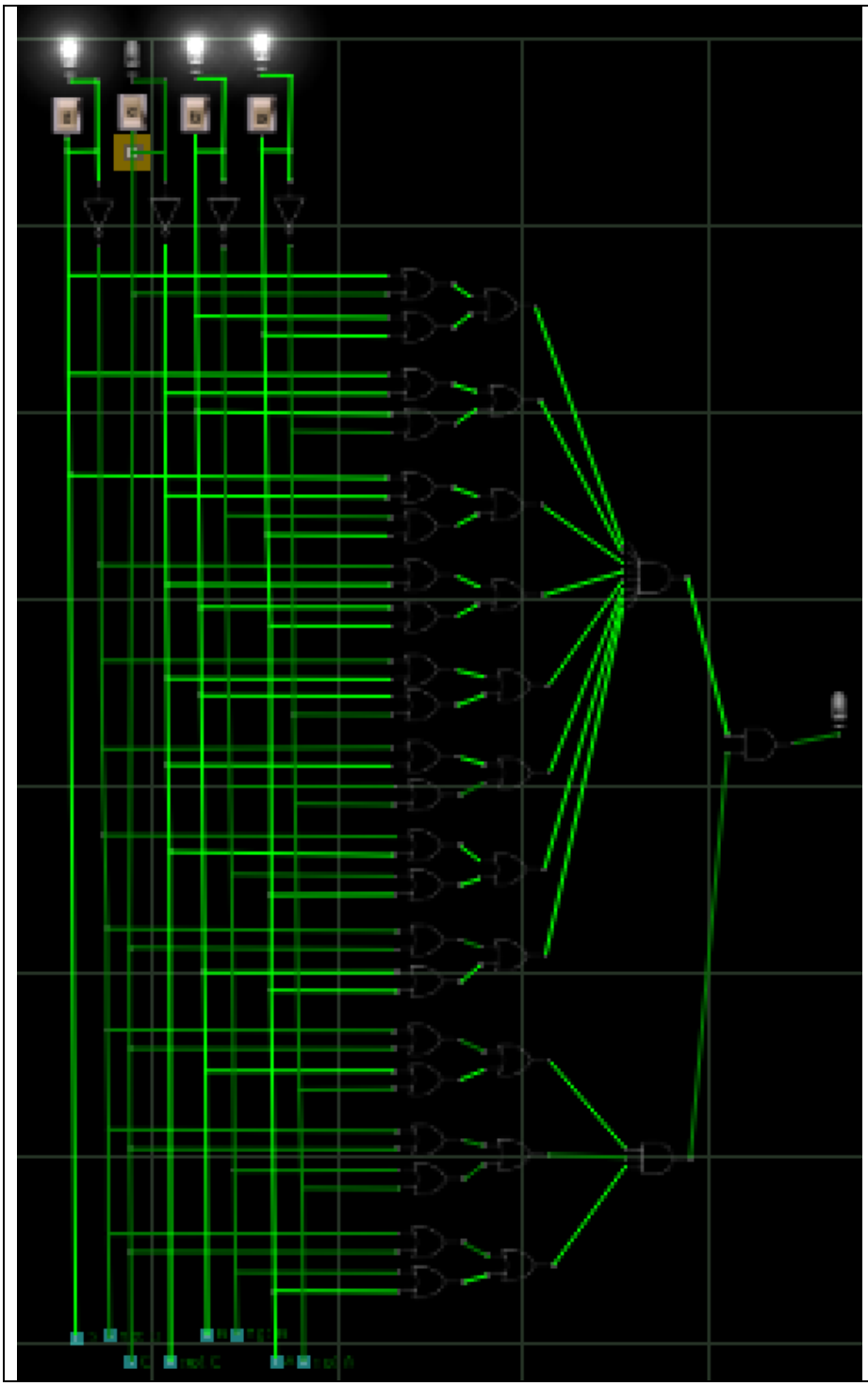
Please write the function.

$$\begin{aligned} Y = {} & (D + C + B + A) * (D + \bar{C} + B + \bar{A}) \\ & * (D + \bar{C} + \bar{B} + A) * (\bar{D} + \bar{C} + B + A) \\ & * (\bar{D} + \bar{C} + B + \bar{A}) * (\bar{D} + \bar{C} + \bar{B} + \bar{A}) \\ & * (\bar{D} + \bar{C} + \bar{B} + A) * (\bar{D} + C + B + A) \\ & * (\bar{D} + C + B + \bar{A}) * (\bar{D} + C + \bar{B} + \bar{A}) \\ & * (\bar{D} + C + \bar{B} + A) \end{aligned}$$

Screenshot of simulation for implicants (use logic functions, not chips)



Screenshot of simulation for implicents (use logic functions, not chips)



Perform reduction for implicants

DC \ BA	00	01	11	10
00	0	1	1	1
01	1	0	1	0
11	0	0	0	0
10	0	0	0	0

Please write a function

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

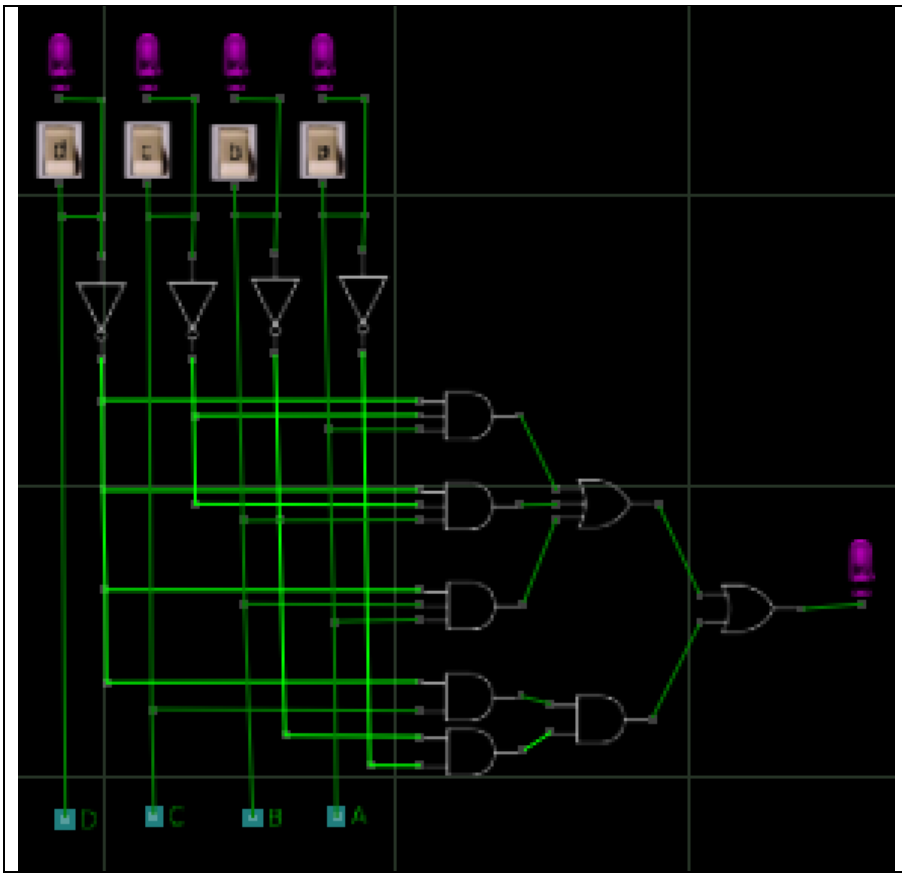
Reduction for implicants

DC \ BA	00	01	11	10
00	0	1	1	1
01	1	0	1	0
11	0	0	0	0
10	0	0	0	0

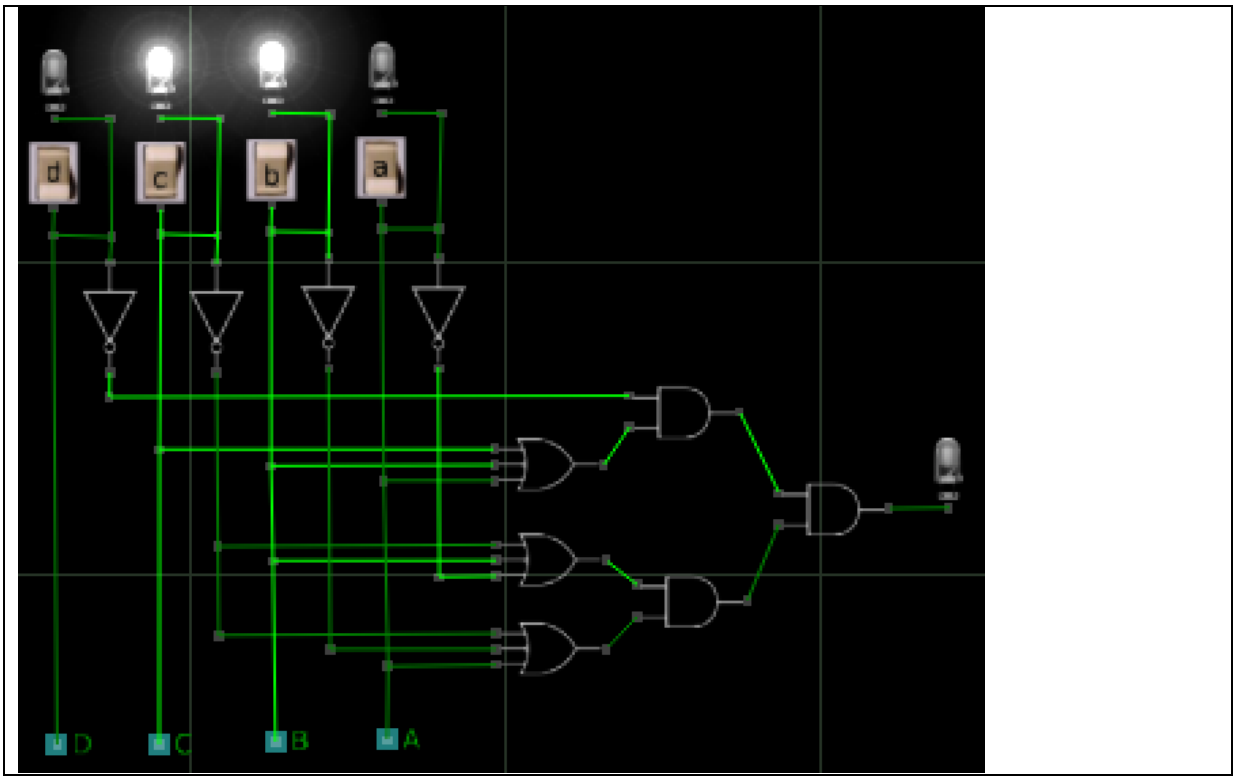
Please write the function.

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

Screenshot of simulation for implicants



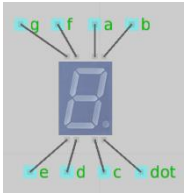
Screenshot of simulation for implicents

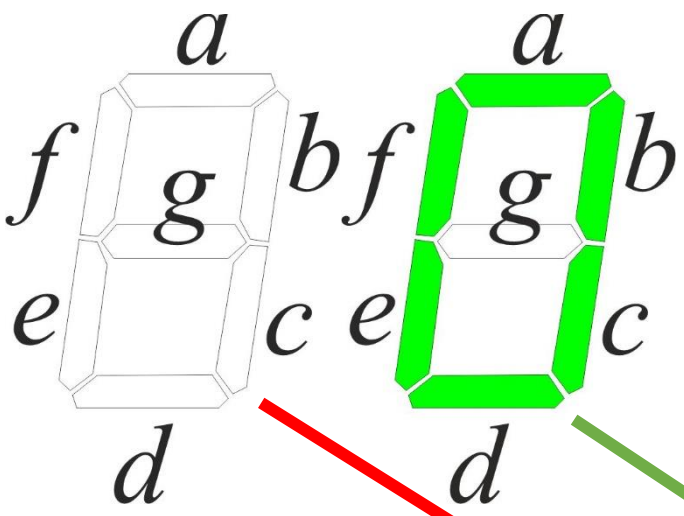


PLEASE REMEMBER TO ATTACH THE SIMULATION FILES

TASK 2 BUILD A NATURAL BINARY CODE DECODER from 0 to 10 FOR THE CODE DISPLAYED ON THE SEVEN-SEGMENT DISPLAY.

Include your index number. If a number does not appear in your index number, the (g) segment should light up on the right display, i.e., a dash (–) should appear, and the left display should be turned off. The exception is the number 10, which does not appear in your index number but should be displayed.

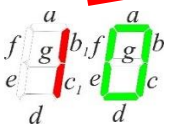




Binary code			
D	C	B	A
0	0	0	0

decimal
0
1
2
3
4
5
6
7
8
9
10
-

bc1	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
0							
0							
0							
0							
0							
0							
0							
0							
0							
1	0	1	1	0	0	0	0
0	0	0	0	0	0	0	1



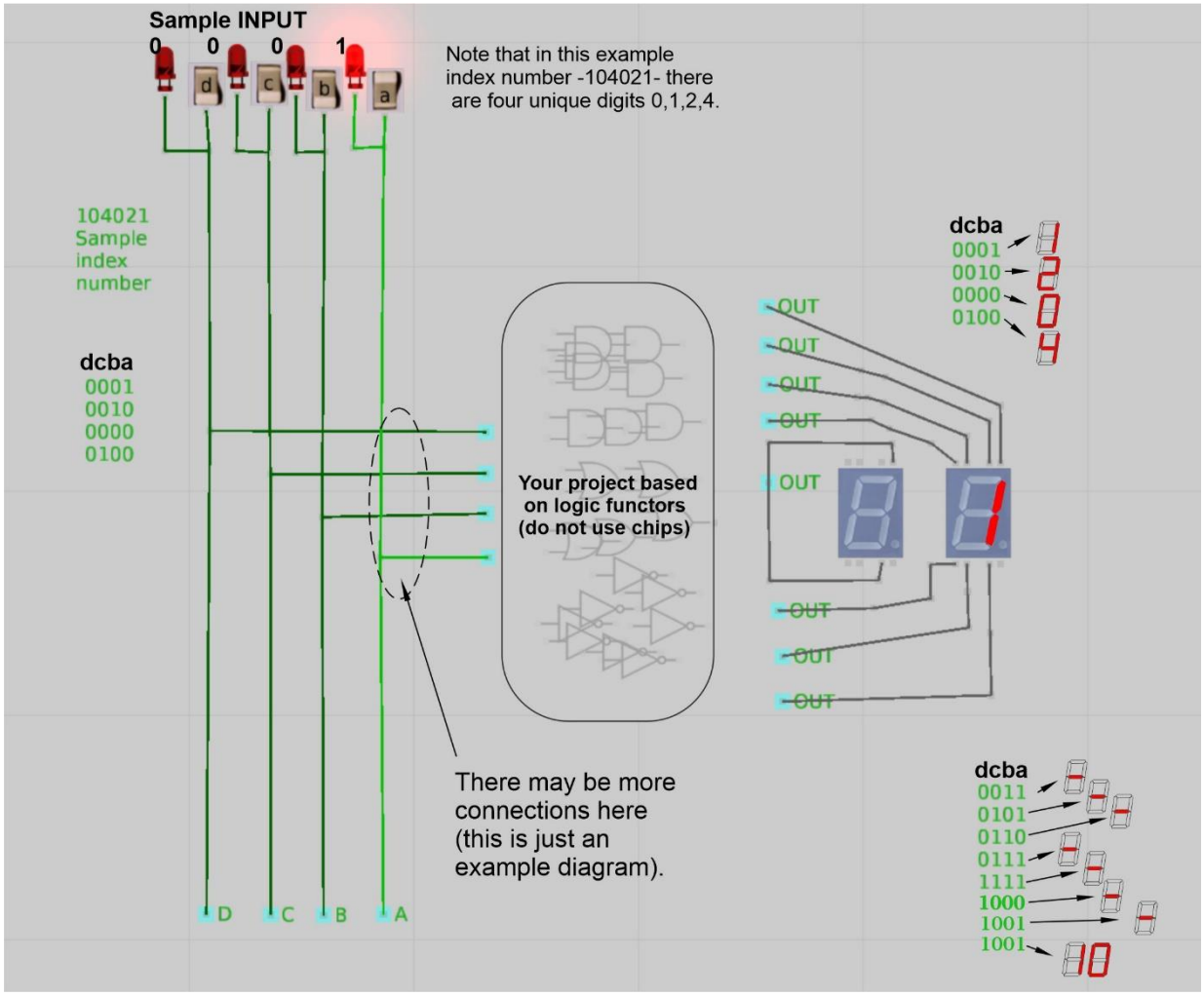
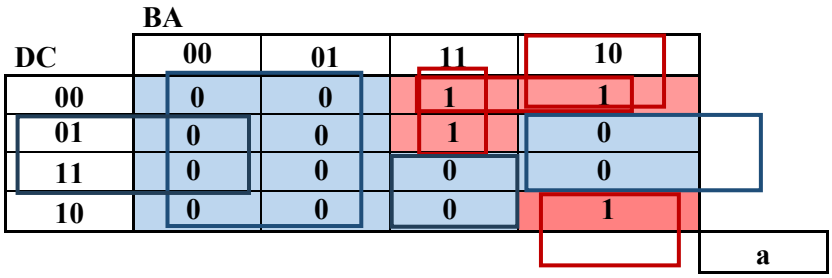
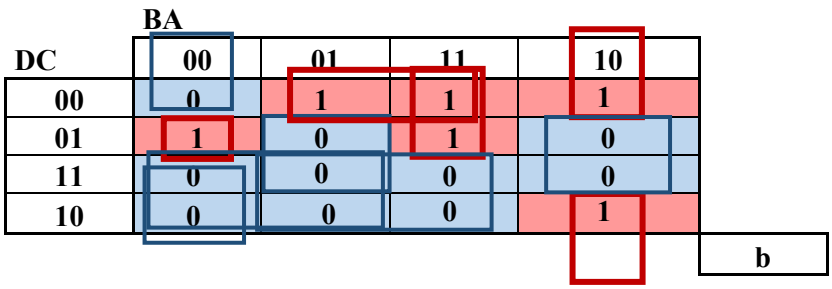


Fig 2. Decoder design using a seven-segment LED display

Binary code				decimal	bc1	a	b	c	d	e	f	g
D	C	B	A									
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	0	0	0	0
0	0	1	0	2	0	1	1	0	1	1	0	1
0	0	1	1	3	0	1	1	1	1	0	0	1
0	1	0	0	4	0	0	1	1	0	0	1	1
0	1	0	1	5	0	0	0	0	0	0	0	1
0	1	1	0	6	0	0	0	0	0	0	0	1
0	1	1	1	7	0	1	1	1	0	0	0	0
1	0	0	0	8	0	0	0	0	0	0	0	1
1	0	0	1	9	0	0	0	0	0	0	0	1
1	0	1	0	10	1	0	1	1	0	0	0	0
1	0	1	1	-	0	0	0	0	0	0	0	1
1	1	0	0	-	0	0	0	0	0	0	0	1
1	1	0	1	-	0	0	0	0	0	0	0	1
1	1	1	0	-	0	0	0	0	0	0	0	1
1	1	1	1	-	0	0	0	0	0	0	0	1

Mark reductions in the same tables using colored loops, e.g., red for implicants and blue for implicants.





DC \ BA	00	01	11	10
00	0	1	1	0
01	1	0	1	0
11	0	0	0	0
10	0	0	0	1

c

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	1

d

DC \ BA	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	1

e

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	0	1

f

DC \ BA	00	01	11	10
00	1	0	1	1
01	1	1	0	1
11	1	1	1	1
10	1	1	1	0

g

DC	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	1

bc1

PLEASE RECORD FUNCTIONS without reduction

$$Y_a = \bar{D}\bar{C}BA + \bar{D}\bar{C}B\bar{A} + \bar{D}CBA + D\bar{C}B\bar{A}$$

$$Y_b = \bar{D}\bar{C}\bar{B}A + \bar{D}\bar{C}BA + \bar{D}\bar{C}B\bar{A} + \bar{D}C\bar{B}\bar{A} + \bar{D}CBA + D\bar{C}B\bar{A}$$

$$Y_c = \bar{D}\bar{C}\bar{B}A + \bar{D}\bar{C}BA + \bar{D}C\bar{B}\bar{A} + \bar{D}CBA + D\bar{C}B\bar{A}$$

$$Y_d = \bar{D}\bar{C}BA + \bar{D}\bar{C}B\bar{A} + D\bar{C}B\bar{A}$$

$$Y_e = \bar{D}\bar{C}B\bar{A} + D\bar{C}B\bar{A}$$

$$Y_f = \bar{D}C\bar{B}\bar{A} + D\bar{C}B\bar{A}$$

$$Y_g = \bar{D}\bar{C}\bar{B}\bar{A} + \bar{D}\bar{C}BA + \bar{D}\bar{C}B\bar{A} + \bar{D}C\bar{B}\bar{A} + \bar{D}C\bar{B}A + \bar{D}CB\bar{A} + DC\bar{B}\bar{A} + DC\bar{B}A + DCBA + DCB\bar{A} + D\bar{C}\bar{B}\bar{A} + D\bar{C}\bar{B}A + D\bar{C}BA$$

$$Y_{bc1} = D\bar{C}B\bar{A}$$

PLEASE ENTER THE FUNCTIONS for IMPLICANTS:

$$Y_a = \bar{D}\bar{C}B + \bar{D}BA + \bar{C}B\bar{A}$$

$$Y_b = \bar{D}\bar{C}A + \bar{D}BA + \bar{C}B\bar{A} + \bar{D}C\bar{B}\bar{A}$$

$$Y_c = \bar{D}\bar{C}A + \bar{D}BA + \bar{D}C\bar{B}\bar{A} + D\bar{C}B\bar{A}$$

$$Y_d = \bar{D}\bar{C}B + \bar{C}B\bar{A}$$

$$Y_e = \bar{C}B\bar{A}$$

$$Y_f = \bar{D}C\bar{B}\bar{A} + D\bar{C}B\bar{A}$$

$$Y_g = \bar{B}\bar{A} + \bar{D}\bar{C}B + C\bar{A} + C\bar{B} + DA$$

$$Y_{bc1} = D\bar{C}B\bar{A}$$

PLEASE ENTER FUNCTIONS for IMPLICENTS:

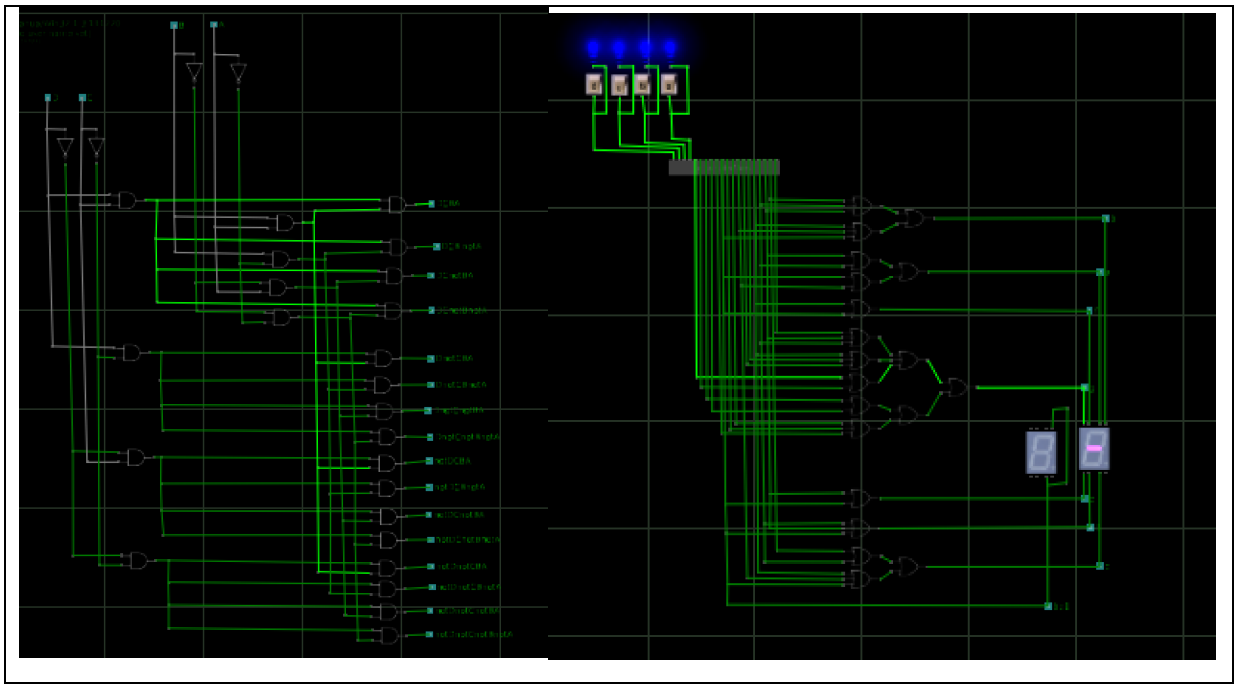
$$Y_a = B * (\bar{C} + A) * (\bar{D} + \bar{B} + \bar{A})$$

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

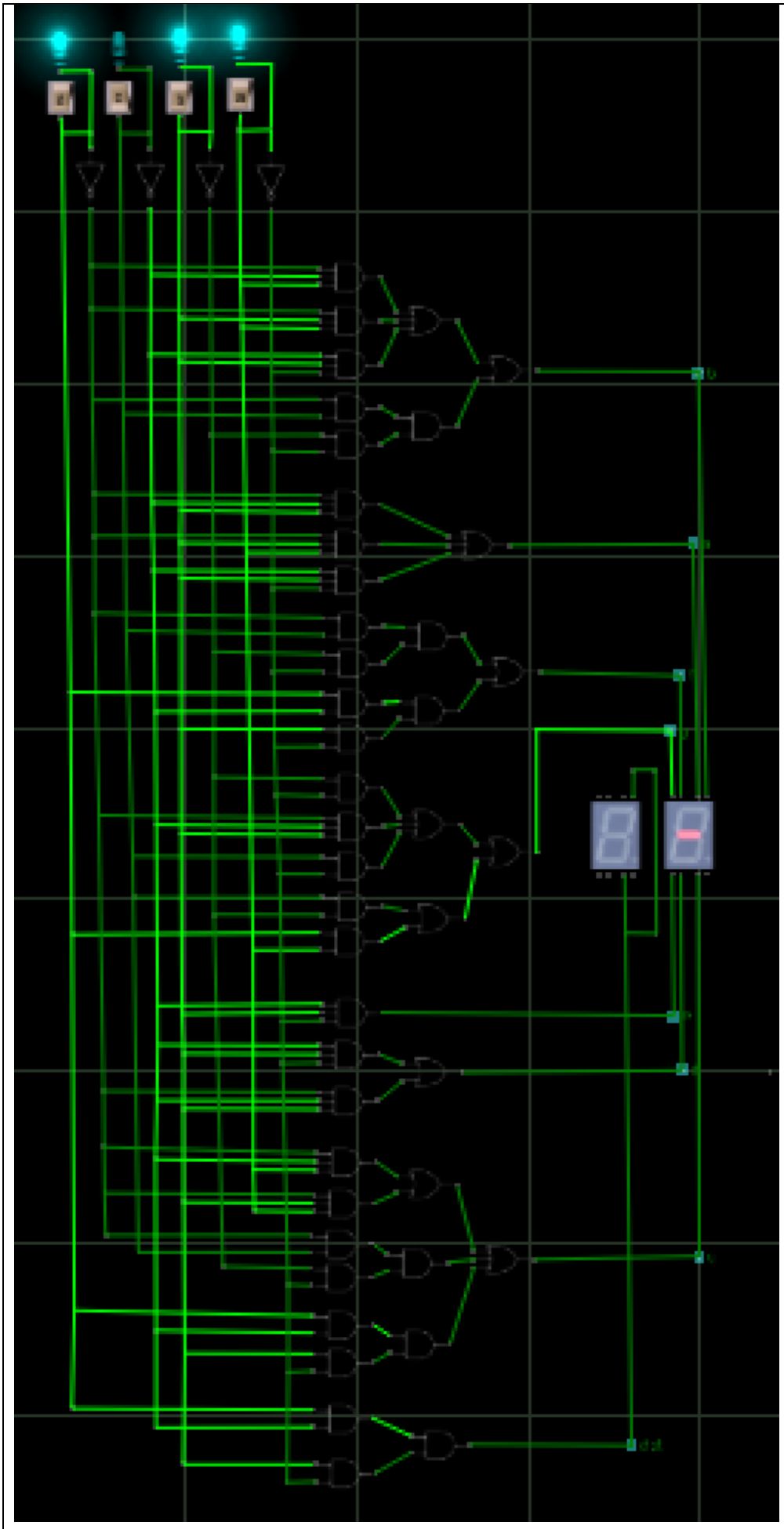
$$Y_c = (D + C + A) * (\bar{C} + B + \bar{A}) * (\bar{C} + \bar{B} + A) * (\bar{D} + B) * (\bar{D} + \bar{A})$$

$$Y_d = B * \overline{C} * (\overline{D} + \overline{A})$$
$$Y_e = B\overline{A}\overline{C}$$
$$Y_f = (D + C) * \overline{A} * (\overline{C} + \overline{B}) * (\overline{D} + B)$$
$$Y_g = (D + C + B + \overline{A}) * (D + \overline{C} + \overline{B} + \overline{A}) * (\overline{D} + C + \overline{B} + A)$$
$$Y_{bc1} = BD\overline{C}$$

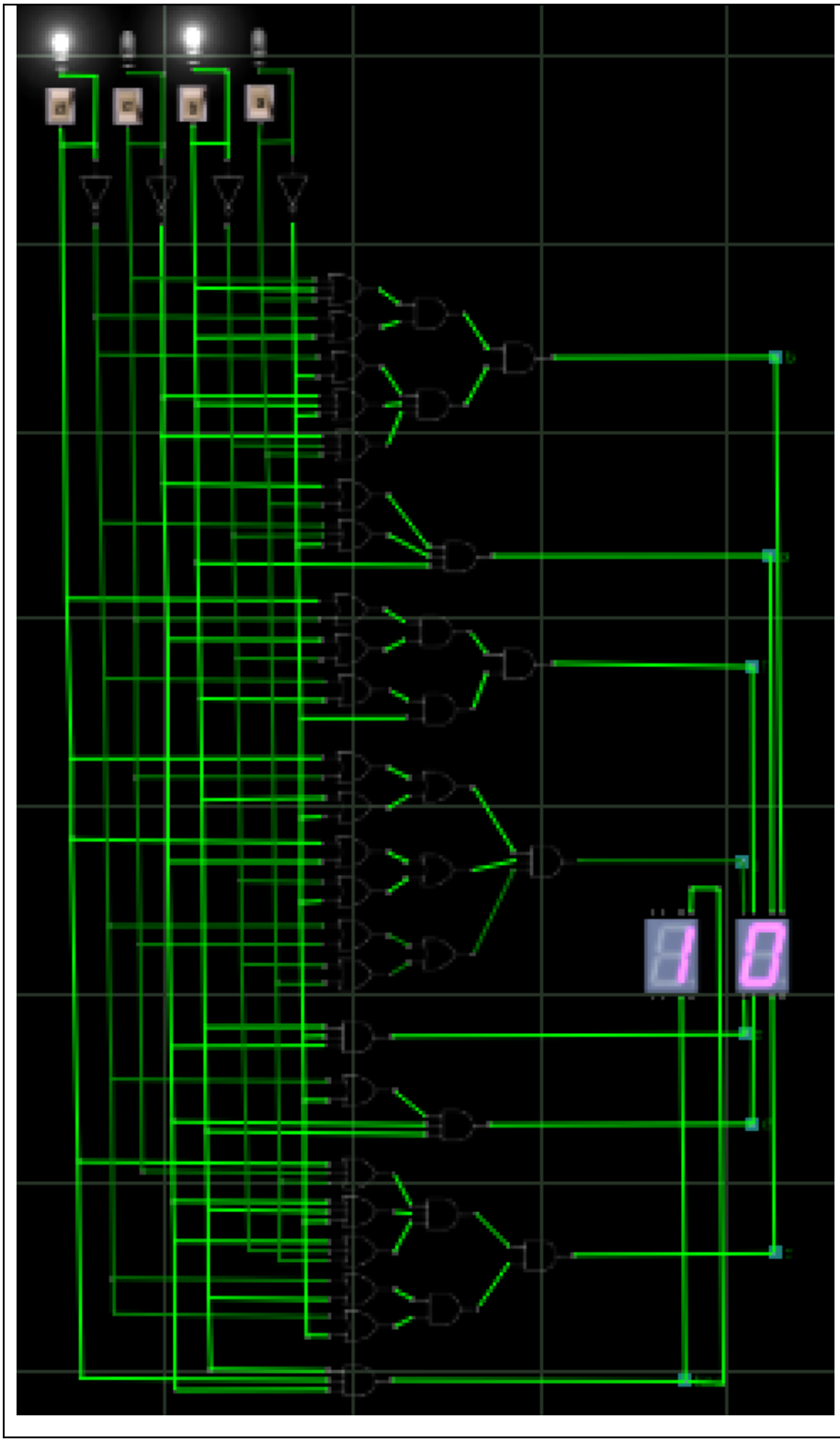
PLEASE PERFORM THE SIMULATION FOR ALL THREE CASES and post it below. Without reduction



Implicant method



The implicent method



PLEASE REMEMBER TO ATTACH THE SIMULATION FILES

Conclusions:

Which case is the most economically justified?

It’s obvious that the most economically both methods justified: implicants and implicents with reduction.