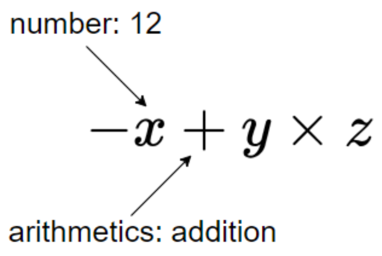
Sétif 1 University
Faculty of Sciences
Common Core Mathematics and Computer Science

1st semester (year 2025/26)
Machine Structures 1 course
Kara Abdelaziz professor

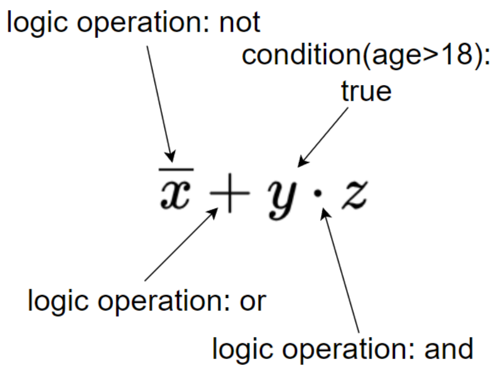
Chapter 4: Boolean Algebra

1. Introduction to Boolean Algebra

- Boolean Algebra is a branch of algebra in which the values of the variables are the truth values (True/False).
- The variables represent *Conditions*, and operations are logic operations, And, Or, Not…etc.
- A condition is an expression that evaluates to either *true* or *false*. It's a statement used in logic. Ex: (age > 18) used as variable *x*.
- Typically true and false are usually denoted with 1 and 0 respectively.
- We have below a representation of an algebraic expression, formed of variables representing numbers and arithmetic operations:



- On the other hand a Boolean expression is represented as follows:



- In a Boolean expression the operations are logic operations and variables are conditions.
- In a Boolean Algebra there is no subtraction, unary minus (ex: -x), or division.
- Historically George Boole in the mid-19th century, laid the groundwork for symbolic logic that formulates and symbolizes human logic.
- Transforming for instance the logic sentence “Going to the army requires being a male and being 18 years old”, becoming in Boolean algebra $x \cdot y$. x is the condition “being a male”, and y “having 18 years old”.
- And later Claude Shannon in the mid-20th century in his master’s thesis showed how Boolean algebra could be used to analyze and synthesize Digital circuits.

- Boolean algebra is now the mathematical foundation for the design and building of all modern Digital Computers and electronic devices.

Remark 1: Claude Shannon is considered the second father of modern Computer Science for the implications of his works. Alan Turing was given the title of the father of modern Computer Science.

2. Basic Concepts and Definitions

2.1. Binary Variables (True/False, 1/0)

- In Boolean Algebra, variables can only have one of two possible values.
- Commonly represented as True/False, On/Off, High/Low, or most importantly, 1/0.

2.2. Literals

- A literal is defined as a single variable or its complement (ex: A or $\overline{A}$).
- The complement of a variable is the logical Not operation.

2.3. Boolean Expressions

- A Boolean expression is a combination of binary variables and Boolean operators.
- For example: $(x \cdot \overline{y}) + z$

2.4. Boolean functions

- Like Analytical functions, Boolean functions are defined with one or many variables that take Boolean values true or false (often as 1 or 0) as input. And produces a single Boolean value as output.
- Boolean functions are expressed by Boolean expressions.
- For example: $F(x, y) = x + \overline{y} \Rightarrow F(1, 1) = 1 + \overline{1} = 1 + 0 = 1$

2.5. Truth Table

- A Truth Table is a tabular representation that shows all possible input combinations for a Boolean expression or Boolean function.
- A Truth Table maps each input combination with its corresponding output.
- For example: the Truth Table of the Boolean expression $x + \overline{y}$ is:

x	y	$x + \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

3. Boolean Operations

- The main well-known 3 Boolean operations are And, Or, Not. But we can add other less-known operations like Nand, Nor, Xor, Xnor.

3.1. And operation

- Also known as the Conjunction operation.
- And is symbolized by a dot (·).
- Its Truth Table is:

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

3.2. Or operation

- Also known as the Disjunction operation.
- And is symbolized by plus (+).
- Its Truth Table is:

a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

3.3. Not operation

- Also known as the Inversion or Complement operation.
- And is symbolized by an overline (‾).
- Its Truth Table is:

a	$\overline{a}$
0	1
1	0

3.4. Nand operation

- Known as Not-And operation.
- Don't have a specific symbol. It is a composition, interpreted as the Not of an And operation. Ex: $\overline{x \cdot y}$
- Its Truth Table is the reverse of the And Truth Table:

a	b	$\overline{a \cdot b}$
0	0	1
0	1	1
1	0	1
1	1	0

3.5. Nor operation

- Known as Not-Or operation.
- Don't have a specific symbol. It is a composition, interpreted as the Not of an Or operation. Ex: $\overline{x+y}$
- Its Truth Table is the reverse of the Or Truth Table:

a	b	$\overline{a+b}$
0	0	1
0	1	0
1	0	0
1	1	0

3.6. Xor operation

- Known as Exclusive-Or operation. By contrast, the normal Or is called Inclusive-Or.
- The logic of Xor is to have the choice between 2 conditions, but they can't be both true at the same time.
- For instance in a vote between two candidates, you have to choose only one candidate at a time, you can't choose both.
- Is symbolized by a circled + ($\oplus$), and has the formula: $x \oplus y = (x \cdot \overline{y}) + (\overline{x} \cdot y)$
- Its Truth Table is:

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

3.7. Xnor operation

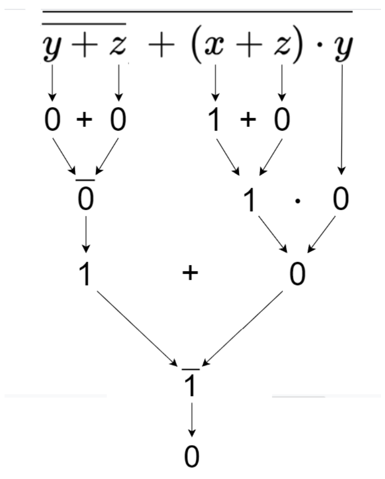
- Known as Not-Xor operation.
- Is symbolized by a circled - ($\ominus$)and has the formula: $x \ominus y = \overline{x \oplus y} = (x \cdot y) + (\overline{x} \cdot \overline{y})$
- Interpreted as the Not of an Xor operation.
- Its Truth Table is the reverse of the Xor Truth Table:

a	b	$a \ominus b$
0	0	1
0	1	0
1	0	0
1	1	1

Remark 2: Even though Or (+) and And (·) are semantically different from arithmetic addition and multiplication, there are some similarities in calculation, like 1+0 = 1, 0+0 = 0, or 1·0 = 0, 1·1 = 1…etc.

4. Boolean Expressions evaluation

- Similar to normal algebra, operations in Boolean algebra have what we call priority order or operator precedence.
- The priority order in Boolean algebra is: parenthesis first, followed by Not, And, Or, in descending order.
- For example: $x + \overline{y} \cdot z$, the first operation to be executed is the Not, then the And, and last the Or.
- Xor followed by Xnor are the last in the operator precedence. Nand and Nor are not included because they are composed of other operations.
- For example: $\overline{\overline{y+z} + (x+z)\cdot y}$ when $(x = 1, y = 0, z = 0)$



- A Truth Table lists the Boolean expression evaluation for all variable combinations:

x	y	z	$\overline{y+z}$	$(x+z)\cdot y$	$\overline{\overline{y+z} + (x+z)\cdot y}$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	1	0	0	1	0
1	1	1	0	1	0

Remark 3: You can see that the sub-expression $\overline{y+z}$ represents actually the Nor operation.

Remark 4: Truth Table by definition requires the input columns (x, y, z), and output column that contains the final result of the expression (the rightmost column). But not the intermediate columns in the middle, they are optional, used to simplify the calculation.

Remark 5: The order used in the input columns. Four 0s and four 1s in x column. Successive two 0s and two 1s in column y. Successive 0s, 1s in column z. Guarantees an ascending order of the binary values on 3-bits $(x\,y\,z)$. (000=0), (001=1), (010=2),...,(111=7).

5. Axioms and Theorems of Boolean Algebra

- Axioms and Theorems are fundamental rules that govern Boolean algebra, which are essential for simplification.
- The rules of Boolean Algebra are much like those of ordinary Algebra but are in some cases simpler.
- Axioms by definition are assumed to be correct and unprovable, they are basic foundations for the Algebra.
- On the other hand, all Theorems are provable using Axioms or other proved Theorems.
- The main characteristic of Axioms and Theorems in Boolean Algebra is that they are dual. The same Axiom/Theorem remains correct by interchanging 0s and 1s, along ($\cdot$) and (+).

5.1. Axioms

- Axioms are the basic definitions of variable values and operations.
- The table below lists the Axioms of Boolean Algebra:

Name	Axiom	reference	Dual	reference
Binary field	$X = 0$ if $X \neq 1$	A1	$X = 1$ if $X \neq 0$	A1’
Not	$\overline{0} = 1$	A2	$\overline{1} = 0$	A2’
And/Or	$0 \cdot 0 = 0$	A3	$1 + 1 = 1$	A3’
And/Or	$1 \cdot 1 = 1$	A4	$0 + 0 = 0$	A4’
And/Or	$0 \cdot 1 = 1 \cdot 0 = 0$	A5	$1 + 0 = 0 + 1 = 1$	A5’

<u>Remark 6</u>: Axioms are referenced from A1 to A5 and their Duals are referenced from A1’ to A5’. In the same manner Theorems will be referenced by T instead of A.

5.1. Theorems

- Theorems will be categorized into two parts, Theorems with one variable, and Theorems with many variables.
- The following table lists Theorems with one variable:

Name	Theorem	reference	Dual	reference
Identity	$X \cdot 1 = X$	T1	$X + 0 = X$	T1’
Null Element	$X \cdot 0 = 0$	T2	$X + 1 = 1$	T2’
Idempotency	$X \cdot X = X$	T3	$X + X = X$	T3’
Involution	$\overline{\overline{X}} = X$	T4	$X = \overline{\overline{X}}$	T4’
Complements	$X \cdot \overline{X} = 0$	T5	$X + \overline{X} = 1$	T5’

- Theorems with many variables are the must used, and are in part used in simplification.
- The following table lists Theorems with many variables:

Name	Theorem	reference	Dual	reference
Commutativity	$X \cdot Y = Y \cdot X$	T6	$X + Y = Y + X$	T6’
Associativity	$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$	T7	$(X + Y) + Z = X + (Y + Z)$	T7’
Distributivity	$X \cdot (Y+Z) = (X \cdot Y) + (X \cdot Z)$	T8	$X + (Y \cdot Z) = (X+Y) \cdot (X+Z)$	T8’
Covering	$X \cdot (X+Y) = X$	T9	$X + (X \cdot Y) = X$	T9’
Combining	$(X \cdot Y) + (X \cdot \overline{Y}) = X$	T10	$(X+Y) \cdot (X+\overline{Y}) = X$	T10’
De-Morgan’s Theorem	$\overline{X_0 \cdot X_1 \cdot X_2 \cdot \ldots} = \overline{X}_0 + \overline{X}_1 + \overline{X}_2 + \ldots$	T11	$\overline{X_0 + X_1 + X_2 + \ldots} = \overline{X}_0 \cdot \overline{X}_1 \cdot \overline{X}_2 \cdot \ldots$	T11’

- De-Morgan’s Theorem (T11) is a particularly powerful tool in Digital Design.

6. Boolean Functions Canonical Forms

- Canonical Forms (or Normal Forms) are standard ways of expressing Boolean Functions.
- A Canonical Form of a function makes a unique representation for similar functions, functions that produce the same results.
- The main purpose of using Canonical Forms is to be able to generate Boolean Functions from a Truth Tables.
- And second, to make Boolean Functions simplification and comparison easier.
- There are two Canonical Forms: Sum-of-Product form and Product-of-Sum form.

6.1. Sum-of-Product form (SoP)

- Also known as Disjunctive Normal Form.
- Defined as a Boolean Function where multiple product (AND) terms are summed (ORed) together.
- Each product term is called a *Minterm*, it is a product of all variables in either true or complemented form (A is the true form, $\overline{A}$ is the complement).
- For each 1 on the Truth Table, a Minterm is created.
- Each Minterm is shaped to evaluate to 1 while calculated by the corresponding inputs in the Truth Table.
- For example, we have to formulate the Sum-of-Product form function from the following Truth Table:

A	B	C	F(A, B, C)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

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

- We have 3 Minterms for each 1 in the Truth Table.
- We can for instance see how a Minterm is formed. The first one ($\overline{A} \cdot \overline{B} \cdot C$), A and B are complemented because their values (in the 2nd raw) are 0, C with 1 is not.

- The conceptual idea of the SoP form is to assure using Minterms. If one of the inputs (A,B,C) of the function producing 1 is entered, all of the ORed Minterms will produce 1. And we arrange the bars in a Minterms so that it produces 1.
- The rule becomes simple to form a Minterm, if the input of a literal is 0 the literal is complemented, and if it is 1 the literal is not complemented.
- Minterms are called this way because, if at least one Minterm is at 1, all the functions produce 1.
- The 2 following representations are also valid to write a SoP form:

$$F(A,B,C)=\sum\Big((\overline{A}\cdot\overline{B}\cdot C),(A\cdot\overline{B}\cdot C),(A\cdot B\cdot C)\Big)$$
$$F(A,B,C)=\sum(1,5,7)$$

6.2. Product-of-Sum form (PoS)

- Also known as Conjunctive Normal Form. And is the complement of SoP.
- Defined by multiple sum (OR) terms multiplied (ANDed) together.
- Each sum term is called a *Maxterm*, it is a sum of all variables in either true or complemented form.
- For each 0 on the Truth Table a Maxterm is created.
- Each Maxterm is shaped to evaluate to 0 while calculated by the corresponding inputs in the Truth Table.
- For example, we have to formulate the Product-of-Sum form function from the previous Truth Table:

A	B	C	F(A, B, C)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

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

- We have 5 Maxterms for each 0 in the Truth Table.
- We can for instance see how a Maxterm is formed. The second one ($A+\overline{B}+C$), B is complemented because its value (in the 3rd raw) is 1, A and C with 0 are not.
- The conceptual idea of PoS form is the complement of SoP, is to assure using Maxterms, if one of inputs (A,B,C) of the function producing 0 is entered, all the ANDed Maxterms will produce 0. And we arrange the bars in a Maxterms so that it produces 0.
- The rule to form a Maxterm is the complement of a Minterm. If the input of a literal is 1, the literal is complemented, and if it is 0, the literal is not complemented.
- Maxterm is called this way because, for the function to produce 1, all the Maxterms should be 1, and no one 0.
- The 2 following representations are also valid to write a PoS form:

$$F(A,B,C) = \prod\left((A+B+C),(A+\overline{B}+C),(A+\overline{B}+\overline{C}),(\overline{A}+B+C),(\overline{A}+\overline{B}+C)\right)$$
$$F(A,B,C) = \prod(0,2,3,4,6)$$

<u>Remark 7</u>: It is clearly obvious that the SoP $F(A,B,C)=\Sigma(1,5,7)$, and PoS $F(A,B,C)=\Pi(0,2,3,4,6)$ are complements. That means if you find one you can deduce the other.

7. Logic simplification

- We know the Boolean Functions and expressions are used to represent Digital Electronics.
- The process of simplification or reduction of those functions and expressions is reflected directly by the reduction of the size of the electronic circuit.
- The simplification or reduction of a function/expression means making it shorter. With fewer literals and operations.
- We have to study 3 methods of logic simplification: Algebraic Simplification (AS), Karnaugh map (KM), and Quine-McCluskey method (QM).

7.1. Algebraic Simplification (AS)

- Also known as Algebraic Minimization.
- Is a method to minimize Boolean Functions and expressions based on Boolean Algebra theorems.
- The Algebra theorems are used to transform and change the functions and expressions, trying to minimize them as short as possible.
- Algebra theorems are used in many strategies, like Factoring, Redundancy, Expanding to reduce,...etc.
- Let us try an example, reducing this expression: $ABC + \overline{A}BC + AB\overline{C}$

Expression	Theorems
$ABC + \overline{A}BC + AB\overline{C}$	
$(A+\overline{A})BC + AB\overline{C}$	T8
$(1)BC + AB\overline{C}$	T5’
$BC + AB\overline{C}$	T1

- We get a reduced solution $BC + AB\overline{C}$ by following the sequence of Theorems T8, T5’, T1.
- But using another developing path, it is possible to get a better solution, like:

Expression	Theorems
$ABC + \overline{A}BC + AB\overline{C}$	
$ABC + ABC + \overline{A}BC + AB\overline{C}$	T3’
$ABC + \overline{A}BC + ABC + AB\overline{C}$	T6’
$(A+\overline{A})BC + AB(C+\overline{C})$	T8
$(1)BC + AB(1)$	T5’
$BC + AB$	T1

- The better solution is $BC + AB$. This done by following the sequence: T3’, T6’, T8,T5’, T1.

- This demonstrates that many paths of Algebraic development are possible. And the best one is the one that leads to the shortest expression.

Remark 8: The And operation in the example above was implicit, no point (·) was used to express the And. It is like multiplication, the point can be omitted.

Remark 9: In the development above it was possible to use T10 directly, instead of using T8, T5'. T10 is one of the powerful theorems used for reduction.

7.2. Karnaugh map (KM)

- Contrary to Algebraic Simplification where many paths are explored. Karnaugh map is a systematic method, where the steps are fixed and well known.
- Karnaugh map is known as a visual method.
- Karnaugh map is basically a Truth Table shaped as a matrix form.
- The structure of the Karnaugh map is that the intersection of the row and the column gives the output of the function.
- Each cell in the map contains 1 or 0. A 1 represents a Minterm, and 0 represents a Maxterm.
- The matrix is configured in such a way every 2 neighbor cells, the term should change only by 1 literal. Ex: 2 neighbor Minterms $A\overline{B}C$ and ABC. Only B changes.
- For an example of 3 variables, we can see below how it is done:

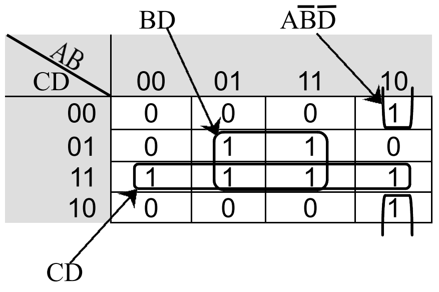
A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

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

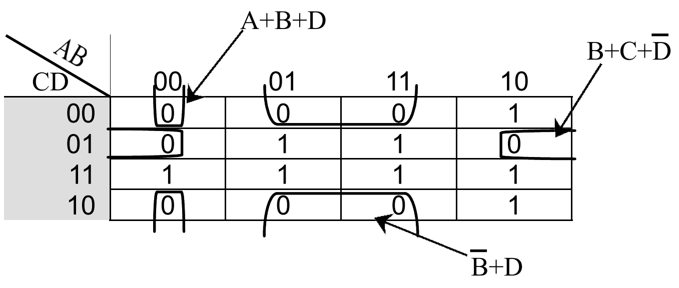
- AB are ordered 00, 01, 11, 10, and C ordered 0, 1. The sequence of AB is not normal, it is called Gray Code sequence. C also is on Gray Code.
- The key property of Gray Code is that to pass to the next number, only one variable should change.
- The Gray Code is about the order not the values.
- And the second clause is that the last number (10) also should check the same condition as with the first (00). Which makes the Karnaugh map circular.
- The circular map guarantees the 1 of (101) is the east neighbor of 1 (001).
- The Gray Code is the reason that makes only 1 literal change in any 2 neighbor Minterms (or 0s in Maxterms).
- This neighborhood characteristic is important. This is the heart of the minimization used in the Karnaugh map method.
- This characteristic applies Theorem T10 (or T10' for 0s, Maxterms) to perform reduction, $(X\cdot Y)+(X\cdot\overline{Y}) = X$.
- T10 Theorem indirectly stipulates that if one literal changes, it should be reduced.

- This is why Karnaugh map, boxes are created to include all 1 (or 0s for Maxterms).
- Those boxes represent many possibly intersected islands of 1s, where it is possible visually to apply reduction using the T1 theorem.
- The boxes should have a power of 2 dimensions (1, 2, 4, 8, 16....).
- Let's see an example on a 4 variable function.

A	B	C	D	F(A,B,C,D)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
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	0
1	1	1	1	1



- The rules for a Karnaugh map are that all the 1s should be covered by a box.
- And create the biggest box possible, even by intersecting other boxes.
- And last, to get the fewer boxes possible allowing to cover all 1s.
- You can see the circular nature of the map while creating the box (1000, 1010).
- The same rules apply for 0s, or Maxterms, but 0s instead of 1s are covered.
- Each box represents a term, we can see the resulting terms after reduction on the map above.
- The rule to reduce for a box is simple. Respecting the limits of the box you need to check the values of AB, which one changes horizontally. And CD which one changes vertically. The variable changing is deleted.
- The variable not changing inside the box limit can be with value 0, in this case, it is a complement variable, and with the value 1 are normal (not complement).
- For example: the box in the middle, with the term BD. We can see that the values of AB inside the box are (01,11). It is the A that changes from 0 to 1, and it is deleted. The value of B is 1, which implies the literal is not complemented. The same for CD=(01,11) inside the box, C is changing and must be deleted, and D=1 makes it not complemented. Which produces the term BD.
- The original function of the Truth Table is extracted using SoP:
  $\mathrm{F(A,B,C,D)} = \bar{\mathrm{A}}\bar{\mathrm{B}}\mathrm{CD} + \bar{\mathrm{A}}\mathrm{B}\bar{\mathrm{C}}\mathrm{D} + \bar{\mathrm{A}}\mathrm{BCD} + \mathrm{A}\bar{\mathrm{B}}\bar{\mathrm{C}}\bar{\mathrm{D}} + \mathrm{A}\bar{\mathrm{B}}\mathrm{C}\bar{\mathrm{D}} + \mathrm{A}\bar{\mathrm{B}}\mathrm{CD} + \mathrm{AB}\bar{\mathrm{C}}\mathrm{D} + \mathrm{ABCD}$
- The final result after the minimization with Karnaugh map is:
  $\mathrm{F(A,B,C,D)} = \mathrm{BD} + \mathrm{CD} + \mathrm{A}\bar{\mathrm{B}}\bar{\mathrm{D}}$
- Let's see how this Karnaugh map works with 0s, or Maxterms.
- Same rules for the boxes persist, but the terms are reversed, like in the Maxterms.
- Let us repeat the same previous example with Maxterms:



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

- A 0s box generates a Maxterm. The literals are deleted the same way, but the literals are complemented in reverse. If the value of the literal is 1 then it is complemented, if it is 0 then it is not.
- We get the reduced result using Maxterms:
  $F(A,B,C,D) = (\overline{B}+D)(A+B+D)(B+C+\overline{D})$

<u>Remark 10</u>: Students should be aware of this common mistake: $\overline{A}\,\overline{B} \neq \overline{AB}$ and $\overline{A}+\overline{B} \neq \overline{A+B}$.

7.3. Quine-McCluskey method (QM)

- One of the main flaws of Karnaugh map is its variables number limit. Works fine up to 4 variables, difficult with 5 and 6 variables, unpractical for more.
- This is where Quine-McCluskey method shines. The method can scale easily for more than 4 variables. And simpler to program.
- Quine-McCluskey method is also called the tabular method because it is organized in tables.
- Like Karnaugh map, Quine-McCluskey method mainly uses the T10 ($(X\cdot Y)+(X\cdot\overline{Y}) = X$) theorem to reduce terms.
- This method is practically based on looking for Minterms changing in only one variable, and congregate (assemble) them. In the same way as the concept of neighborhood in the Karnaugh map.
- When two Minterms are aggregated, they form what we call *Implicant*. <u>Ex</u>: $\overline{A}BCD$ and $ABCD$ produce the implicant $\text{-}BCD$.
- With the same way, two implicant are aggregated to eliminate another variable. <u>Ex</u>: $\text{-}B\overline{C}D$ and $\text{-}BCD$ produce $\text{-}B\text{-}D$.
- And the process of aggregation continues to go up in the hierarchy, eliminating more and more variables, until it becomes impossible to aggregate anymore.
- This is the first Stage of the Quine-McCluskey method, that aims to list all the implicants.
- Let us try an example for the first stage on our previous function:
  $F(A,B,C,D) = \overline{A}\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}\overline{C}\overline{D} + A\overline{B}C\overline{D} + A\overline{B}CD + AB\overline{C}D + ABCD$

Minterms	Pass 1	Pass 2
(8) 1000✓	(8,10) 10-0✗	(5,7,13,15) -1-1✗
(3) 0011✓	(3,7) 0-11✓	(3,7,11,15) --11✗
(5) 0101✓	(3,11) -011✓	~~(3,11,7,15) --11~~✗
(10) 1010✓	(5,7) 01-1✓	~~(5,13,7,15) -1-1~~✗
(7) 0111✓	(5,13) -101✓	
(11) 1011✓	(10,11) 101-✗	
(13) 1101✓	(7,15) -111✓	
(15) 1111✓	(11,15) 1-11✓	
	(13,15) 11-1✓	

- The first column contains Minterms. They are organized in groups to ease paired comparison. 1st group of Minterms with only one 1, 2nd with two 1s…etc.
- Knowing that we are looking for changes only on one variable, elements of the 1st group are compared with 2nd group, the 2nd with the 3rd, 3rd with 4th…etc.
- If we have a matching result, it is marked in the second column (pass 1). The deleted variable is marked by a dash (-) in the implicant.
- The groups in pass 1 columns are formed by the 2 previous compared groups. For example the 1st group in pass 1 is the group of matching from the 1st and 2nd groups of Minterms, the 2nd group is from the 2nd and 3rd of Minterms …etc.
- The column pass 2 contains the aggregation of implicants from pass 1, it represents the implicants with 2 deleted variables.
- Each implicants is denoted by the aggregated Minterms in parentheses. Like in (3,11) -011. 3,11 are the aggregated Minterms.
- Groups in pass 2 are organized differently, each group contains the matching of terms that their dash (-) is at a certain position. For instance, the 1st group contains matching of implicants in pass 1 with a dash in the first position, group 2 in pass 2 contains matching implicants with the dash in the 2nd position...etc.
- Starting from pass 2 it is possible to get similar results, like in the table above. They are simply deleted, like ~~(3,11,7,15) --11~~.
- We stop at pass 2 because it is no longer possible to aggregate implicants.
- It is observable from the table that some implicants are marked as ✓ and others as ✗. The latter are highlighted, like (3,7,11,15) --11.
- ✓ mark is used for implicant or Minterms aggregated in the next pass. ✗ is for those not aggregated, and they represent what we call Prime Implicants (PI).
- Knowing that aggregation is hierarchical, the Prime Implicants are implicants that all Minterms aggregate to.
- That makes the result of the first Stage of Quine-McCluskey method the PIs that cover all the Minterms (PI the are same as Karnaugh map islands).

<u>Remark 11</u>: A Minterm if it is not aggregated is also considered a Prime Implicant.

- The second stage of the Quine-McCluskey method is to choose the minimum number of Prime Implicants that cover all the Minterms.
- To do so, a matrix called the Prime Implicant table is used to show which Minterm is covered by which Prime Implicant:

Minterm Implicant	(3) 0011	(5) 0101	(7) 0111	(8) 1000	(10) 1010	(11) 1011	(13) 1101	(15) 1111
(5,7,13,15)-1-1		x	x				x	x
(3,7,11,15)--11	x		x			x		x
(8,10)10-0				x	x			
(10,11)101-					x	x		

- By intersection of the rows and the columns, it is possible to know which Minterm is covered by which Prime Implicant.
- We can see that some Minterms are covered only by one Implicant, they are solitary Minterms, like (3, 5, 8, 13). Their implicants are necessary otherwise not all Minterms are covered.
- Those Implicants covering the solitary Minterms are called Essential Prime Implicants (EPI). And are necessarily present in the final result.
- The list of Essential Prime Implicants is: $\text{EPI} = \{(5,7,13,15)\text{ -1-1},\ (3,7,11,15)\text{ --11},\ (8,10)\text{ 10-0}\}$.
- The next step is to list all the Minterms covered by Essential Prime Implicants: $\text{EPI}_{\text{Minterms}}\{3, 5, 7, 8, 10, 11, 13, 15\}$
- The next step is to list the left Minterms not covered by the Essential Prime Implicants: $\overline{\text{EPI}}_{\text{Minterms}}\{\}$ empty set.
- The rule is that, normally from the remaining non-covered Minterms, we need to choose the minimum number of non-EPI implincants to cover them.
- In our case EPI covers all Minterms and we don't need another PI.
- We are still following the same rule as in the Karnaugh map, to take the minimum number of islands to cover all the 1s.
- All EPIs are chosen for the final result, plus the Non EPIs that cover whatis left uncovered by EPIs.
- The final solution is to change the selected PIs to Minterms: $\text{-1-1} = BD,\ \text{--11} = CD,\ \text{10-0} = A\overline{B}\overline{D}$.
- Which results in the function: $F(A,B,C,D) = BD + CD + A\overline{B}\overline{D}$

<u>Remark 12</u>: You can see that Karnaugh map and Quine-McCluskey produce exactly the same result. Because they are the same, the first is visual, and the second is tabular.

- To well understand the final steps of the Quine-McCluskey method we need to imagine a hypothetical case where not all Minterms are covered by EPI.
- Let take the following Prime Implicants table as an example:

Mintrem Implicant	(3) 0011	(5) 0101	(7) 0111	(8) 1000	(10) 1010	(11) 1011	(13) 1101	(15) 1111
(5,7,13,15)-1-1	x	x	x	x			x	x
(3,7,11,15)--11	x		x			x	x	
(8,10)10-0				x	x			
(10,11)101-					x	x	x	

- The $\text{EPI} = \{(5,7,13,15)\text{ -1-1}\}$. Because (5, 15) the solitary Minterms are covered by only $(5,7,13,15)$ -1-1.

- The non Essential Prime Implicants is what left from the EPI:
  $$\mathrm{NEPI} = \{(3,7,11,15)\text{--}11,\ (8,10)10\text{-}0, (10,11)101\text{-}\}.$$
- The total Minterms covered by EPI is: $\mathrm{EPI}_{\mathrm{Minterms}}\{3, 5, 7, 8, 13, 15\}$.
- The Minterms not covered by EPI are: $\overline{\mathrm{EPI}}_{\mathrm{Minterms}}\{10, 11\}$.
- In this situation, we need to choose from NEPI the minimum number of PIs to cover all $\overline{\mathrm{EPI}}_{\mathrm{Minterms}}$. We can see from the table that $(10,11)101$- is a better fit.
- Then the solution is: $(5,7,13,15)$ -1-1 and $(10,11)101$-.
- We can calculate the reduced Minterms: -1-1 $= \mathrm{BD}$ and $101\text{-} = \mathrm{A\overline{B}C}$
- The reduced function is: $\mathrm{F(A,B,C,D) = BD + A\overline{B}D}$

<u>Remark 13</u>: It is obvious that the Quine-McCluskey method could perfectly be performed using Maxterms, with of course the respect of the rules related to Maxterms.