Introduction to Basic Logic Circuits.

Subject: Machine Structure 2

Subject Content:

- **Chapter 1: Introduction.**
- **Chapter 2: Combinatorial Logic.**
- **Chapter 3: Sequential Logic.**
- **Chapter 4: Integrated Circuits.**

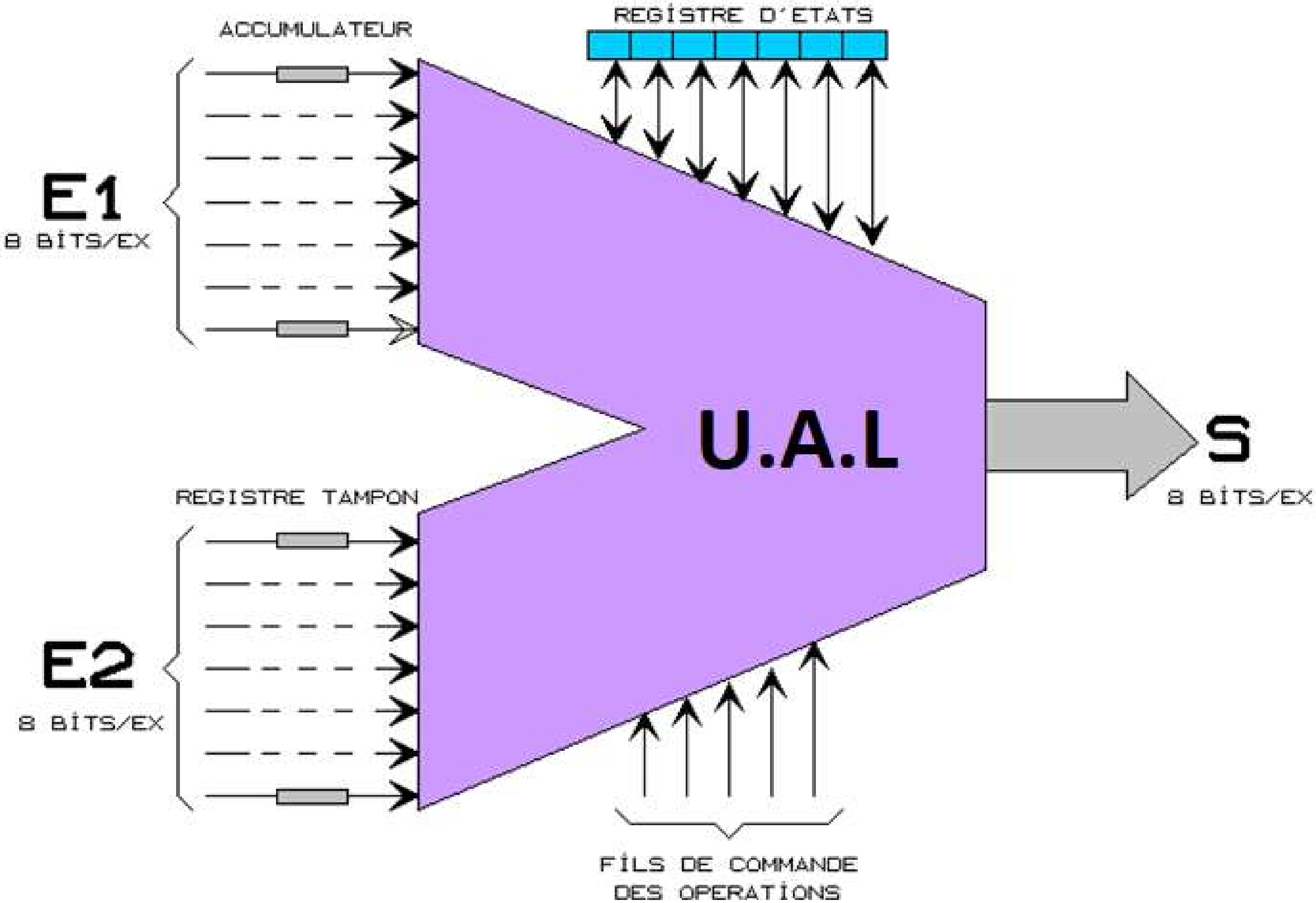
- **Assessment Method: Exam (60%),**
- **Continuous assessment (40%).**

Introduction

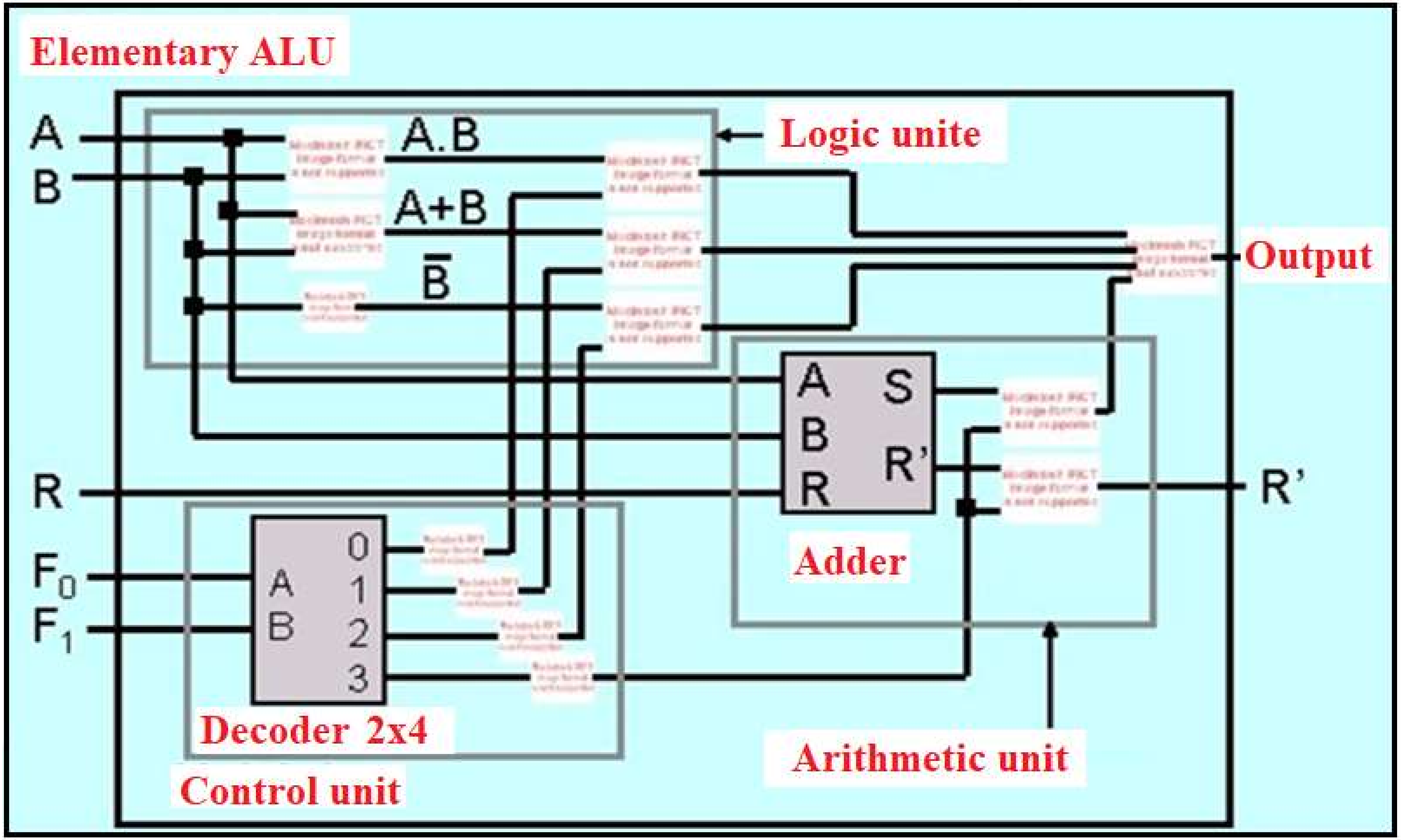
- Every computer is designed using integrated circuits, each with a specialized function:
- (Arithmetic and Logic Unit (ALU),
- Memory,
- Instruction decoding circuit, etc.).

These circuits are made up of logic circuits whose purpose is to perform operations on logical variables (binary).

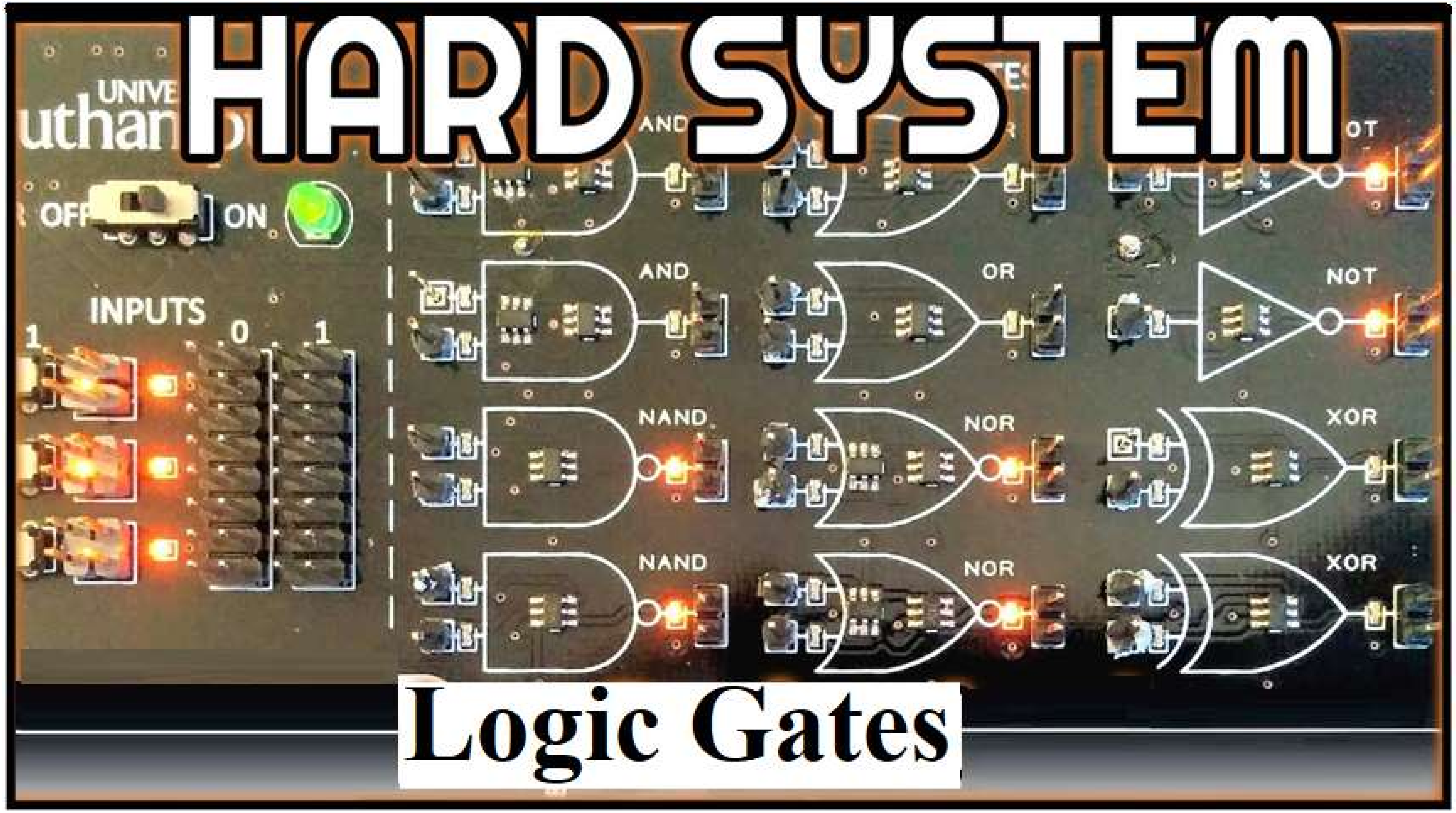
Introduction



Introduction



Introduction



Introduction

Logic circuits are constructed from
Electronic components,
such as transistors.

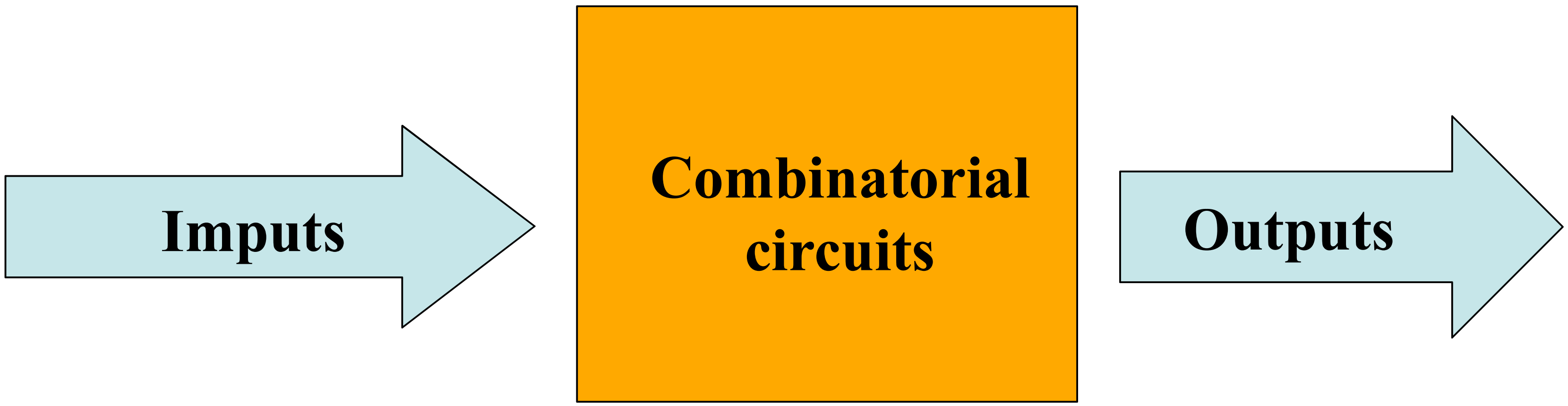
Types of logic circuits:

Combinatorial
Sequential

Combinatorial circuits

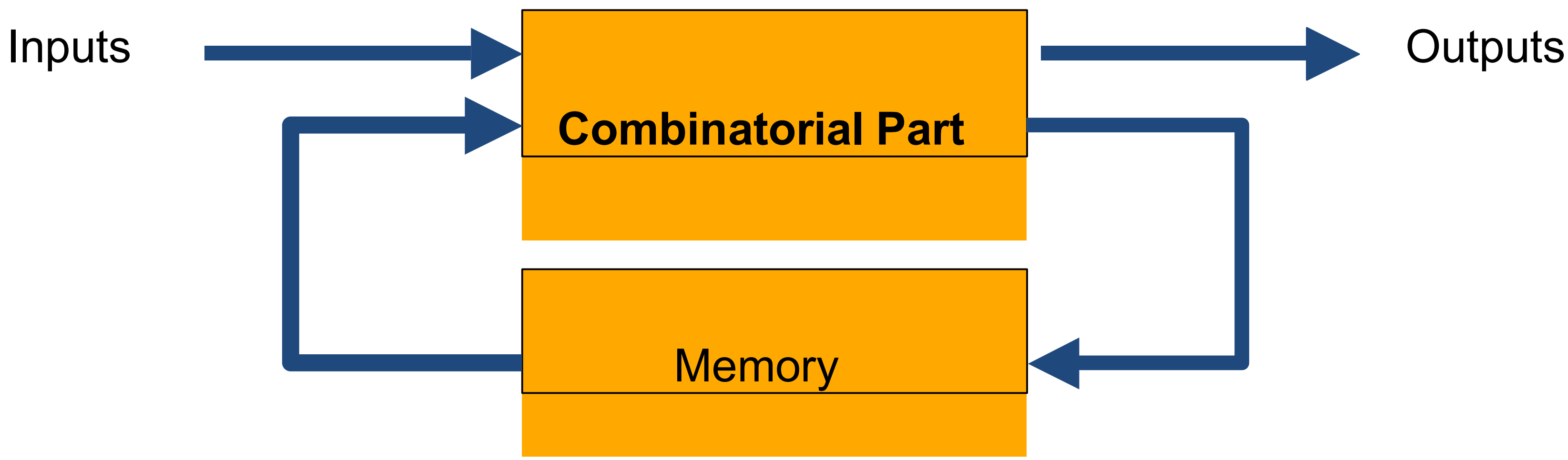
Theoretical foundation → Boolean algebra
The output functions are expressed in logical expressions of only the input variables.

A combinatorial circuit is defined by one or more logical functions.



Sequential Circuits or Memory Circuits

- Theoretical Basis – FSM (Finite State Machine)
- The output functions depend not only on the current state of input variables but also on the previous state of certain output variables (memory properties).



Reminder: Boolean Variables

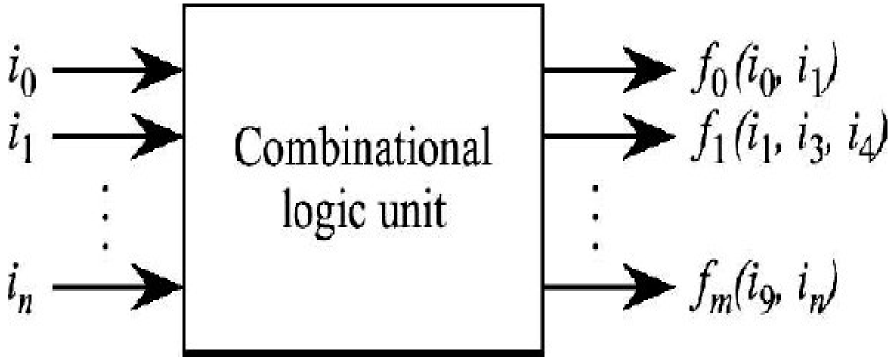
- A binary system is a system that can only exist in two permitted states.
- Various notations can be used to represent these two states:
  - ✓ Numeric: 1 and 0
  - ✓ Logical: true and false
  - ✓ Electronic: ON and OFF, high and low
  - ✓ A logical variable is a variable that can take two states or values: true (T) or false (F).

By associating T with the binary digit 1 and F with the binary digit 0, this type of variable becomes a Boolean or binary variable.

Combinatorial Circuits

- A combinational circuit is defined when its number of inputs, number of outputs, and the state of each output based on the inputs have been specified.
- This information is provided through a truth table.
- The truth table of a function with n variables has 2^n rows - input states.
- Boolean algebra and logical functions form the theoretical basis of combinationals circuits.

Truth tables



i_0	i_1	$F_0(i_0, i_1)$
0	0	
0	1	
1	0	
1	1	

i_1 i_3 i_4	$F_1(i_1, i_3, i_4)$
0 0 0	
0 0 1	
0 1 0	
0 1 1	
. . .	
1 1 1	

i_0 i_1 i_2 . . . i_n	$F_0(i_0, i_1)$	$F_1(i_1, i_3, i_4)$	. . .	$F_m(i_9, i_n)$
0 0 0 … 0				
0 0 0 … 1				
. . .				
. . .				
1 1 1… 1				

logic Gates

- In electronics, the two states of a Boolean variable are associated with two voltage levels:
- V(0) and V(1) for states 0 and 1, respectively.

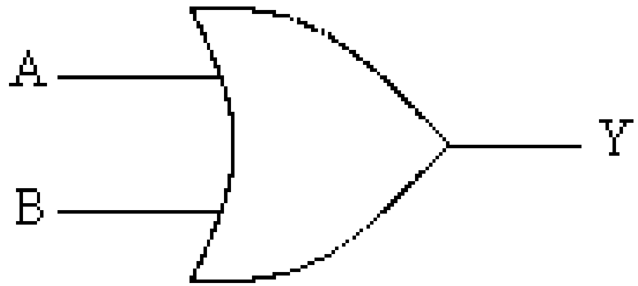
Level	**Positive Logic**	**Negative Logic**
High	1	0
Low	0	1

- Any logical function can be implemented using a set of basic logical functions called gates. A circuit is represented by a logic diagram.

logic Gate OR

- At least two inputs.
- The output of an OR function is in state 1 if at least one of its inputs is in state 1.

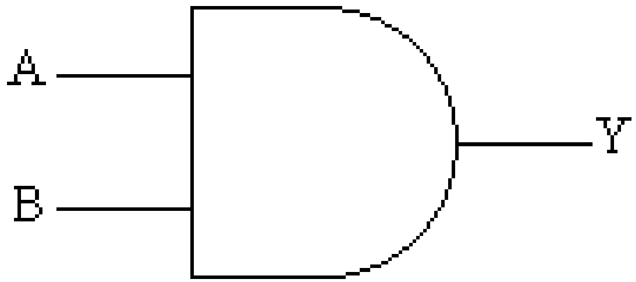
A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1



logic Gate AND

- At least two inputs.
- The output of an AND function is in state 1 if and only if all of its inputs are in state 1.

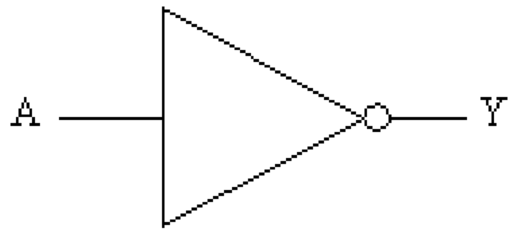
A	B	Y = A • B
0	0	0
0	1	0
1	0	0
1	1	1



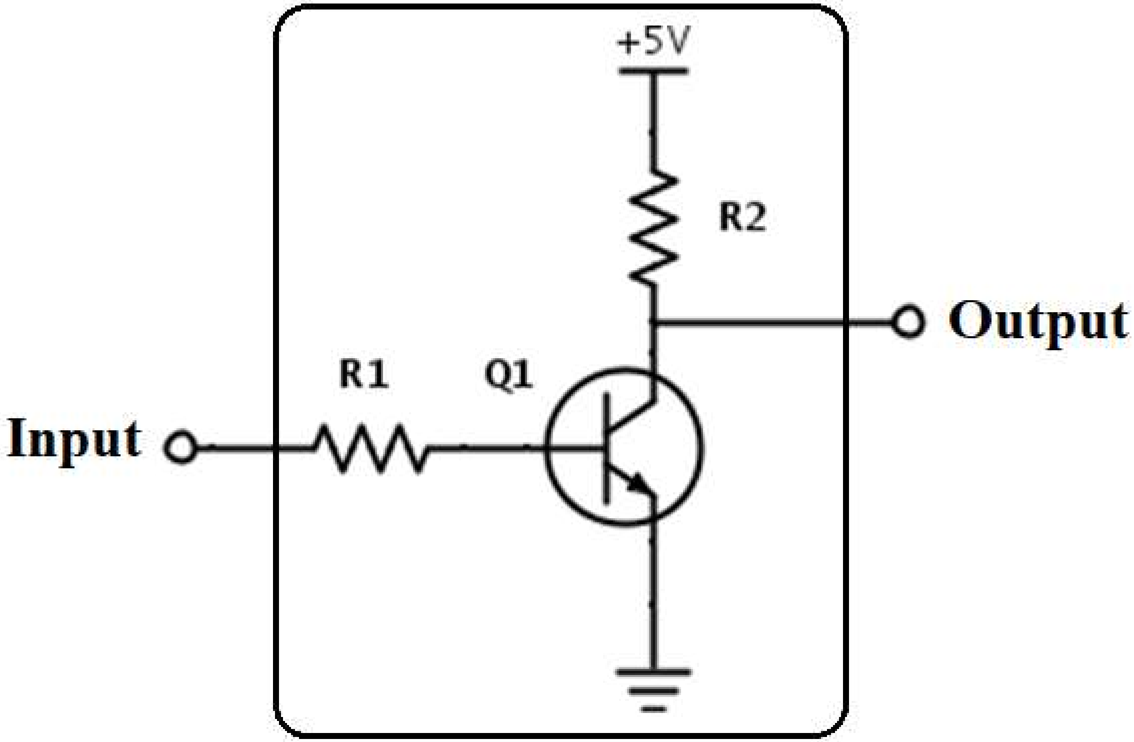
logic Gate NOT

- Single input and single output.
- The output of a NOT function is in state 1 if and only if its input is in state 0.

A	$Y = \overline{A}$
0	1
1	0



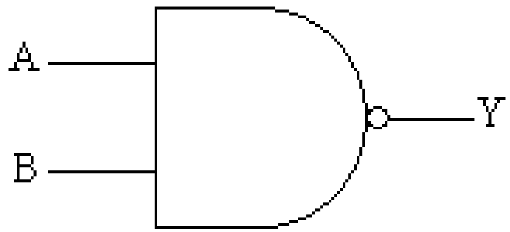
The "NOT" gate has only one input and one output. It simply inverts the signal: if the input signal is HIGH, the output signal is LOW. If the input signal is LOW, then the output signal is HIGH.



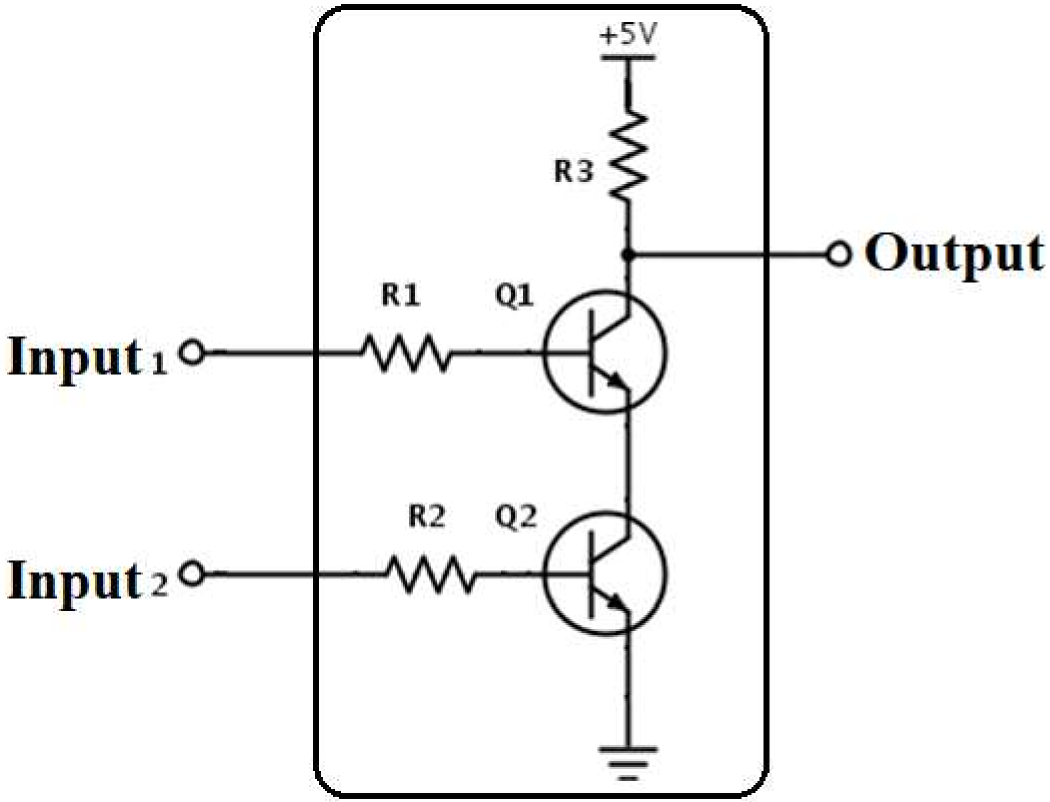
logic Gate NOT AND (NAND)

- Is formed by an inverter at the output of an AND gate.

A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0



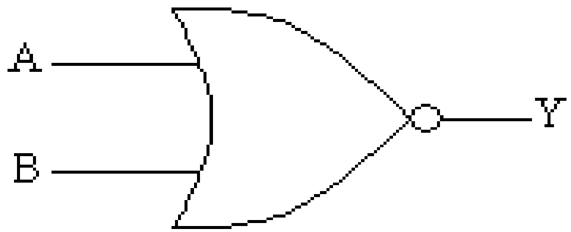
The NAND gate does exactly the opposite of an AND gate, so its output is low only if all of its inputs are high.



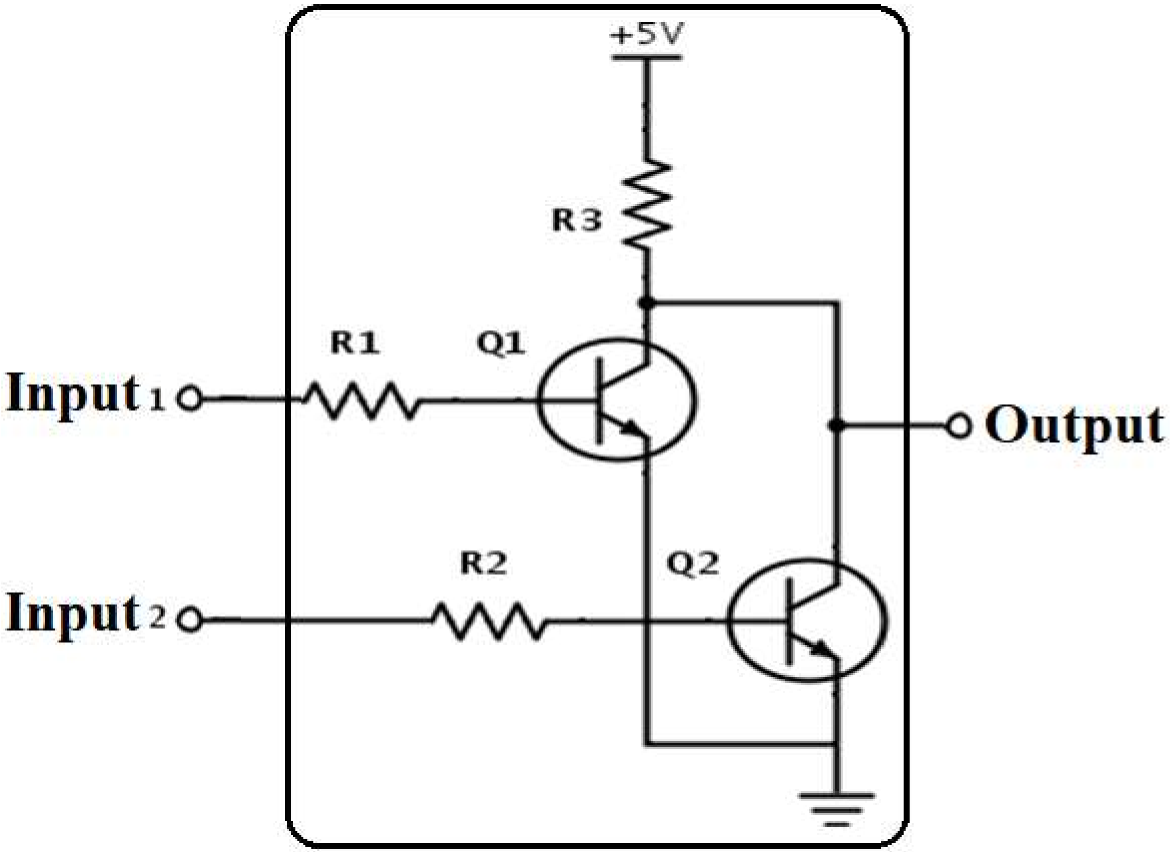
logic Gate NOT OR (NOR)

- A negation at the output of an OR gate constitutes a NOR function (NOT OR).

A	B	$Y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0



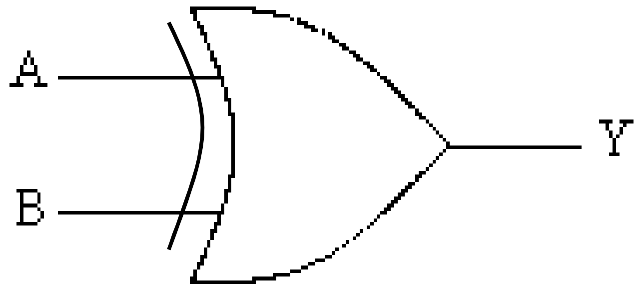
And here is the transistor-based circuit that allows obtaining a NOR gate (transistors).



logic Gate Exclusive OR (XOR)

- At least two inputs.
- The output of an XOR function is in state 1 if the number of its inputs at 1 is an odd number.

A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



Implementation of Boolean Functions

- Any logical function can be implemented using gates.
- Implementation of a Boolean function: Write the equation of the function based on its truth table.
- Simplify the equation. Implement the equation using available gates.

How to turn a truth table into a Boolean function

From the truth table, we can have two analytical forms, known as canonical forms:

- Canonical sum of products (Minterm)
- Canonical product of sums (Maxterm)

Canonical expressions

- 3 variables, a product term, which we call a minterm, equal to the AND of the variables that make up this combination.

				P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
	x	**y**	**z**	x y z	x y z	x y z	x y z	x y z	x y z	x y z	x y z
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
1	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
2	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
3	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
4	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
5	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
6	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
7	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

Example of canonical expressions

A B C	F	$P_3 + P_5 + P_6 + P_7$
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	1	1
1 0 0	0	0
1 0 1	1	1
1 1 0	1	1
1 1 1	1	1

This general way of writing a Boolean function is called the canonical sum of products.

$$\mathbf{F(A, B, C) = P_3 + P_5 + P_6 + P_7}$$

$$F(A,B,C) = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC = \sum(3,5,6,7)$$

Canonical Expressions (POS)

- 3 variables, sum term, referred to as maxterm, equal to the OR of the variables that make up this combination.

				S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
	X	Y	Z	$X+Y+Z$	$X+Y+\overline{Z}$	$X+\overline{Y}+Z$	$X+\overline{Y}+\overline{Z}$	$\overline{X}+Y+Z$	$\overline{X}+Y+\overline{Z}$	$\overline{X}+\overline{Y}+Z$	$\overline{X}+\overline{Y}+\overline{Z}$
0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
2	0	1	0	1	1	0	1	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1
4	1	0	0	1	1	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0	1	1
6	1	1	0	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	0

Canonical Expressions (POS)

X	Y	Z	F	$S_0 \cdot S_1 \cdot S_2 \cdot S_4$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$$\mathbf{F(X, Y, Z) = S_0 \cdot S_1 \cdot S_2 \cdot S_4}$$

$$F(X, Y, Z) = (\overline{X} + Y + Z)(X + \overline{Y} + Z)(X + Y + \overline{Z})(X + Y + Z)$$

This expression is called the canonical product of sums.

Canonical Expressions

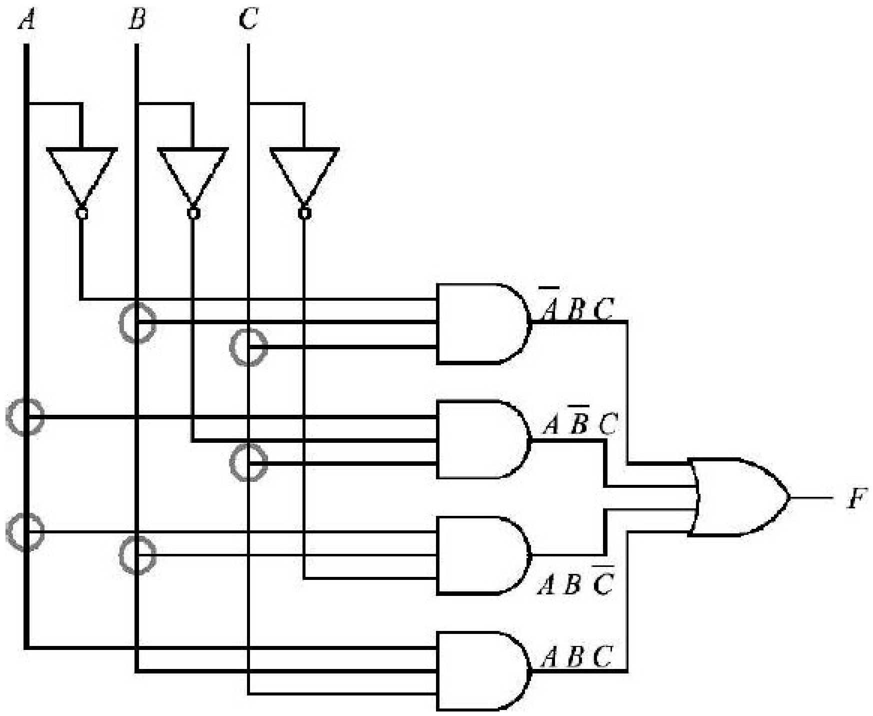
Canonical expressions express a Boolean function using the logical operators AND, OR, NOT.

A function can be implemented using the gates AND, OR, NOT.

Canonical Expressions of a Logical Function

$\overline{A}BC$

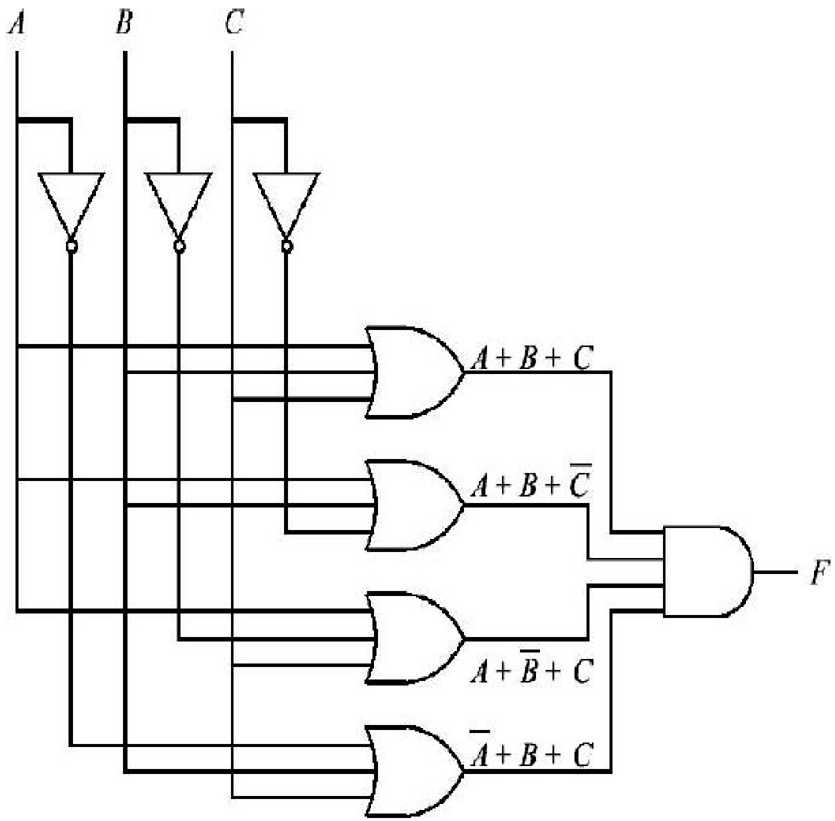
A	B	C	F	$P_3 + P_5 + P_6 + P_7$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



Canonical Expressions of a Logical Function

A+B+C

A	B	C	F	$S_0 \cdot S_1 \cdot S_2 \cdot S_4$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



Equivalence Relationship of Circuits

- Major Concerns for Designers
- Reduce the number of gates required for system implementation.
  - ✓ Minimize the cost in terms of the number of packages.
  - ✓ Electrical power consumption.
- Minimize complexity.
  - ✓ Create an equivalent system with certain optimized parameters.
- Search for equivalence.
  - ✓ Use the laws and theorems of Boolean algebra.

Summary of Basic Boolean Identities

OR	(A + B) + C = A + (B + C) = A + B + C A + B = B + A A + A = A A + 0 = A A + 1 = 1
AND	(A • B) • C = A • (B • C) = A • B • C A • B = B • A A • A = A A • 1 = A A • 0 = 0
Distributivity	A • (B + C) = (A • B) + (A • C) A + (B • C) = (A + B) • (A + C)

Summary of Basic Boolean Identities

NOT	$\overline{\overline{A}} = A$ $\overline{A} + A = 1$ $\overline{A} \bullet A = 0$
Absorption Law,	$A + (A \bullet B) = A$ $A \bullet (A + B) = A$
De Morgan's Law	$\overline{A \bullet B \bullet C \bullet \ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots$ $\overline{A + B + C + \ldots} = \overline{A} \bullet \overline{B} \bullet \overline{C} \bullet \ldots$
Exclusive OR	$A \oplus B = (A + B) \bullet (\overline{A \bullet B})$ $A \oplus B = (A \bullet \overline{B}) + (B \bullet \overline{A})$ $A \oplus B = (\overline{(A \bullet B) + (\overline{A} \bullet \overline{B})})$ $A \oplus B = (A + B) \bullet (\overline{A} + \overline{B})$ $A \oplus B = \overline{A} \bullet B + A \bullet \overline{B}$

Equivalence Relationship of Circuits

- Algebraic Manipulation

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

$$= C \bullet (\underline{\overline{A} \bullet \overline{B} + A \bullet B}) + \overline{C} \bullet (\underline{A \bullet \overline{B} + \overline{A} \bullet B})$$

$$= C \bullet (\overline{A \oplus B}) + \overline{C} \bullet (A \oplus B) = A \oplus B \oplus C$$

Equivalence Relationship of Circuits

Two logical functions are equivalent:

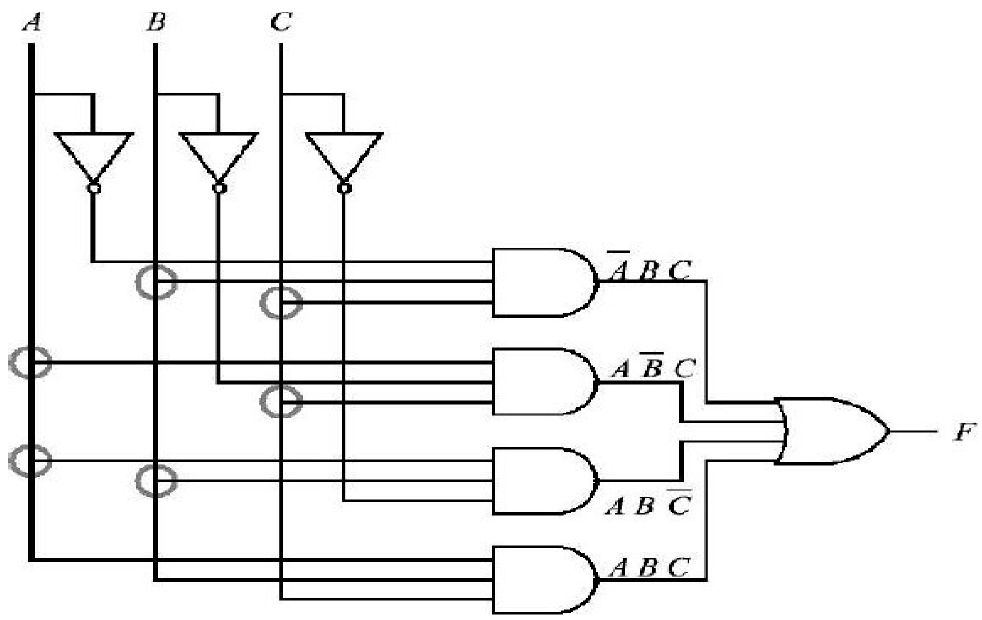
if and only if,

the values of their outputs are the same for all identical configurations of their input variables.

logical Fonctions

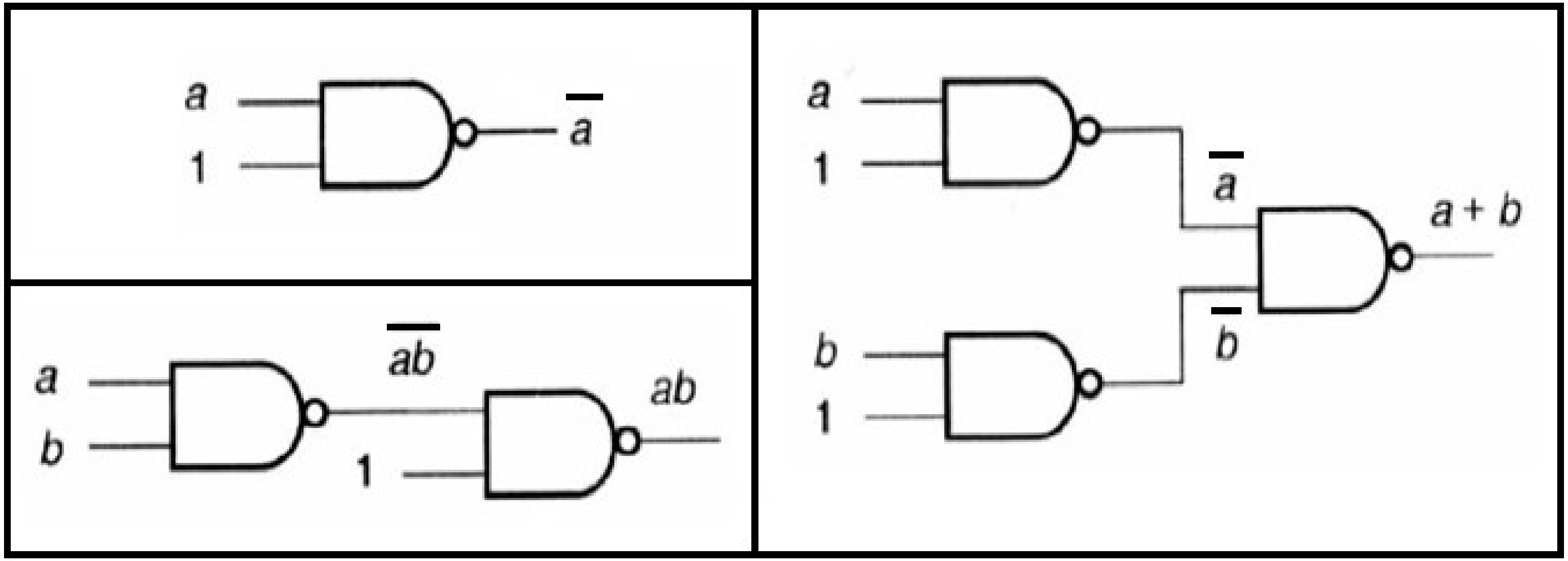
- Any Boolean function of any number of variables can be expressed using the three basic functions AND, OR, and NOT.
- The set { AND, OR, NOT } is complete.

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



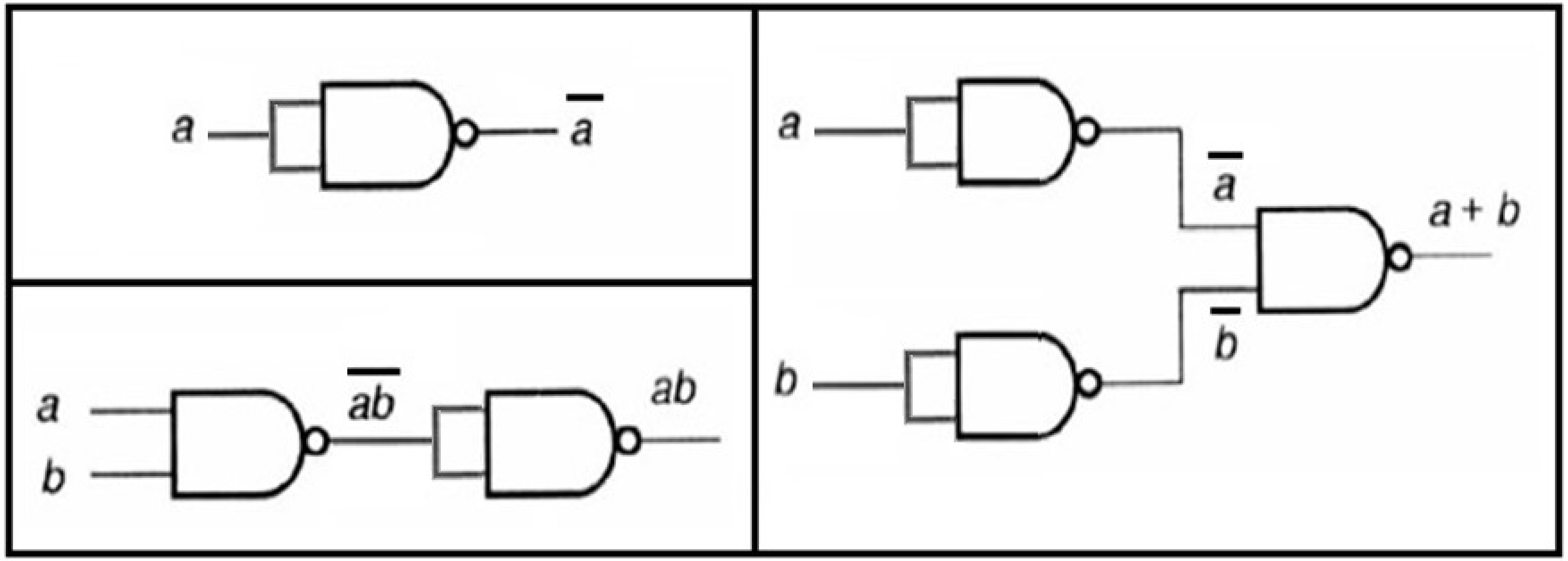
Set { NOT-AND (NAND) }

- { NOT-AND (NAND) } is complete and minimal. The gates NOT, OR, and AND can be obtained from NOT-AND gates.



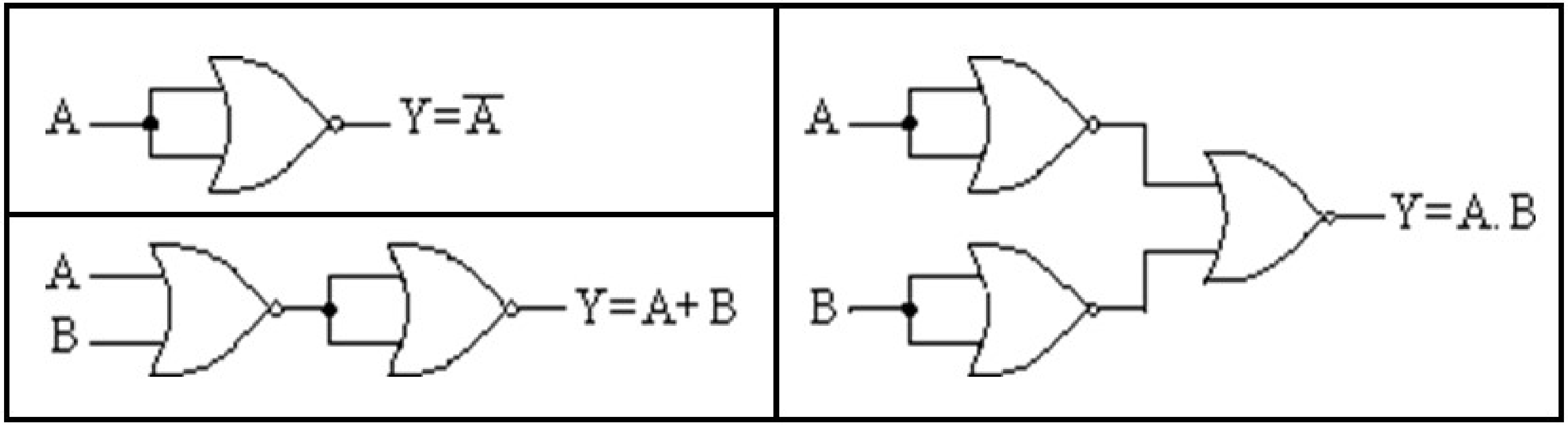
Set { NOT-AND (NAND) }

- { NOT-AND (NAND) } is complete and minimal. The gates NOT, OR, and AND can be obtained from NOT-AND gates.



Set { NOT-OR (NOR) }

- { NOT-OR (NOR) } is complete and minimal. The gates NOT, OR, and AND can be obtained from NOT-OR gates.

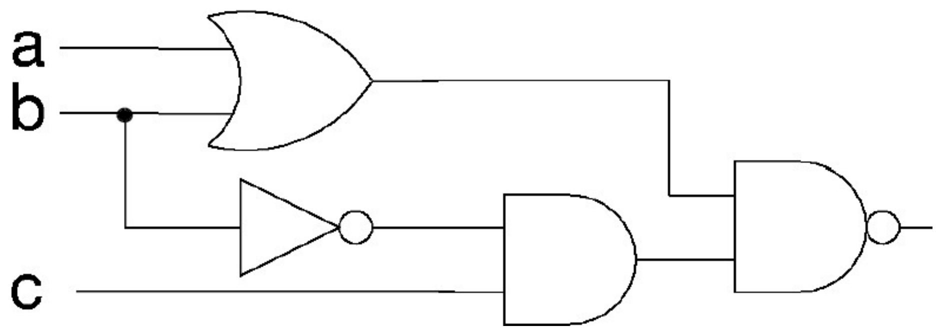


Logical Circuit Analysis

- Finding its logical function
- Principle
- Provide the expression of the outputs for each gate/component based on the input values.
- Finally deduce the logical function(s) of the circuit.
- Next, one can Determine the truth table of the circuit.
- Simplify the logical function.

Logical Circuit Analysis

Example: 3 inputs, 1 output Composed uniquely of OR, AND, and NOT logic gates.

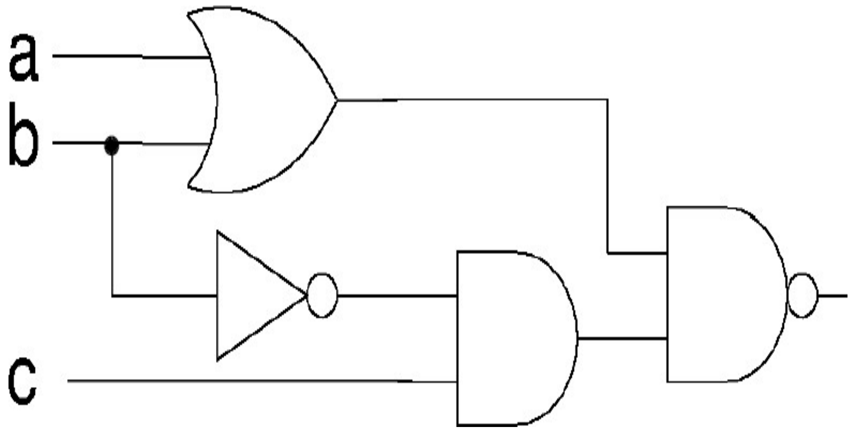


- From its logic diagram

$$f(a,b,c) = \overline{(a+b) \bullet (\overline{b} \bullet c)}$$

Logical Circuit Analysis

$$f(a,b,c) = \overline{(a+b) \bullet (\overline{b} \bullet c)}$$



$$f(a,b,c) = (\overline{a} + b + \overline{c})$$

a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Synthesis of a logical circuit

- From a logical function, find the corresponding logic diagram for that function
- Principle
- Simplify the logical function using two methods:

  - The algebraic method (Boolean algebra)
  - The Karnaugh map method
  - Deduce the corresponding logic diagram.

Simplification of Boolean Expression

- The algebraic method (Boolean algebra) The Karnaugh map method

$$F(A, B, C) = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$= \sum(3, 5, 6, 7)$$

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The Karnaugh Map Method

Graphical Simplification Methods

The Karnaugh map of a logical function is a graphical transformation of the truth table that enables the visualization of all minterms.

The Karnaugh Map Method

- A minterm is represented by a cell in the Karnaugh map. The cells are arranged in such a way that minterms differing only by the state of a single variable share a common border either in a row or a column, or are located at the ends of a row or column.

$$F(A, B, C) = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$
$$= \sum(3, 5, 6, 7)$$

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

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

The Karnaugh Map Method

1. Translation of the truth table into a Karnaugh map;
2. Formation of groups of 1, 2, 4, 8 terms (powers of 2);
3. Minimization of groups (maximization of terms within a group); If a group has only one term, then no action is taken; Elimination of variables that change state, and retention of the product of variables that have not changed state within the group;
4. The final logical expression is the union of the groups after the elimination of variables.

The Karnaugh Map Method

- Formation of groups of 1, 2, 4, 8 terms (powers of 2)
- Minimization of groups
- Maximization of terms within a group

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

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

The Karnaugh Map Method

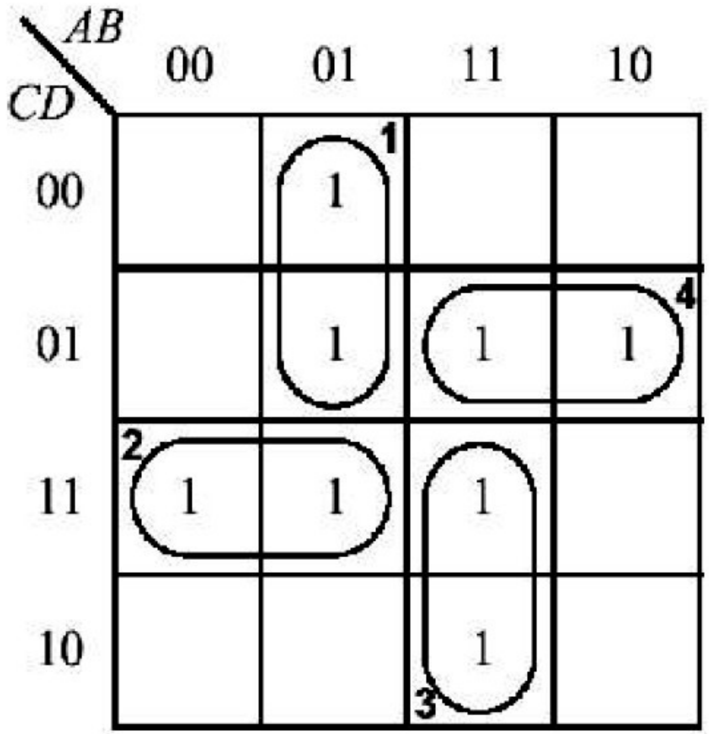
- We eliminate the variables that change state and retain the product of variables that have not changed state within the group.

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

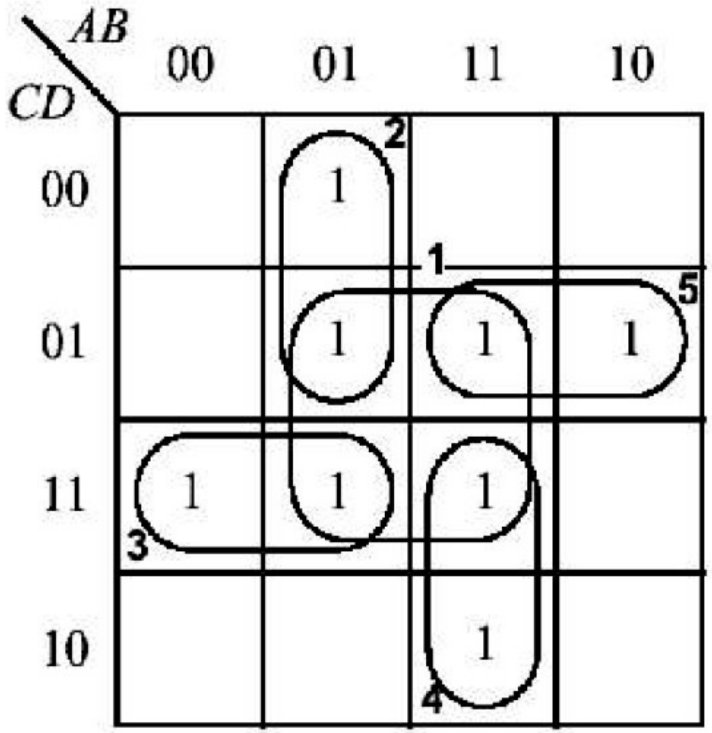
C \ AB	00	01	11	10
0			1	
1		1	1	1

F = AB + BC + AC

Minimal and Non-minimal Grouping



$$F = \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$



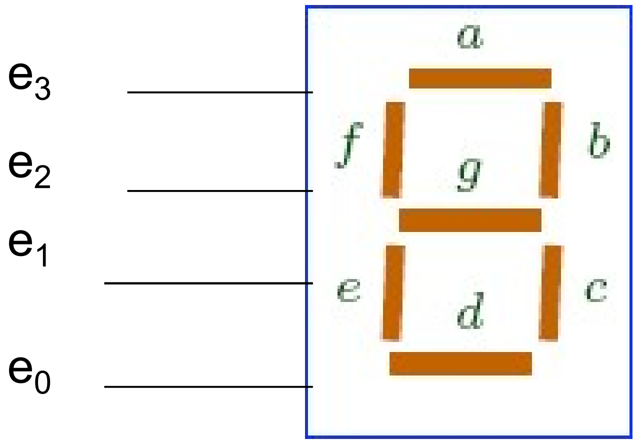
$$F = B\,D + \overline{A}\,B\,\overline{C} + \overline{A}\,C\,D + A\,B\,C + A\,\overline{C}\,D$$

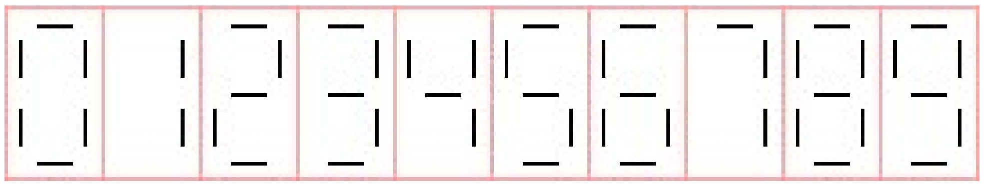
Incompletely Specified Boolean Functions

- There are Boolean functions for which there are no values associated with certain product terms.
- These terms are never 'selected,' and the associated value can be either 0 or 1 indifferently.
- They are noted as 'd' (don't care).
- The 7-segment display is a particular example of an incompletely specified Boolean function.

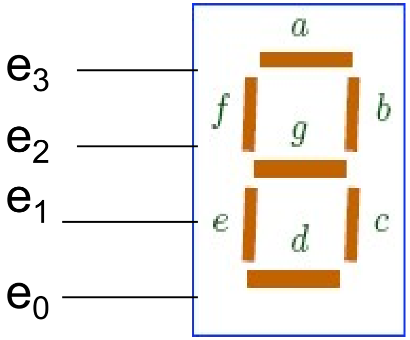
The 7-segment display

- We want to display the 10 decimal digits using 7 segments, labeled from a to g, which can be either 0 (off) or 1 (on). The encoding of the 10 decimal digits requires 4 bits, which can be noted as e_3 to e_0.





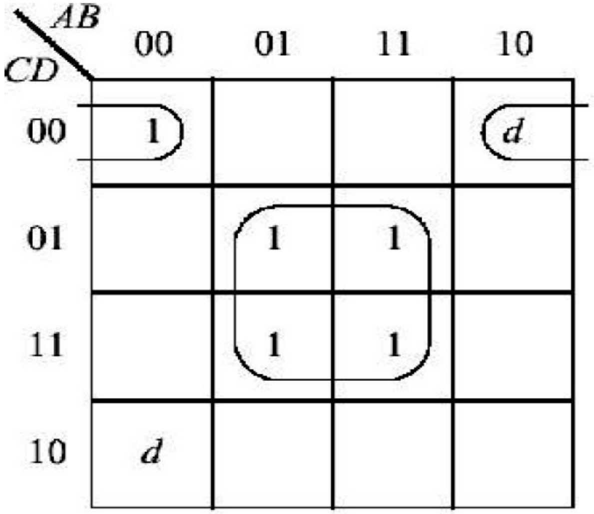
The 7-segment display



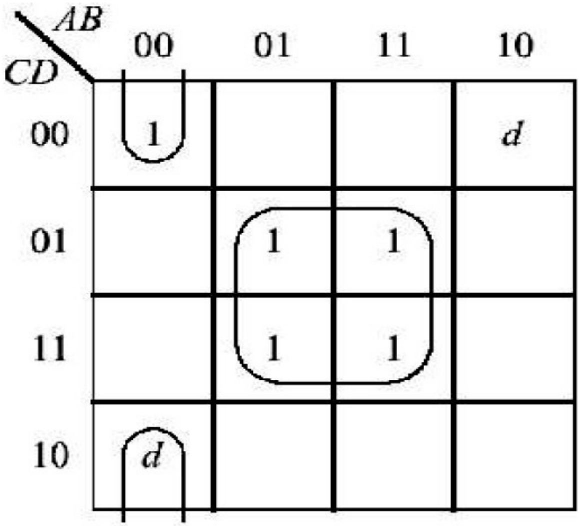
e3	e2	e1	e0		a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	1	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	0	0	1	1
1	0	1	0	10	d	d	d	d	d	d	d
1	0	1	1	11	d	d	d	d	d	d	d
1	1	0	0	12	d	d	d	d	d	d	d
1	1	0	1	13	d	d	d	d	d	d	d
1	1	1	0	14	d	d	d	d	d	d	d
1	1	1	1	15	d	d	d	d	d	d	d

Karnaugh Map and (Don't Care)

- When a variable can be either a '1' or a '0,' symbolized by a 'd' (don't care), there may be more than one minimal grouping.



$F = \overline{B}\,\overline{C}\,\overline{D} + B\,D$



$F = \overline{A}\,\overline{B}\,\overline{D} + B\,D$

End of the introduction chapter.