Ecole Supérieure Polytechnique Internationale Privée de Sfax

# Boolean Algebra and Logic Circuits

**Dr. Eng. Yasmine KOUBAA**
PhD in electrical engineering

Academic year 2023-204

## Introduction

- Computers can process and store data, perform calculations, and execute various tasks through a programmed set of instructions.
- Computers are used in a wide range of applications, from personal computing and gaming to scientific research, business operations, and more.

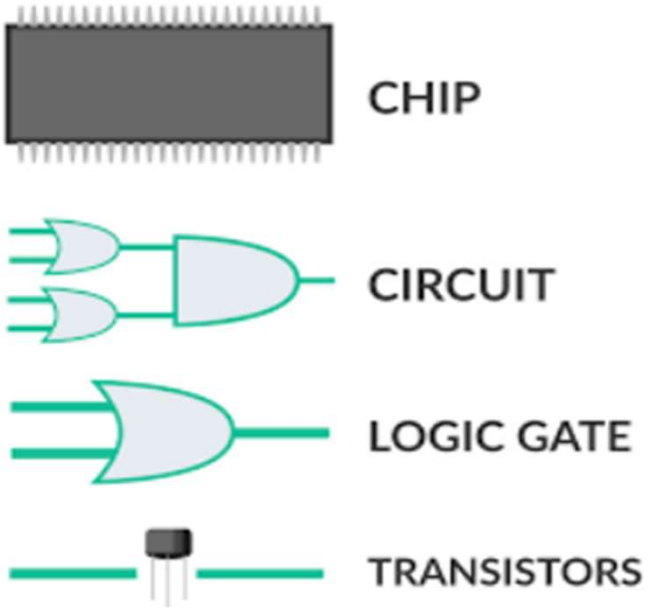
Smartphone

Tablet

Laptop

- Computers can process large amounts of data and do a great amount of computation very quickly

## Introduction




- An electronic circuit used in computers performs a logical operation on its two or more input signals.

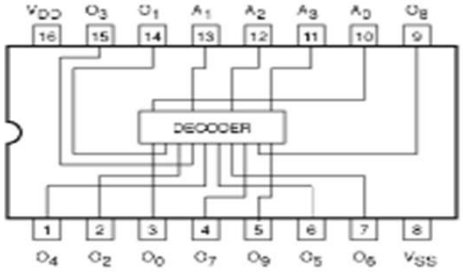
- Digital signals are processed by the digital system which can be built with various logic gates.
- These logic circuits are made of various logic gates , by connecting them in certain combinations, in order to produce the required output.

.

- Circuits enable **computers** to do more complex operations than they could accomplish with just a single gate. The smallest circuit is a chain of 2 **logic gates**

## Introduction

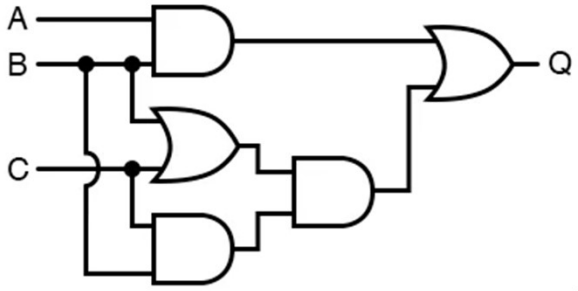
- Digital logic circuits are mainly classified into two types , sequential logic circuits and combinational logic circuits.
- **Combinational Logic Circuits** are made up from basic logic NAND, NOR or NOT gates that are “combined” or connected together to produce more complicated switching circuits. These logic gates are the building blocks of combinational logic circuits.
- An example of a combinational circuit is a decoder, which converts the binary code data present at its input into a number of different output lines, one at a time producing an equivalent decimal code at its output.



## Introduction

The three main ways of specifying the function of a **combinational logic circuit** are:

1. **Boolean Algebra** : This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False that results in a logic “1” output.
2. **Truth Table** : A truth table defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible combination of input variable that the gate could encounter.
3. **Logic Diagram** : This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol, that implements the logic circuit.



## Boolean algebra

- 1854: George Boole developed an algebraic system now called Boolean algebra.
- 1904: E. V. Huntington formulated a set of postulates that formally define the Boolean algebra
- 1938: C. E. Shannon introduced a two-valued Boolean algebra called switching algebra that represented the properties of bistable electrical switching circuits



## Boolean algebra

- Two valued Boolean algebra is called switching algebra
- A set of two value B={0,1}
- The basic operations are : AND, OR and NOT
- The **AND** operator is denoted by a dot (.)

  *x.y or xy is read : x AND y
- The **OR** operator is denoted by a plus (+)

  * x+y is read: x OR y
- The **NOT** operator is denoted (‘) or an overbar ( ¯ )

  $x'$ or $\bar{x}$ is the complement of $x$

## Boolean algebra

**Why we learn Boolean algebra ?**

- The objective is to learn how to design digital circuits
- These circuits use signals with two possible values
  - Logic 0 is a low voltage signal (around 0 volts)
  - Logic 1 is a high voltage signal (e.g. 5 or 3.3 volts)
- Having only two logic values (0 and 1) simplifies the implementation of the digital circuit

## Boolean algebra

### Postulates of Boolean Algebra

1. Closure: the result of any Boolean operation is in B = {0, 1}
2. Identity element with respect to + is 0: $x + 0 = 0 + x = x$

   Identity element with respect to · is 1: $x \cdot 1 = 1 \cdot x = x$
3. Commutative with respect to +: $x + y = y + x$

   Commutative with respect to ·: $x \cdot y = y \cdot x$
4. · is distributive over +: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$

   + is distributive over ·: $x + (y \cdot z) = (x + y) \cdot (x + z)$
5. For every $x$ in B, there exists $x'$ in B (called complement of $x$) such that: $x + x' = 1$ and $x \cdot x' = 0$

## Boolean algebra

### Boolean operations

- The following tables define $x \cdot y$, $x + y$, and $x'$
- $x \cdot y$ is the **AND** operator
- $x + y$ is the **OR** operator
- $x'$ is the **NOT** operator

| *x* | *y* | *xy* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| $x.x = x$ |
|---|
| $x.0 = 0$ |
| $x.1 = x$ |
| $x.\bar{x} = 0$ |

| *x* | *y* | *x+y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| $x + x = x$ |
|---|
| $x + 0 = x$ |
| $x + 1 = 1$ |
| $x + \bar{x} = 1$ |

| x | x'ou $\bar{x}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

## Boolean algebra

### Boolean Functions

- Boolean functions are described by expressions that consist of :
  - Boolean variables, such as: $x$, $y$, etc.
  - Boolean constants: 0 and 1
  - Boolean operators: AND (·), OR (+), NOT (‘, $\bar{}$ )
  - Parentheses, which can be nested
- Operator precedence: to avoid ambiguity in expressions
  - Expressions within parentheses should be evaluated first
  - The NOT (‘, $\bar{}$ ) operator should be evaluated second
  - The AND (·) operator should be evaluated third
  - The OR (+) operator should be evaluated last

## Truth table

- A truth table can represent a Boolean function
- List all possible combinations of 0's and 1's assigned to variables
- If n variables then $2^n$ rows
- Example: Truth table for $f = xy' + x'z$

$$2^n = 2^3 = 8\ rows$$

| x | y | z | y’ | xy’ | x’ | x’z | f= xy’+x’z |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

## DeMorgan's Theorem

DeMorgan's Theorem represents two of the most important rules of Boolean algebra:

- $(x + y)' = x'\,y'$
- $(x\,y)' = x' + y'$

It can be verified using a truth table

| X | y | x' | y' | x+y | (x+y)' | x'y' | xy | (xy)' | x'+y' |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

Identical

Identical

Generalized DeMorgan's Theorem:

- $x_1 + x_2 + \cdots + x_n{}' = x_1{}' \cdot x_2{}' \cdot \cdots \cdot x_n{}'$
- $x_1 \cdot x_2 \cdot \cdots \cdot x_n{}' = x_1{}' + x_2{}' + \cdots + x_n'$

## Complementing Boolean Functions

- What is the complement of $f = x'yz' + xy'z'$ ?
- Use DeMorgan's Theorem:
  - Complement each variable and constant
  - Interchange AND and OR operators
- So, what is the complement of $f = x'yz' + xy'z'$ ? Answer: $f' = (x + y' + z)(x' + y + z)$

- Example 2: Complement $g = (a' + bc)d' + e$
- Answer: $g' = (a(b' + c') + d)e'$

## Algebraic Manipulation of Expressions

- The objective is to acquire skills in manipulating Boolean expressions, to transform them into **simpler form**.
- **Example 1**: prove $x + xy = x$
- Proof: $x + xy = x \cdot 1 + xy$
  $= x \cdot (1 + y) = x \cdot 1 = x$

## Boolean algebra

### Duality Principle

- The dual of a Boolean expression can be obtained by:
  - Interchanging AND (·) and OR (+) operators
- - Interchanging 0's and 1's
- Example: the dual of $x(y + z')$ is $x + yz'$
  - The complement operator does not change
- The properties of Boolean algebra appear in dual pairs
  - If a property is proven to be true then its dual is also true

| | Property | Dual Property |
|---|---|---|
| Identity | $x + 0 = x$ | $x.1 = x$ |
| Complement | $x + x' = 1$ | $x.x' = 0$ |
| Distributive | $x(y + z) = xy + xz$ | $x + yz = (x + y)(x + z)$ |

## Summary of Boolean Algebra

| | Property | Dual Property |
|---|---|---|
| Identity | $x + 0 = x$ | $x \cdot 1 = x$ |
| Complement | $x + x' = 1$ | $x \cdot x' = 0$ |
| Null | $x + 1 = 1$ | $x \cdot 0 = 0$ |
| Idempotence | $x + x = x$ | $x \cdot x = x$ |
| Involution | $(x')' = x$ | |
| Commutative | $x + y = y + x$ | $x\,y = y\,x$ |
| Associative | $(x + y) + z = x + (y + z)$ | $(x\,y)\,z = x\,(y\,z)$ |
| Distributive | $x\,(y + z) = xy + xz$ | $x + yz = (x + y)(x + z)$ |
| Absorption | $x + xy = x$ | $x(x + y) = x$ |
| Simplification | $x + x'y = x + y$ | $x(x' + y) = xy$ |
| De Morgan | $(x + y)' = x'\,y'$ | $(x\,y)' = x' + y'$ |

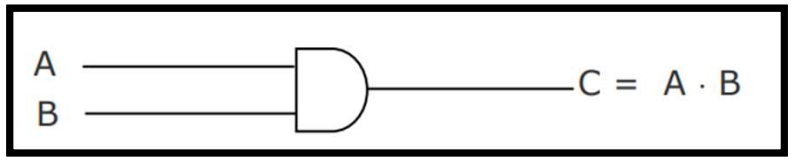
## Logic gates and symbols

- Logic gates are electronic circuits that operate on one or more input signals to produce standard output signals,
- Logic gates are the building blocks of all circuits in a computer
- Some of the more basic and useful logic gates are: AND, OR, NOT, NAND, NOR.

### AND gate

- Physical realization of logical multiplication (AND) operation
- Generates an output signal of 1 only if all input signals are also 1

Block diagrame symbol




Truth table

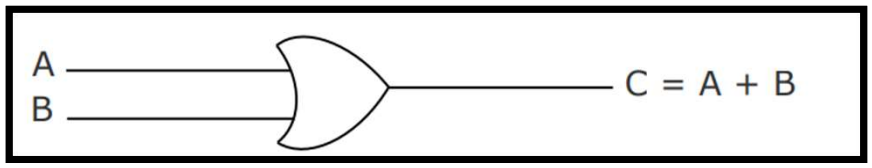
| Input | | output |
|---|---|---|
| A | B | C=A.B |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## Logic gates and symbols

### OR gate

- Physical realisation of a logical addition (OR) operation
- Generates an output signal of 1 if at least one of the input signal is also 1

Block diagrame symbol




Truth table

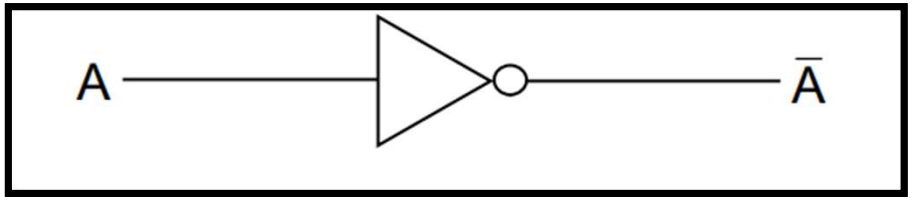
| Input | | output |
|---|---|---|
| **A** | **B** | **C=A+B** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## Logic gates and symbols

**NOT gate**

- Physical realisation of complementation operation
- Generates an output signal, which is the reverse of input signal

Block diagrame symbol




Truth table

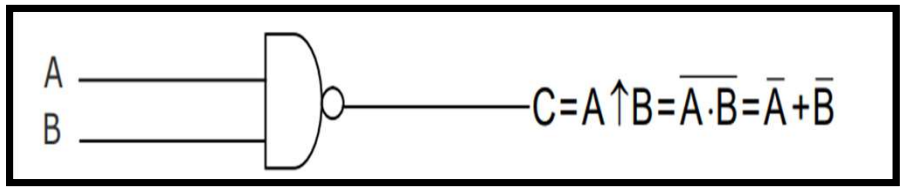
| Input | Output |
|---|---|
| $A$ | $\bar{A}$ |
| 0 | 1 |
| 1 | 0 |

## Logic gates and symbols

### NAND gate

- Complemented of AND gate
- Generates an output signal of:
  - 1 if any one of the input is 0
  - 0 when all the inputs are 1

Block diagrame symbol




Truth table

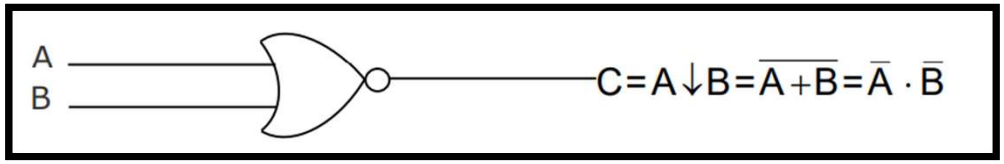
| Input | | output |
|---|---|---|
| **A** | **B** | $C = \bar{A} + \bar{B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## Logic gates and symbols

### NOR gate

- Complemented of OR gate
- Generates an output signal of:
  - 1 when all inputs are 0
  - 0 if any one of the inputs is a 1

Block diagrame symbol




Truth table

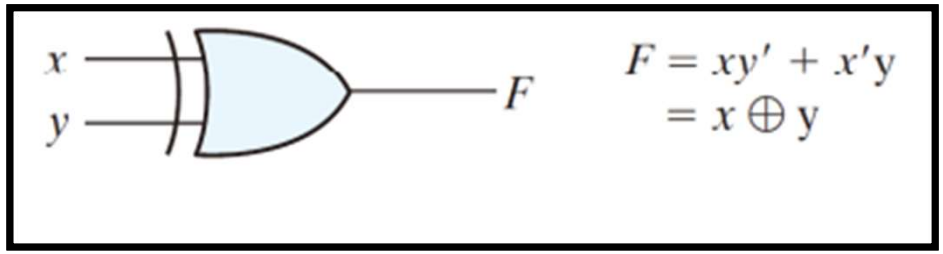
| Input | | output |
|---|---|---|
| **A** | **B** | $C = \overline{A}.\overline{B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## Logic gates and symbols

Exclusive OR (XOR)

- Exclusive OR (XOR) is an important Boolean operation used extensively in logic circuits

Block diagrame symbol




Truth table

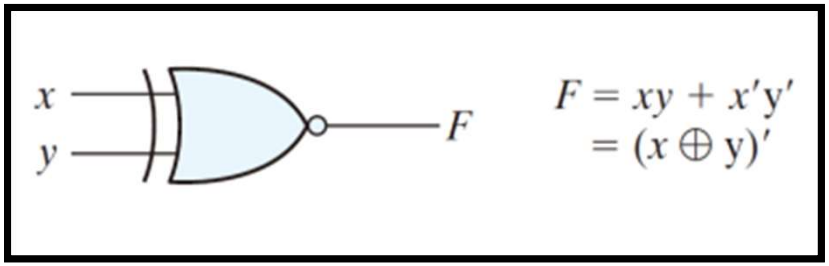
| Input | | output |
|---|---|---|
| **x** | **y** | ***F*** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## Logic gates and symbols

### Exclusive NOR (XNOR)

- Exclusive NOR (XNOR) is the complement of XOR

Block diagrame symbol




Truth table

| Input | | output |
|---|---|---|
| **x** | **y** | ***F*** |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## Logic gates and symbols

- The XOR function is: $x \oplus y = xy' + x'y$
- The XNOR function is: $(x \oplus y)' = xy + x'y'$
- XOR and XNOR gates are complex
  - Can be implemented as a true gate, or by
  - Interconnecting other gate types
- Uses for XOR and XNOR gates include:
  - Adders, subtractors, multipliers, counters, incrementers, decrementers
  - Parity generators and checkers

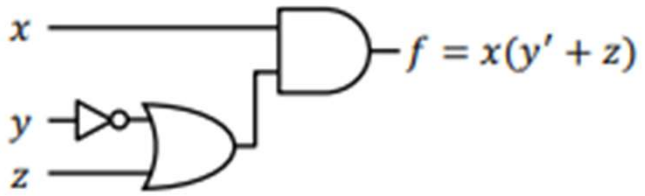
## Truth table and logic Diagram

- Given the following logic function : *f=x(y'+z)*
- Draw the corresponding truth table and logic diagram

**Truth Table**

| x y z | y'+z | f = x(y'+z) |
|---|---|---|
| 0 0 0 | 1 | 0 |
| 0 0 1 | 1 | 0 |
| 0 1 0 | 0 | 0 |
| 0 1 1 | 1 | 0 |
| 1 0 0 | 1 | 1 |
| 1 0 1 | 1 | 1 |
| 1 1 0 | 0 | 0 |
| 1 1 1 | 1 | 1 |

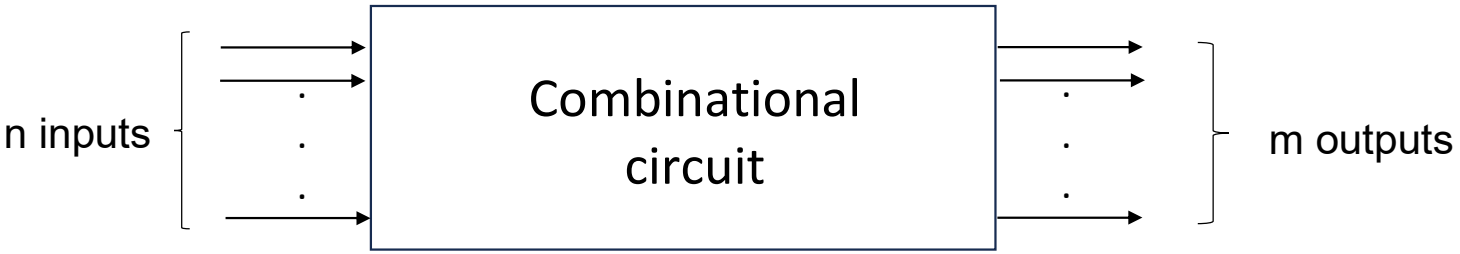
**Logic Diagram**




Truth Table and Logic Diagram describe the same function $f$. Truth table is unique, but logic expression and logic diagram are not. This gives flexibility in implementing logic functions

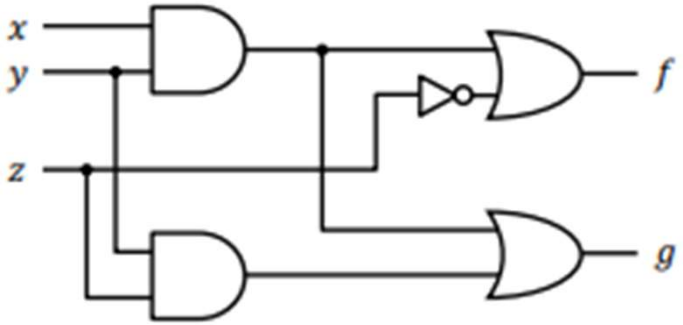
## Combinational circuit

- A combinational circuit is a block of logic gates having: $n$ inputs: $x_1$ , $x_2$ , … , $x_n$ and $m$ outputs: $f_1$ , $f_2$ , … , $f_m$
- Each output is a function of the input variables
- Each output is determined from present combination of inputs
- Combination circuit performs operation specified by logic gates



## Combinational circuit

**Example of a simple combinational circuit**




- The above circuit has:
  - Three inputs: $x$, $y$, and $z$
  - Two outputs: $f$ and $g$
- What are the logic expressions of $f$ and $g$ ?
- Answer: $f = xy + z'$

  $g = xy + yz$

## From turth tables to gate implementation

- Given the truth table of a boolean function f, how do we implement the truth table using logic gate ?

| x | y | z | f |
|---|---|---|---|
| 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 |

What is the logic expression of f ?

What is the gate implementation of f ?

To answer these quetions we need to:

Represent boolean functions : Canonical form

Define Minterms and Maxterms

## Representation of Boolean function

### Minterms and Maxterms

- n-binary variables have $2^n$ possible combinations
- Minterms are **AND** terms with every variable present in either true or complement form,
- Maxterms are **OR** terms with every variable present in either true or complement form

- For n variables, there are $2^n$ Minterms and Maxterms

- Minterms and Maxterms are designated with an index
- The index for the Minterm or Maxterm, expressed as a binary number, is used to determine whether the variable is shown in the true or complemented form

## Representation of Boolean function

### Minterms and Maxterms

- For Minterms:
  - '1' means the variable is Not Complemented
  - '0' means the variable is Complemented
- For Maxterms:
  - '0' means the variable is Not Complemented
  - '1' means the variable is Complemented

Minterms and Maxterms for 2 variables $x$ and y

| x | y | index | Minterms | Maxterms |
|---|---|---|---|---|
| 0 | 0 | 0 | $m_0 = x'y'$ | $M_0 = x + y$ |
| 0 | 1 | 1 | $m_1 = x'y$ | $M_1 = x + y'$ |
| 1 | 0 | 2 | $m_2 = xy'$ | $M_2 = x' + y$ |
| 1 | 1 | 3 | $m_3 = xy$ | $M_3 = x' + y'$ |

## Representation of Boolean function

**Sum-Of-Product (SOP) Canonical Form**

### Truth table

| x | y | z | f | Minterm |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 1 | $m_2 = x'yz'$ |
| 0 | 1 | 1 | 1 | $m_3 = x'yz$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | $m_5 = xy'z$ |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 1 | $m_7 = xyz$ |

- For the Sum of Product, the function output must be 1

  Focus on the '1' entries

  $f = m2 + m3 + m5 + m7$

  $f = \sum(2, 3, 5, 7)$

  $f = x'yz' + x'yz + xy'z + xyz$

## Representation of Boolean function

### Examples of Sum-Of-Product

- $f\ (a, b, c, d) = (2, 3, 6, 10, 11)$
- $f\ (a, b, c, d) = m2 + m3 + m6 + m10 + m11$
- $f\ (a, b, c, d) = a\,'b\,'\,cd\,' + a\,'b\,'\,cd + a'bcd' + ab'cd' + ab'cd$
- $g\ (a, b, c, d) = (0, 1, 12, 15)$
- $g\ (a, b, c, d) = m0 + m1 + m12 + m15$
- $g(\ a, b, c, d) = a\,'b\,'\,c'd\,' + a\,'b\,'\,c'd + abc'd' + abc$d

## Representation of Boolean function

### Product-Of-Sum (POS) Canonical Form

#### Truth table

| x | y | z | f | Maxterm |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $M_0 = x + y + z$ |
| 0 | 0 | 1 | 0 | $M_1 = x + y + z'$ |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | $M_4 = x' + y + z$ |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | $M_6 = x' + y' + z$ |
| 1 | 1 | 1 | 1 | |

- For the Product of Sum, the function output must be 0

Focus on the '0' entries

$$f = M_0 . M_1 . M_4 . M_6$$

$$f = \prod(0,1,4,6)$$

$$f = (x + y + z)(x + y + z')(x' + y + z)(x' + y' + z)$$

## Representation of Boolean function

### Examples of Product-Of-Sum

$f\ a, b, c, d = \prod (1, 3, 11)$

$f(a, b, c, d) = M1 \cdot M3 \cdot M11$

$f(a, b, c, d) = (a + b + c + d\ ')(\ a + b + c\ ' + d\ ')\ (a\ ' + b + c\ ' + d\ ')$

$g(\ a, b, c, d) = \prod (0, 5, 13)$

$g(a, b, c, d) = M0 \cdot M5 \cdot M13$

$\mathrm{g}(a, b, c, d) = (a + b + c + d)(\ a + b\ ' + c + d\ ')(a\ ' + b' + c + d\ ')$

## Conversions between Canonical Forms

- The same Boolean function $f$ can be expressed in two ways:
  - Sum-of-Product $f = m0 + m2 + m3 + m5 + m7 = \sum(0, 2, 3, 5, 7)$
  - Product-of-Sum $f = M1 \cdot M4 \cdot M6 = \prod(1, 4, 6)$

### Truth table

| x | y | z | f | Minterms | Maxterms |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | $m_0 = x'y'z'$ | |
| 0 | 0 | 1 | 0 | | $M_1 = x + y + z'$ |
| 0 | 1 | 0 | 1 | $m_2 = x'yz'$ | |
| 0 | 1 | 1 | 1 | $m_3 = x'yz$ | |
| 1 | 0 | 0 | 0 | | $M_4 = x' + y + z$ |
| 1 | 0 | 1 | 1 | $m_5 = xy'z$ | |
| 1 | 1 | 0 | 0 | | $M_6 = x' + y' + z$ |
| 1 | 1 | 1 | 1 | $m_7 = xyz$ | |

To convert from one canonical form to another, interchange the symbols $\sum$ and $\prod$ and list those numbers missing from the original form.

## Function Complement

### Truth table

| x | y | z | f | f’ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Given a Boolean function $f$

$$f(x,y,z) = \sum(0,2,3,5,7) = \prod(1,4,6)$$

Then, the complement $f'$ of function $f$

$$f'(x,y,z) = \prod(0,2,3,5,7) = \sum(1,4,6)$$

The complement of a function expressed by a Sum of Minterms is the Product of Maxterms with the same indices. Interchange the symbols $\sum$ and $\prod$, but keep the same list of indices

## Summary

- There are $2^n$Minterms and Maxterms for Boolean functions with n variables, indexed from 0 to$2^{n-1}$
- Minterms correspond to the 1-entries of the function
- Maxterms correspond to the 0-entries of the function
- Any Boolean function can be expressed as a Sum-of-Product and as a Product-of-Sum
- For a Boolean function, given the list of Minterm indices one can determine the list of Maxterms indices (and vice versa)
- The complement of a Sum-of-Minterms is a Product-of-Maxterms with the same indices (and vice versa)