Designing a 7-Segment Display Circuit for Hexadecimal Numbers Using Karnaugh Maps

Buege Mahara Putra - 13523037[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]buege.putra@gmail.com, 13523037@std.stei.itb.ac.id

***Abstract*— This paper presents the design and optimization of a 7-segment display circuit capable of representing hexadecimal numbers (0–9, A–F) using Karnaugh maps. Starting with the construction of a truth table that maps 4-bit binary inputs to the respective segment outputs, Karnaugh maps are then utilized to simplify the Boolean expressions for each segment (a–g), resulting in reduced logic complexity and a cost-effective circuit design. This study showcases the utility of Karnaugh maps in minimizing Boolean expressions and demonstrates their application in practical digital circuit design.**

***Keywords*—7-segment display, digital circuit design, hexadecimal numbers, Karnaugh maps.**

I. Introduction

7-segment displays are ubiquitous in digital electronics, serving as an efficient and straightforward method for displaying numerical digits. Composed of seven individual segments (labeled a to g), these displays can form various characters by illuminating specific segments in different patterns. Due to their simplicity and wide application in devices such as clocks, calculators, and digital meters, 7-segment displays have become a staple in many consumer and industrial products.

The ability to display hexadecimal numbers (0–9, A–F) on a 7-segment display is particularly valuable in digital systems, as hexadecimal notation is frequently used in computing to represent binary data in a more compact and human-readable format. Each hexadecimal digit can be represented by a 4-bit binary value, and there are 16 unique symbols in the hexadecimal system. The task of designing a circuit to correctly display these 16 hexadecimal digits requires determining how to map each 4-bit binary input to the corresponding combination of segments on the 7-segment display.

This paper demonstrates the design and optimization of a 7-segment display circuit for hexadecimal numbers using Karnaugh maps. The design process begins with creating a truth table that associates each 4-bit input (representing hexadecimal digits) with the correct segment outputs (a to g). Through the use of Karnaugh maps, the Boolean expressions for each segment can be simplified, reducing the complexity of the required logic circuit.

The objective of this paper is to showcase the application of Karnaugh maps for minimizing Boolean expressions and deriving a circuit that can efficiently drive the 7-segment display for all 16 hexadecimal digits. By reducing the number of gates and simplifying the circuit design, this approach ensures an optimized, cost-effective solution for implementing a 7-segment display system.

II. Theoretical Basis

A. 7-Segment Display

A 7-segment display is an electronic display device composed of seven individual light-emitting elements (or segments) arranged in a figure-eight configuration. These segments, labeled from "a" to "g," can be lit individually or in combinations to form different numeric or alphanumeric characters. The display is typically used to represent decimal digits (0–9), but can also be adapted to show letters and other symbols in certain cases.

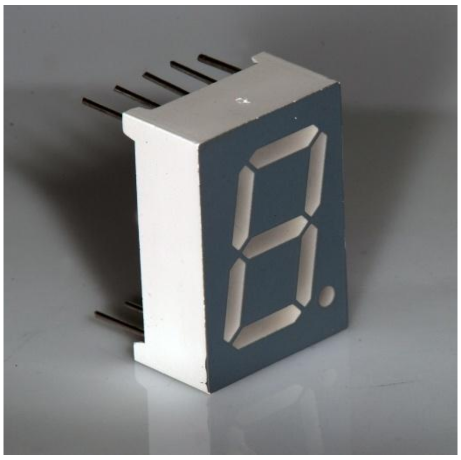


Fig. 1. 7-segment display.
Source: commons.wikimedia.org/wiki/File: Seven_segment_01_Pengo.jpg

Each of the seven segments (a to g) can either be ON (lit) or OFF, allowing for a total of 128 (2^7) possible combinations. For basic numerical displays, only a subset of these combinations are utilized to represent digits 0–9. For instance, to display the digit "0", segments a, b, c, e, f, and g are lit, leaving segment d off.

For hexadecimal numbers, which require the display of the digits 0–9 as well as the letters A-F, more combinations of segments are used. This necessitates a precise mapping between each hexadecimal digit and the specific segments to be lit. The 7-segment display offers a straightforward way to convey numeric values visually, but designing the underlying circuit to accurately light the correct segments for each input requires careful analysis of the segment patterns.

7-segment displays offer a straightforward solution for displaying numeric values efficiently while maintaining versatility in design and functionality. Their widespread adoption stems from their ability to integrate seamlessly into various electronic systems, providing reliable and readable outputs under different conditions [1].

B. Hexadecimal Number System

The hexadecimal number system, commonly referred to as hex, is a base-16 numeral system that employs sixteen distinct symbols to represent values. These symbols include the digits 0 through 9 and the letters A through F, where A represents 10, B represents 11, C represents 12, D represents 13, E represents 14, and F represents 15. Each hexadecimal digit also corresponds to a 4-bit binary value, as the base-16 system can be expressed as a direct mapping to 4-bit binary numbers. The significance of the hexadecimal system lies in its ability to provide a more compact and human-readable representation of binary numbers, which are fundamental to computer operations and digital electronics.

The hexadecimal number system is extensively utilized in various computing applications due to its compactness and ease of conversion to and from binary. It serves as a bridge between human-readable formats and machine-level data representation. Common applications include memory addressing in programming, color codes in web design (e.g., RGB color values), and data representation in cryptography.

C. Boolean Algebra

Boolean algebra is a mathematical framework that deals with binary variables and logical operations. It was introduced by the English mathematician George Boole in his book *The Mathematical Analysis of Logic* (1847), aiming to formalize the principles of logical reasoning through a symbolic representation. Boolean algebra is fundamentally different from traditional algebra, as it operates on two discrete values: true (1) and false (0). This binary nature allows for the manipulation of logical statements and the construction of complex logical expressions.

At its core, Boolean algebra employs three primary operations: conjunction (AND), disjunction (OR), and negation (NOT). These operations correspond to logical connectives that combine or modify propositions. The AND operation, denoted as $A \wedge B$ or $A \cdot B$, yields true only when both operands are true. Conversely, the OR operation, represented as $A \vee B$ or $A + B$, is true if at least one operand is true. The NOT operation, indicated as $\neg A$ or A', inverts the truth value of a proposition.

Boolean algebra consists of several fundamental identities that govern the behavior of logical operations. These identities are as follows [2].

1. Law of the double complement

$$(x')' = x \tag{1}$$

2. Idempotent laws

$$x + x = x \tag{2}$$

$$x \cdot x = x \tag{3}$$

3. Identity laws

$$x + 0 = x \tag{4}$$
$$x \cdot 1 = x \tag{5}$$

4. Domination laws

$$x + 1 = 1 \tag{6}$$
$$x \cdot 0 = 0 \tag{7}$$

5. Commutative laws

$$x + y = y + x \tag{8}$$
$$xy = yx \tag{9}$$

6. Associative laws

$$x + (y + z) = (x + y) + z \tag{10}$$
$$x(yz) = (xy)z \tag{11}$$

7. Distributive laws

$$x + yz = (x + y)(x + z) \tag{12}$$
$$x(y + z) = xy + xz \tag{13}$$

8. De Morgan’s laws

$$(xy)' = x' + y' \tag{14}$$
$$(x + y)' = x'y' \tag{15}$$

9. Absorption laws

$$x + xy = x \tag{16}$$
$$x(x + y) = x \tag{17}$$

10. Unit property

$$x + x' = 1 \tag{18}$$

11. Zero property

$$xx' = 0 \tag{19}$$

A literal is defined as a Boolean variable or its complement. A minterm of the Boolean variables $x_1, x_2, \dots, x_n$ is a Boolean product $y_1 y_2 \dots y_n$, where each y_i is either x_i or x_i'. In essence, a minterm is the product of n literals, with exactly one literal for each variable [2]. To form a minterm, each variable that has a value of 0 is expressed in its complement form, while the variable with a value of 1 is expressed as is [3].

A maxterm of the Boolean variables $x_1, x_2, \dots, x_n$ is a Boolean sum $y_1 + y_2 + \cdots + y_n$, where each y_i is either x_i or x_i'. Thus, a maxterm is the sum of n literals, with one literal corresponding to each variable [2]. To form a maxterm, each variable that has a value of 0 is expressed as is, while the variable with a value of 1 is expressed in its complement form [3].

Canonical forms are standardized ways of representing Boolean functions using minterms or maxterms. These forms ensure that a Boolean function is expressed in a consistent manner, which is useful for systematic analysis and simplification.

There are two primary types of canonical forms, Sum of Minterms (SOP) and Product of Maxterms (POS). The Sum Of Minterms (SOP) form expresses a Boolean function as a sum of its minterms, for example:

$$f(x,y,z) = x'y'z + xy'z' + xyz \tag{20}$$

The Product of Maxterms (POS) represents a Boolean function as a product of its maxterms, for example:

$$g(x,y,z) = (x+y+z)(x+y'+z)(x'+y+z')(x'+y'+z) \tag{21}$$

D. Karnaugh Map

Karnaugh map (K-map) is a graphical method to simplify Boolean functions involving a relatively small number of variables, by reducing the number of terms in the expression. Introduced by Maurice Karnaugh in 1953, the method is based on earlier work by E.W. Veitch. K-maps are typically applied to functions with six or fewer variables and provide a visual approach for simplifying sum-of-products expansions. However, they are not designed for automating the process [2].

A K-map is a grid where each cell represents a possible combination of input values for the variables. The cells are arranged in a way that ensures adjacent cells differ by only one variable (this is known as the Gray code order). The number of cells in the K-map corresponds to the number of possible input combinations, which is 2^n, where n is the number of variables in the Boolean function.

To begin simplifying a Boolean functions, the cells of the K-map are filled with 1s and 0s, corresponding to the values of the function for each combination of variables. A 1 is placed in a cell if the Boolean function evaluates to 1 for the combination of variables represented by that cell; otherwise, a 0 is placed.

The goal of using the K-map is to group adjacent cells that contain 1s. These groups must have sizes that are powers of 2 (e.g., 1, 2, 4, 8 cells, etc.). The larger the group, the simpler the resulting Boolean expression will be. Each group represents a simplified product term in the sum-of-products (SOP) form. The simplified Boolean expression is obtained by writing a term for each group, where the variables that change between the cells of the group are excluded.

By forming the largest possible groups of 1s, the number of terms in the Boolean expression is minimized. Overlapping groups can be formed to cover all the 1s in the map. The goal is to cover all the 1s with the fewest groups.

To demonstrate, consider the Karnaugh map of the function $f(w,x,y,z)$ in Fig. 2. After grouping the 1s, the function can be simplified into:

$$f(w,x,y,z) = wy' + yz' + w'x'z \tag{22}$$

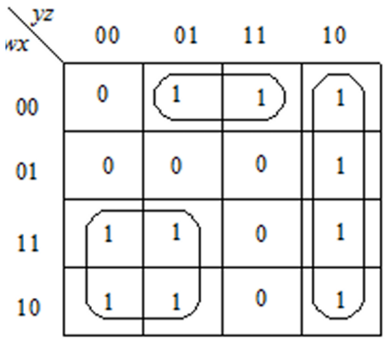


Fig. 2. K-map of $f(w,x,y,z)$.
Source: [4]

In some circuits, we may only care about the output for specific combinations of input values, while other combinations are either impossible or irrelevant. This flexibility allows us to further simplify the circuit because the output values for those combinations can be chosen freely. These combinations, for which the output can be arbitrarily assigned, are known as *don't care conditions*.

In a K-map, these don't care conditions are marked with an "X" or "D". During the minimization process, these don't care conditions are treated as 1s in order to form the largest possible groups in the K-map, further reducing the number of terms in the simplified Boolean expression, leading to a simpler and more efficient circuit.

E. Logic Gates

Logic gates are the fundamental building blocks of digital circuits. They perform basic logical operations on one or more binary inputs to produce a single binary output. Because of this nature, logic circuits can be modelled using Boolean algebra [2].

There are various types of logic gates, but all of them are composed of 3 fundamental types: AND gate, OR gate, and NOT gate (inverter).

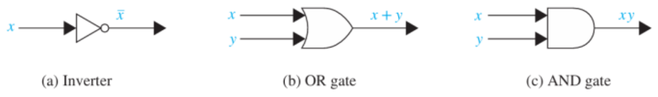


Fig. 3. Fundamental logic gates.
Source: [2]

Logic gates are used extensively in digital electronics to implement Boolean functions, enabling the design of circuits that process and manipulate binary data. Logic gates form the basis for more complex components like multiplexers, decoders, and flip-flops, which are integral to modern digital systems.

III. IMPLEMENTATION

To design the circuit, we will follow these steps: constructing a truth table, using Karnaugh maps to simplify the Boolean expressions, and finally implementing the simplified Boolean expressions into a logic circuit.

A. Truth Table Design

The 7-segment display has seven segments labeled a–g, which can be turned on or off depending on the hexadecimal digit being displayed. For reference, we will use the 7-segment display labeling in Fig. 3. We will begin by constructing a truth table that maps each of the 16 hexadecimal digits (0–F) to the corresponding combination of segments. The input to the display will be a 4-bit binary number representing one of the hexadecimal digits (0–F).

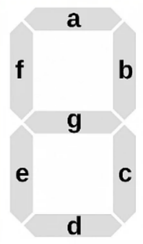


Fig. 4. 7-segment display labelled a—g.
Source: flyrobo.in/blog/7-segment-display

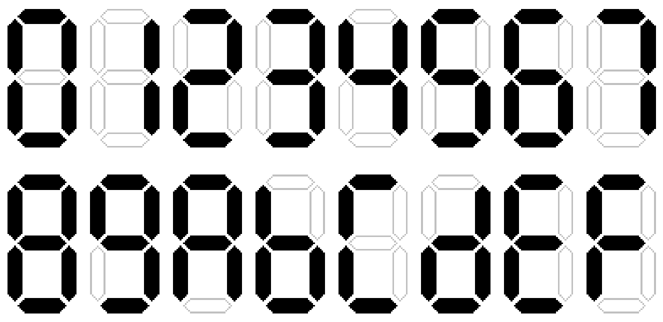


Fig. 5. Hexadecimal digits in 7-segment display.
Source: electronics.stackexchange.com/questions/373034/hex-to-7-segment-decoder-for-a-common-anode-7-seg-display

Table 1. Truth table of hexadecimal in 7-segment display.

Hex	Binary	a	b	c	d	e	f	g
0	0000	1	1	1	1	1	1	0
1	0001	0	1	1	0	0	0	0
2	0010	1	1	0	1	1	0	1
3	0011	1	1	1	1	0	0	1
4	0100	0	1	1	0	0	1	1
5	0101	1	0	1	1	0	1	1
6	0110	1	0	1	1	1	1	1
7	0111	1	1	1	0	0	0	0
8	1000	1	1	1	1	1	1	1
9	1001	1	1	1	1	0	1	1
A	1010	1	1	1	0	1	1	1
B	1011	0	0	1	1	1	1	1
C	1100	1	0	0	1	1	1	0
D	1101	0	1	1	1	1	0	1
E	1110	1	0	0	1	1	1	1
F	1111	1	0	0	0	1	1	1

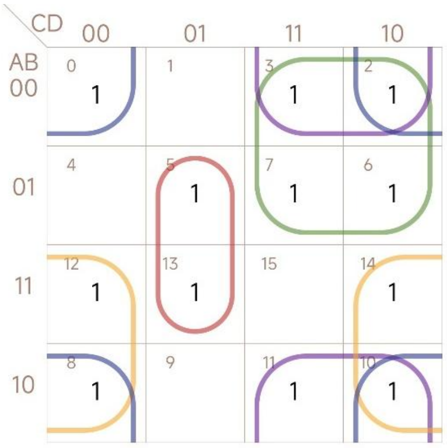
In this truth table, a "1" indicates that the corresponding segment is lit, and a "0" indicates that the segment is off. For the binary, we will name the digits as literal A, B, C, D according to the order of the digits, starting from left to right. For example, the hex A, which has the binary 1010, will have $A = 1,\ B = 0,$ $C = 1$, and $D = 0$.

B. Karnaugh Map Simplification

Once the truth table is complete, we can now use Karnaugh maps to simplify the Boolean expressions for each segment (a–g). A Karnaugh map helps minimize the Boolean expression by grouping adjacent "1" cells and deriving the simplest sum-of-products (SOP) expressions.

Once we have the simplified Boolean expressions for each segment, we can implement the circuit. The expressions can be realized using basic logic gates such as AND, OR, and NOT gates. The final circuit diagram will consist of seven independent circuits, one for each segment (a–g).

1. Segment a



AB \ CD	00	01	11	10
00	0: 1	1:	3: 1	2: 1
01	4:	5: 1	7: 1	6: 1
11	12: 1	13: 1	15:	14: 1
10	8: 1	9:	11: 1	10: 1

Fig. 6. Karnaugh map of segment a.

$$a(A, B, C, D) = BC'D + A'C + B'C + AD' + B'D' \qquad (23)$$

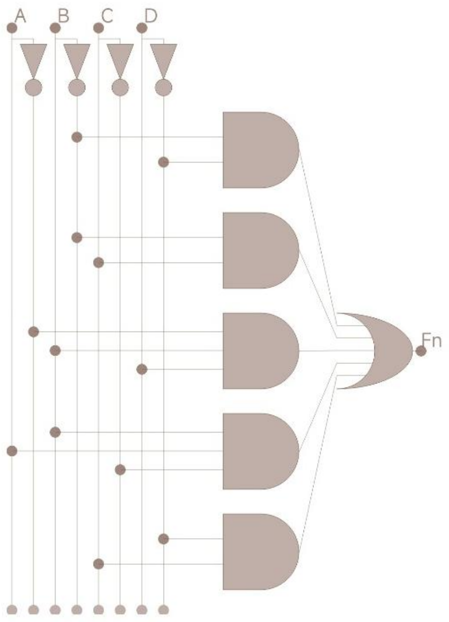


Fig. 7. Logic circuit of segment a.

2. Segment b

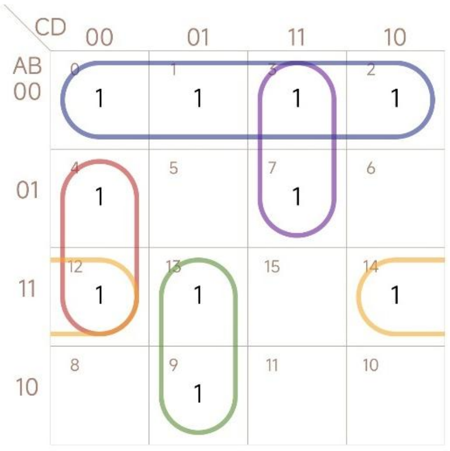


Fig. 8. Karnaugh map of segment b.

$$b(A,B,C,D) = BC'D' + AC'D + A'CD + ABD' + A'B' \quad (24)$$

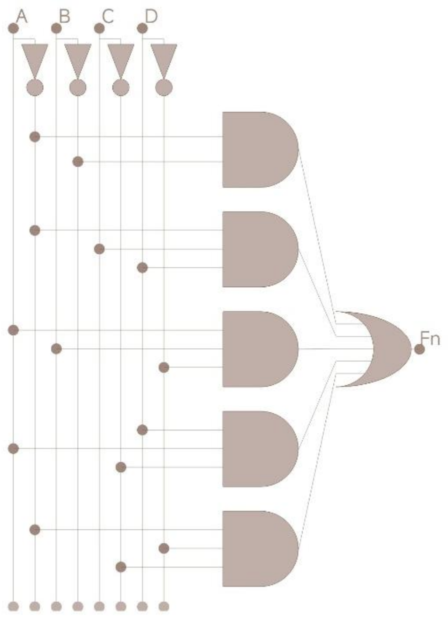


Fig. 9. Logic circuit of segment b.

3. Segment c

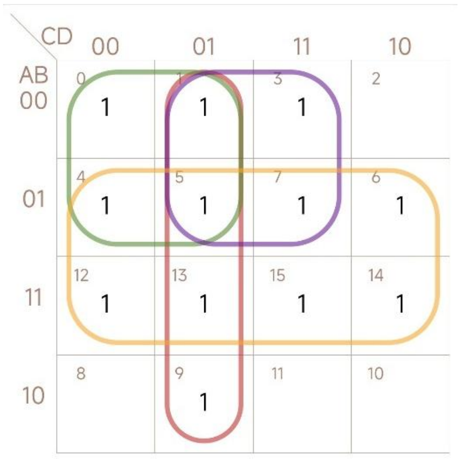


Fig. 10. Karnaugh map of segment c.

$$c(A,B,C,D) = C'D + A'C' + A'D + B \quad (25)$$

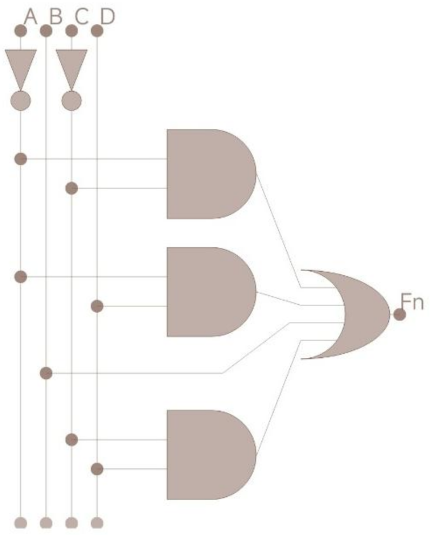


Fig. 11. Logic circuit of segment c.

4. Segment d

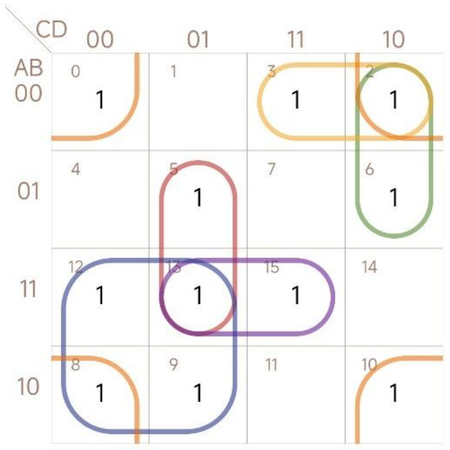


Fig. 12. Karnaugh map of segment d.

$$d(A,B,C,D) = BC'D + A'CD' + ABD + A'B'C + AC' + B'D' \quad (26)$$

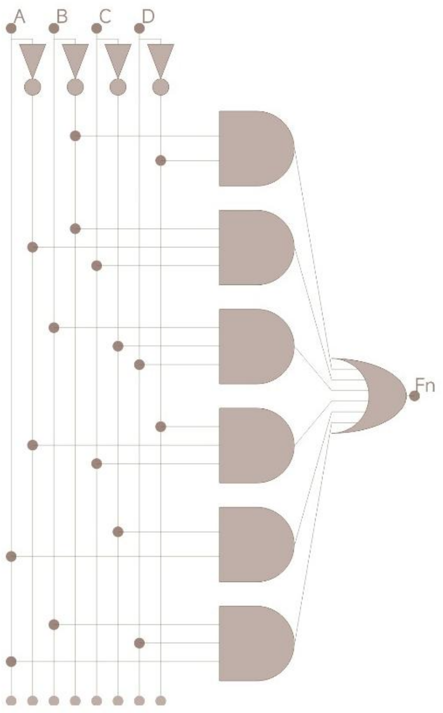


Fig. 13. Logic circuit of segment d.

5. Segment e

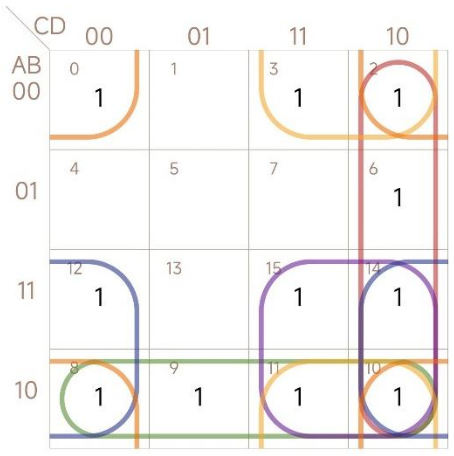


Fig. 14. Karnaugh map of segment e.

$$e(A,B,C,D) = CD' + AB' + AC + B'C + AD' + B'D' \quad (27)$$

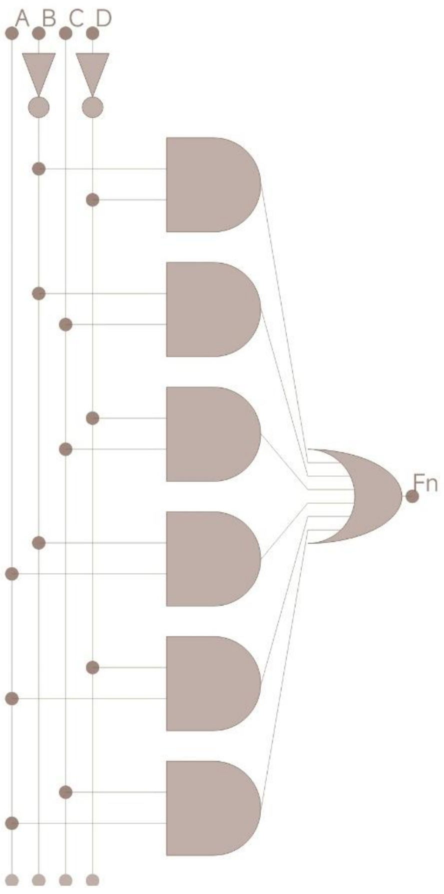


Fig. 15. Logic circuit of segment e.

6. Segment f

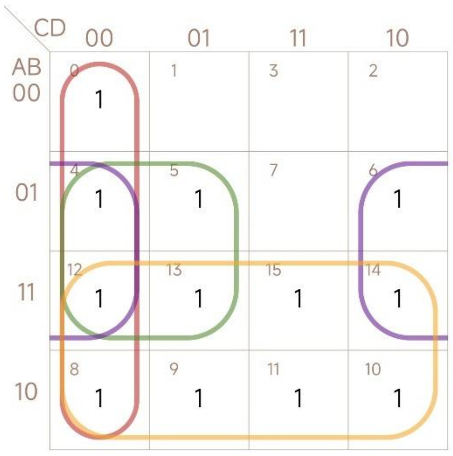


Fig. 16. Karnaugh map of segment f.

$$f(A,B,C,D) = C'D' + BC' + BD' + A \qquad (28)$$

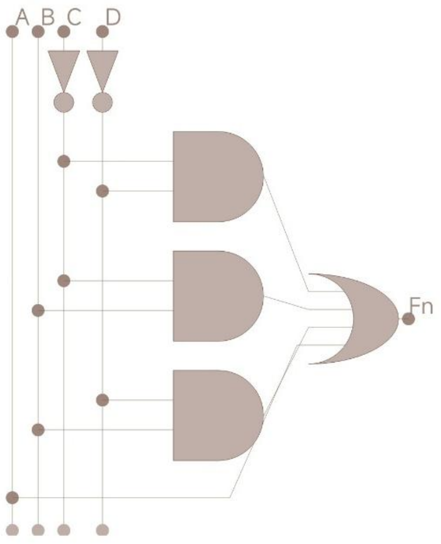


Fig. 17. Logic circuit of segment f.

7. Segment g

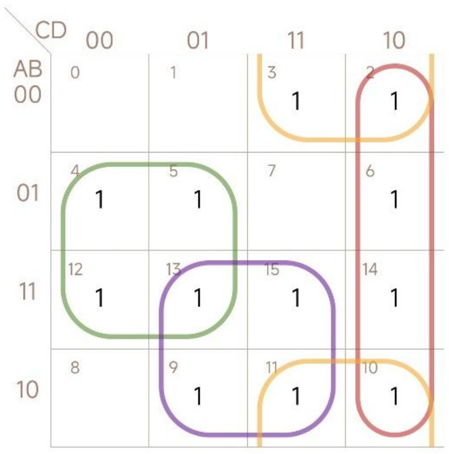


Fig. 18. Karnaugh map of segment g.

$$g(A,B,C,D) = CD' + BC' + AD + B'C \qquad (29)$$

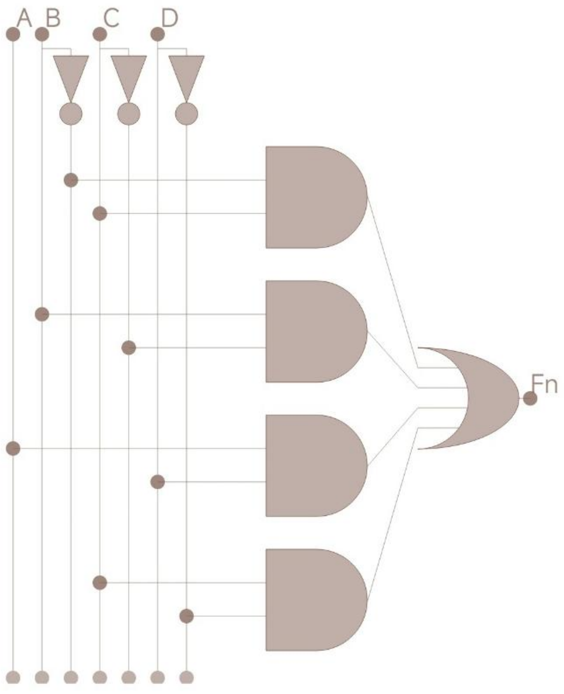


Fig. 19. Logic circuit of segment g.

VI. Conclusion

The design and optimization of a 7-segment display circuit for hexadecimal numbers demonstrate the effective use of Boolean algebra and Karnaugh maps in digital logic design. By systematically constructing a truth table, simplifying the Boolean expressions for each segment, and implementing the resulting logic with gates, we developed a circuit capable of accurately displaying all 16 hexadecimal digits.

The approach shown in this paper provides a robust methodology for designing similar digital systems. The principles of Boolean algebra, combined with visual simplification tools like Karnaugh maps, are versatile and applicable to a wide range of problems in digital electronics.

VII. Acknowledgment

The author sincerely expresses gratitude towards the lecturers of IF1220 Discrete Mathematics, particularly Dr. Ir. Rinaldi Munir, M.T., the lecturer for class K-01, for his continuous guidance and expertise throughout the semester. The author also expresses gratitude to their friends and family for their endless support during the period of writing this paper.

References

[1] "7-Segment LED display." tutorialspoint.com/digital-electronics/seven-segment-led-display.htm (accessed Jan. 8, 2025 at 19.00 WIB).

[2] K. H. Rosen, *Discrete mathematics and its applications*. New York, Ny: Mcgraw-Hill, 2019.

[3] R. Munir, "Aljabar Boolean (Bag. 1)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/12-Aljabar-Boolean-(2024)-bagian1.pdf (accessed Jan. 8, 2025 at 07.34 WIB).

[4] R. Munir, "Aljabar Boolean (Bag. 2)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/13-Aljabar-Boolean-(2024)-bagian2.pdf (accessed Jan. 8, 2025 at 08.02 WIB).

Statement

I hereby declare that the paper I have written is my own work, not an excerpt or translation of someone else's paper, and is not plagiarized.

Bandung, 8 January 2025

[illegible]

Buege Mahara Putra
13523037