ÇUKUROVA ÜNİVERSİTESİ

Department
of
Computer Engineering

CEN 263
Digital Design

Autumn 2024

Lecture 3

Week 3 Outlines

- Minimization Techniques for Boolean Expressions.

- Karnaugh Maps (K-Maps) for Simplification.

Definitions

Minterms and **Maxterms** are fundamental concepts in Boolean algebra, commonly used in Boolean function simplification.

Let’s break them down:

Minterms

A minterm is a product (AND operation)
of all the variables in a Boolean function, where each variable appears in either its true form or complemented form (inverted).

Minterms correspond to individual rows in the truth table where the output of the Boolean function is 1 (TRUE).

Minterms

Key Points:

- Minterms are ANDed terms that represent a specific combination of input variables where the function is true.
- For n variables, there are __ possible minterms.
- Each minterm represents one row in the truth table where the output is 1.

Minterms

A	B	Y	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	0	$A\ B$	m_3

Figure 2.8 Truth table and minterms

Minterms

Example:

Let's consider a function with three variables: A, B, and C.

- **Minterm for $A = 0, B = 0, C = 0$:**

$$Minterm : \overline{A} \cdot \overline{B} \cdot \overline{C}$$

- **Minterm for $A = 1, B = 0, C = 1$:**

$$Minterm : A \cdot \overline{B} \cdot C$$

- **Sum-of-Minterms Expression**: If the function is true for certain minterms, you can express it as a **sum-of-minterms** (disjunction of ANDed terms). For example, if a function is true for minterms 1, 3, and 7:

$$F(A, B, C) = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C + A \cdot B \cdot C$$

In sum-of-minterms form, **each term represents a specific combination of inputs where the function is true**.

Minterms

$$F(A,B) = \Sigma(m_1, m_3)$$

$$F(A,B) = \Sigma(1,3)$$

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

$$Y = \Sigma(0,4,5)$$

Minterms

Adjacency of minterms:

Two minterms are adjacent if they differ by only one variable.

Minterms

Example:

Let's look at the minterms for 3 variables A, B, and C. The minterms for three variables are represented in binary as follows:

Minterm	ABC (Binary Representation)	Boolean Expression
0	000	$\overline{A}\overline{B}\overline{C}$
1	001	$\overline{A}\overline{B}C$
2	010	$\overline{A}B\overline{C}$
3	011	$\overline{A}BC$
4	100	$A\overline{B}\overline{C}$
5	101	$A\overline{B}C$
6	110	$AB\overline{C}$
7	111	ABC

Minterms

- **Adjacent minterms** differ by only **one bit** in their binary representations. For example:
  - Minterms 0 (000) and 1 (001) are adjacent because they differ only in the value of C (third bit).
  - Minterms 1 (001) and 3 (011) are adjacent because they differ only in the value of B (second bit).

Minterms

- **Minterm 0 (000)** is adjacent to:
  - Minterm 1 (001) because they differ only in C.
  - Minterm 2 (010) because they differ only in B.
  - Minterm 4 (100) because they differ only in A.
- **Minterm 1 (001)** is adjacent to:
  - Minterm 0 (000), as mentioned above.
  - Minterm 3 (011), which differs only in B.
  - Minterm 5 (101), which differs only in A.

Minterms

Why Adjacency Matters in Simplification

Minterms

Why Adjacency Matters in Simplification:

The goal of Karnaugh maps is to group adjacent minterms together to simplify the Boolean function by removing variables that change between them. If two minterms are adjacent, you can combine them to eliminate the variable in which they differ.

Example:

Let's look at the minterms for 3 variables A, B, and C. The minterms for three variables are represented in binary as follows:

Minterm	ABC (Binary Representation)	Boolean Expression
0	000	$\overline{A}\overline{B}\overline{C}$
1	001	$\overline{A}\overline{B}C$
2	010	$\overline{A}B\overline{C}$
3	011	$\overline{A}BC$
4	100	$A\overline{B}\overline{C}$
5	101	$A\overline{B}C$
6	110	$AB\overline{C}$
7	111	ABC

Example of Combining Adjacent Minterms:

Consider the minterms 0 and 1, which are adjacent in the K-map:

- Minterm 0: $\overline{A}\overline{B}\overline{C}$
- Minterm 1: $\overline{A}\overline{B}C$

Since these minterms differ only in C, we can combine them by factoring out $\overline{A}\overline{B}$:

$$\overline{A}\overline{B}(\overline{C} + C) = \overline{A}\overline{B} \cdot 1 = \overline{A}\overline{B}$$

Thus, the two minterms simplify to $\overline{A}\overline{B}$, effectively removing the C variable.

Maxterms

A maxterm is a sum (OR operation)
of all the variables in a Boolean function, where each variable appears either in its true form or complemented form.

Maxterms correspond to rows in the truth table where the output of the Boolean function is 0 (FALSE).

Maxterms

Key Points:

- Maxterms are ORed terms that represent a specific combination of input variables where the function is false.

- For n variables, there are __ possible maxterms.

- Each maxterm represents one row in the truth table where the output is 0.

Maxterms

Example:

Let's consider the same function with three variables: A, B, and C.

- **Maxterm for $A = 0, B = 0, C = 0$:**

$$Maxterm : (A + B + C)$$

- **Maxterm for $A = 1, B = 0, C = 1$:**

$$Maxterm : (\overline{A} + B + \overline{C})$$

- **Product-of-Maxterms Expression**: If the function is false for certain maxterms, you can express it as a **product-of-maxterms** (conjunction of ORed terms). For example, if a function is false for maxterms 0, 2, and 5:

$$F(A, B, C) = (A + B + C) \cdot (A + \overline{B} + C) \cdot (\overline{A} + B + \overline{C})$$

In **product-of-maxterms** form, **each term represents a specific combination of inputs where the function is false**.

Minterms & Maxterms

Truth Table Example:

Let's look at a simple truth table for 2 variables A and B:

A	B	Minterm	Maxterm	Function Output (F)
0	0	$\overline{A} \cdot \overline{B}$	$A + B$	0
0	1	$\overline{A} \cdot B$	$A + \overline{B}$	1
1	0	$A \cdot \overline{B}$	$\overline{A} + B$	1
1	1	$A \cdot B$	$\overline{A} + \overline{B}$	0

- **Sum-of-minterms (SOP)** for when $F = 1$ is:

$$F(A, B) = \overline{A} \cdot B + A \cdot \overline{B}$$

- **Product-of-maxterms (POS)** for when $F = 0$ is:

$$F(A, B) = (A + B) \cdot (\overline{A} + \overline{B})$$

Minterms & Maxterms

A	B	Binary	Minterm (m)	Maxterm (M)
0	0	00	$m_0 = \overline{A} \cdot \overline{B}$	$M_0 = (A + B)$
0	1	01	$m_1 = \overline{A} \cdot B$	$M_1 = (A + \overline{B})$
1	0	10	$m_2 = A \cdot \overline{B}$	$M_2 = (\overline{A} + B)$
1	1	11	$m_3 = A \cdot B$	$M_3 = (\overline{A} + \overline{B})$

Boolean theorems of one variable

Table 2.2 Boolean theorems of one variable

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

Theorems of Several Variables

Table 2.3 Boolean theorems of several variables

	Theorem		Dual	Name
T6	$B \bullet C = C \bullet B$	T6′	$B + C = C + B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	T7′	$(B + C) + D = B + (C + D)$	Associativity
T8	$(B \bullet C) + (B \bullet D) = B \bullet (C + D)$	T8′	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Distributivity
T9	$B \bullet (B + C) = B$	T9′	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	T10′	$(B + C) \bullet (B + \overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D)$ $= B \bullet C + \overline{B} \bullet D$	T11′	$(B + C) \bullet (\overline{B} + D) \bullet (C + D)$ $= (B + C) \bullet (\overline{B} + D)$	Consensus
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots}$ $= (\overline{B}_0 + \overline{B}_1 + \overline{B}_2 \ldots)$	T12′	$\overline{B_0 + B_1 + B_2 \ldots}$ $= (\overline{B}_0 \bullet \overline{B}_1 \bullet \overline{B}_2 \ldots)$	De Morgan's Theorem

Example

Example 2.6 EQUATION MINIMIZATION

Minimize Equation 2.3: $\overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} + A\overline{B}C.$

Example

Table 2.4 Equation minimization

Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$	
1	$\overline{B}\,\overline{C}(\overline{A} + A) + A\overline{B}C$	T8: Distributivity
2	$\overline{B}\,\overline{C}(1) + A\overline{B}C$	T5: Complements
3	$\overline{B}\,\overline{C} + A\overline{B}C$	T1: Identity

Example

Table 2.5 Improved equation minimization

Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	
1	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	T3: Idempotency
2	$\overline{B}\,\overline{C}(\overline{A}+A)+A\overline{B}(\overline{C}+C)$	T8: Distributivity
3	$\overline{B}\,\overline{C}(1)+A\overline{B}(1)$	T5: Complements
4	$\overline{B}\,\overline{C}+A\overline{B}$	T1: Identity

FROM LOGIC TO GATES

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

FROM LOGIC TO GATES

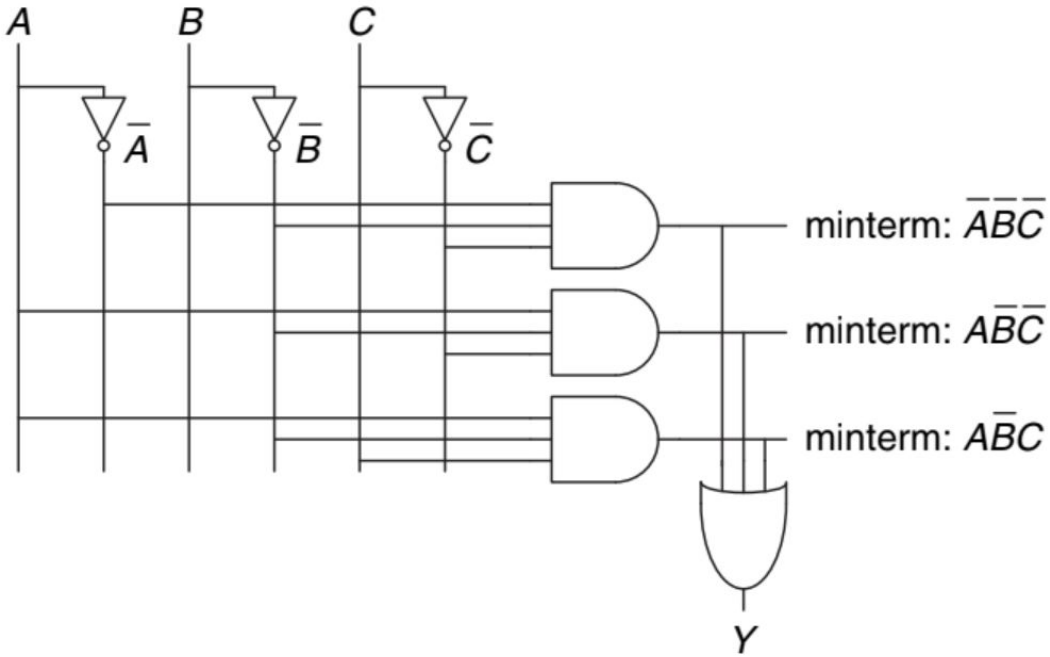


Figure 2.23 Schematic of $Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$

FROM LOGIC TO GATES

Table 2.5 Improved equation minimization

Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	
1	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	T3: Idempotency
2	$\overline{B}\,\overline{C}(\overline{A}+A)+A\overline{B}(\overline{C}+C)$	T8: Distributivity
3	$\overline{B}\,\overline{C}(1)+A\overline{B}(1)$	T5: Complements
4	$\overline{B}\,\overline{C}+A\overline{B}$	T1: Identity

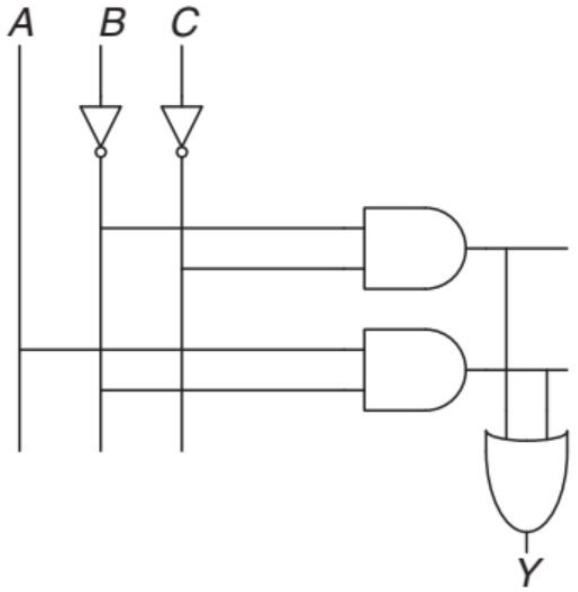


Figure 2.25 Schematic of $Y = \overline{B}\,\overline{C}+A\overline{B}$

KARNAUGH MAPS (K-MAPS)

Karnaugh Maps (K-maps) are a manual method for minimizing Boolean functions, typically used for up to 6 variables.

There are alternative methods to minimize Boolean functions, especially when the number of variables increases, making K-maps impractical.

KARNAUGH MAPS (K-MAPS)

Alternative Methods:

1. **Quine-McCluskey Method (Tabulation Method):**
   - This is a more systematic and algorithmic approach compared to K-maps.
   - It can handle any number of variables but becomes computationally expensive as the number of variables grows.
   - It involves creating a truth table, generating prime implicants, and reducing them to a minimal form.
2. **Espresso Algorithm:**
   - A heuristic algorithm used in computer-aided design (CAD) tools for logic minimization.
   - Efficient for handling a large number of variables.
   - Used in industry for optimizing logic circuits with many variables.
3. **Boolean Algebraic Simplification:**
   - Manually applying Boolean algebra rules (such as De Morgan's laws, distribution, absorption) to simplify functions.
   - Best suited for small functions or where intuition can lead to simplifications.
4. **Binary Decision Diagrams (BDD):**
   - A data structure used to represent Boolean functions.
   - Can handle a large number of variables efficiently, especially in automated systems.
5. **Petrick's Method:**
   - Another systematic method that complements the Quine-McCluskey algorithm for selecting a minimal cover from prime implicants.

Alternative methods will not be included in exams!

(Boolean Simplifications will be in the exam)

KARNAUGH MAPS (K-MAPS)

Why K-maps Are Limited to 6 Variables

Visualization Constraints: K-maps are drawn as a 2D grid, where each cell represents a minterm of the Boolean function. As the number of variables increases, the number of cells grows exponentially. For example:

- 3 variables require 8 cells.
- 4 variables require 16 cells.
- 6 variables require 64 cells.

Beyond 6 variables (which requires 2^6 = 64 cells), the map becomes **too large** and **complex** for easy **human visualization**, making it difficult to spot patterns or perform the grouping of adjacent 1's to minimize the function.

Pattern Recognition: The strength of K-maps lies in human pattern recognition. As the number of variables increases, it becomes harder for humans to identify adjacent groupings of 1's, which is critical for minimizing the function.

Truth Table vs. Karnaugh Map

2-variable function, f(A,B)

ID	A	B	f(A,B)
0	0	0	f(0,0)
1	0	1	f(0,1)
2	1	0	f(1,0)
3	1	1	f(1,1)

	B=0	B=1
A=0	f(0,0)	f(0,1)
A=1	f(1,0)	f(1,1)

Truth Table

An example of 2-variable function, f(A,B)

ID	A	B	f(A,B)	minterm
0	0	0	0	
1	0	1	1	A’B
2	1	0	1	AB’
3	1	1	1	AB

Function can be represented by sum of minterms:

$$f(A,B) = A'B+AB'+AB$$

This is not optimal however!

We want to minimize the number of literals and terms.

To minimize the number of terms.
We factor out common terms –

A’B+AB’+AB
= A’B+AB’+AB+AB
=(A’+A)B+A(B’+B)=B+A

Hence, we have f(A,B) = A+B

How can we guarantee the most reduced expression was reached?

- Boolean expressions can be minimized by combining terms
- K-maps minimize equations graphically

ID	A	B	f(A,B)
0	0	0	f(0,0)
1	0	1	f(0,1)
2	1	0	f(1,0)
3	1	1	f(1,1)

	B=0	B=1
A=0	A’B’	A’B
A=1	AB’	AB

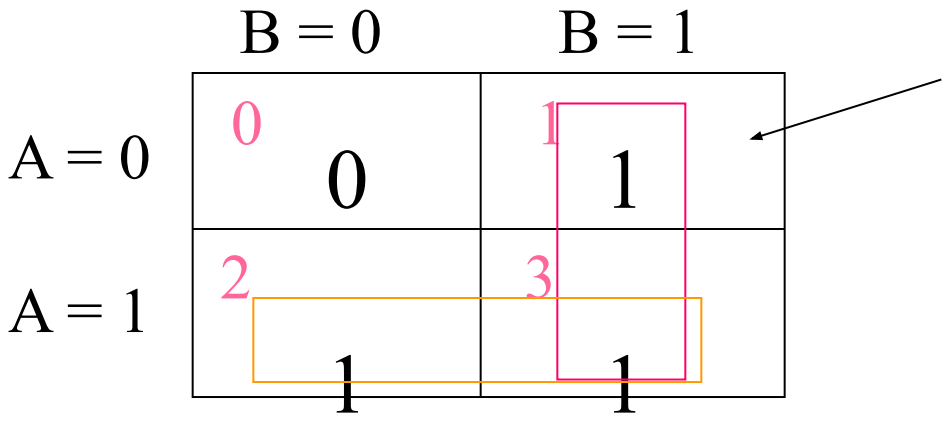
K-Map: Truth Table in 2 Dimensions

ID	A	B	f(A,B)
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

	B = 0	B = 1
A = 0	0 0	1 1
A = 1	2 1	3 1

K-Map: Truth Table in 2 Dimensions

ID	A	B	f(A,B)
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1



$$f(A,B) = A + B$$

Truth table

Id	a	b	c	f (a,b,c)
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Corresponding three variable K-map

Gray code (b,c)

	(0,0)	(0,1)	(1,1)	(1,0)
a = 0	0	1	3	2
a = 1	4	5	7	6

Truth table

Id	a	b	c	f (a,b,c)
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Corresponding three variable K-map

Gray code (b,c)

	(0,0)	(0,1)	(1,1)	(1,0)
a = 0	1	0	0	1
a = 1	1	0	0	1

Karnaugh Maps (K-Maps)

- K-maps minimize equations graphically
- Note that the label decides the order in the map

A	B	C	Y
0	0	0	**1**
0	0	1	**1**
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

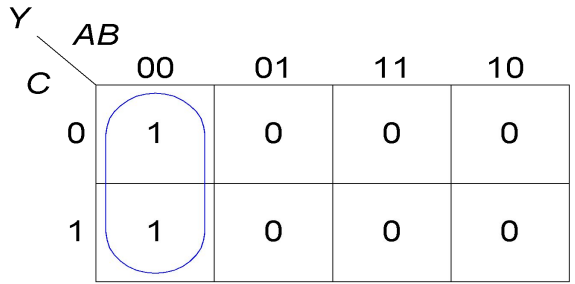
Y: C \ AB	00	01	11	10
0	**1**	0	0	0
1	**1**	0	0	0

Y: C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

K-map

- Circle 1's in adjacent squares
- Find rectangles which correspond to product terms in Boolean expression

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



$$y(A,B) = A'B'C'+A'B'C = A'B'(C'+C) = A'B'$$

Another 3-Input truth table

Id	a	b	c	f (a,b,c)
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	X

Corresponding K-map

	(0,0)	(0,1)	(1,1)	(1,0)
a = 0	0: 0	1: 0	3: 0	2: 1
a = 1	4: 1	5: 1	7: X	6: 1

Another 3-Input truth table

Id	a	b	c	f (a,b,c)
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	X

Corresponding K-map

	(0,0)	(0,1)	(1,1)	(1,0)
a = 0	0: 0	1: 0	3: 0	2: 1
a = 1	4: 1	5: 1	7: X	6: 1

f(a,b,c) = a + bc’

What about 4 Variable K-Map?

Assignment!

Consensus Theorem in View of K Maps

Consensus Theorem: A'B+AC+BC = A'B+AC

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Y: C \ AB	00	01	11	10
0	**0**	**1**	0	0
1	**0**	**1**	**1**	**1**

Y: C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Consensus Theorem in View of K Maps

Consensus Theorem: A’B+AC+BC = A’B+AC

The **Consensus Theorem** in Boolean algebra is a simplification rule that allows us to eliminate redundant terms in a Boolean expression. The theorem states:

$$A'B + AC + BC = A'B + AC$$

Explanation of the Theorem

The left side of the equation $A'B + AC + BC$ has three terms, and we are looking to simplify it by removing BC. According to the Consensus Theorem, we can eliminate this term without changing the overall output of the expression. Let's go through why this works.

Consensus Theorem in View of K Maps

Step-by-Step Proof

1. **Original Expression:**

$$A'B + AC + BC$$

2. **Rewrite the Term BC:** The term BC is only true when both B and C are true. However, if both B and C are true, then one of the other terms in the expression—$A'B$ or AC—will also be true, making BC redundant. Here's why:

3. **Case Analysis:** Let's analyze when each term is true:

   - $A'B$: This term is true when $A = 0$ and $B = 1$.
   - AC: This term is true when both $A = 1$ and $C = 1$.
   - BC: This term is true when both $B = 1$ and $C = 1$.

Consensus Theorem in View of K Maps

Notice that when $B = 1$ and $C = 1$ (making $BC = 1$), at least one of the other terms, $A'B$ or AC, will also be true depending on the value of A:

- If $A = 0$: $A'B$ will be true, making BC unnecessary.
- If $A = 1$: AC will be true, again making BC unnecessary.

4. **Conclusion:** Because BC is redundant in both cases, we can eliminate it without changing the expression's output.

Consensus Theorem in View of K Maps

Simplified Expression

Therefore, the simplified expression is:

$$A'B + AC$$

This means that:

$$A'B + AC + BC = A'B + AC$$

Intuition Behind the Consensus Theorem

The Consensus Theorem essentially removes redundant conditions. In this case, the term BC is redundant because the presence of either $A'B$ or AC ensures the expression will be true when B and C are true.

This simplification is particularly useful in digital logic design because it reduces the number of logic gates needed, optimizing the circuit design.

End of Week 3!