DIGITAL TECHNICS I

Dr. Bálint Pődör

Óbuda University, Microelectronics and Technology Institute

2. LECTURE:
LOGIC (BOOLEAN) ALGEBRA AND APPLICATIONS

1st year BSc course 1st (Autumn) term 2018/2019

1

BOOLEAN ALGEBRA
AND DIGITAL/LOGIC NETWORKS

Boolean algebra is the basic tool of analysis
and synthesis of logic networks

The math of Boolean algebra is an (improper) subset of the field of Discrete Mathematics and also formal logic.

Discrete math is an important part of understanding computational reasoning.

Boolean algebra can be used to describe logic operations in a concise form.

LOGIC (BOOLEAN) ALGEBRA: ITS ROOTS

Boolean algebra is a brand of mathematics that was first developed systematically, because of its applications to logic, by the English mathematician *George Boole*, around 1850. Contributions by *Augustus De Morgan* (Scottish mathematician, contemporary of Boole) and by *Claude Shannon* dubbed the father of information theory (1930'ies and 1940'ies, Bell Labs) are also important.
Beginning from 1847 *Boole* and *De Morgan* developed systematically formal logic in mathematical form (Boolean algebra). Binary systems were then already in use to control watches. The two fundamental works (books) of Boole are

The Mathematical Analysis of Logic (1847)

and

An Investigation of Laws of Thought (1854)

BOOLEAN ALGEBRA AND SWITCHING NETWORKS

A modern engineering application is to switching, digital and computer circuit design.

The connection between Boolean algebra and switching circuits has been established by Claude Shannon in the 1930's (in his Master's thesis).
Claude Elwood Shannon (1916-2001) pioneer of information theory, an MIT graduate then an employee of Bell Labs recognized in 1938 the applicability of Boolean algebra for the analysis and synthesis of (phone-)switching systems built from mechanical relays.

Boolean algebra is the main analytical tool for the analysis and synthesis of logic circuits and networks.

BOOLEAN ALGEBRA: RELEVANCE

In the 1930s, while studying switching circuits, Claude Shannon observed that one could also apply the rules of Boole's algebra in this setting, and he introduced switching algebra as a way to analyze and design circuits by algebraic means in terms of logic gate.

Shannon already had at his disposal the abstract mathematical apparatus, thus he cast his switching algebra as the two-element Boolean algebra.

In circuit engineering settings today, there is little need to consider other Boolean algebras, thus "switching algebra" and "Boolean algebra" are often used interchangeably.

BOOLEAN ALGEBRA: RELEVANCE

Efficient implementation of Boolean functions is a fundamental problem in the design of combinational circuits.

Modern electronic design automation tools for VLSI circuits often relay on an efficient representation of Boolean functions like (reduced ordered) binary decision diagrams (BDD) for logic synthesis and formal verification.

Connection Between
Boolean Calculus and Physical Circuits
Shannon 1938

Shannon
1916-2001

A Symbolic Analysis of Relay and Switching Circuits*

*Claude E. Shannon***

Shannon's advisor both MSc and PhD – a mathematician

* *Transactions American Institute of Electrical Engineers*, vol. 57, 1938. (Paper number 38-80, recommended by the AIEE committees on communication and basic sciences and presented at the AIEE summer convention, Washington, D.C., June 20-24, 1938. Manuscript submitted March 1, 1938; made available for preprinting May 27, 1938.)

** Claude E. Shannon is a research assistant in the department of electrical engineering at Massachusetts Institute of Technology, Cambridge. This paper is an abstract of a thesis presented at MIT for the degree of master of science. The author is indebted to Doctor F. L. Hitchcock, Doctor Vannevar Bush, and Doctor S. H. Caldwell, all of MIT, for helpful encouragement and criticism.

Connection Between
Boolean Calculus and Physical Circuits
Shannon 1938

- **A1. Identities:** $a+0=a$ and $a\cdot 1=a$
- **A2. Complements:** $a+\bar{a}=1$ and $a\cdot\bar{a}=0$
- **A3. Commutativity:** $a+b=b+a$ and $a\cdot b=b\cdot a$
- **A4. Distributivity:** $a+b\cdot c=(a+b)\cdot(a+c)$ and $a\cdot(b+c)=a\cdot b+a\cdot c$

???

Boolean functions ⟷ Relay circuits

BOOLEAN ALGEBRA: LOGIC VARIABLES

Logic variables can be used for an abstract description of individual events. They can have two values, **1** or **0**, depending on whether the event in question occurs or not.

If the event occurs,
then the value of the logic variable is **1**.

If the event does nor occur,
then the value of the logic variable is **0**.

SET OF VALUES OF LOGIC VARIABLES

TRUE/FALSE, YES/NO or 1/0 refers to the occurrence of the event.
Here 1 and 0 are not digits, they meaning is symbolic:

TRUE ↔ 1 and FALSE ↔ 0.

The meaning of HIGH/LOW is connected with the usual electrical representation of logic vales, they correspond to high(er) and low(er) potentials (voltage levels) e.g. nominally +5 V and 0 V respectively.

BOOLEAN ALGEBRA: LOGIC VARIABLES

Logic variables can be classified into two groups, as

independent,

and

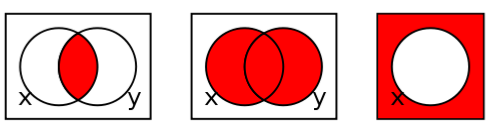
dependent variables.

Notations (used here): A, B, C, ... X, Y, Z.

The letters in the first part if the alphabet will be reserved (mostly…) for the independent variables.

VISUALISATION: VENN DIAGRAMS

A Venn diagram is a representation of a Boolean operation using shaded overlapping regions. There is one region for each variable, all circular in the examples here. The interior and exterior of region *x* corresponds respectively to the values 1 (true) and 0 (false) for variable *x*. The shading indicates the value of the operation for each combination of regions, with dark denoting 1 and light 0 (some authors use the opposite convention).

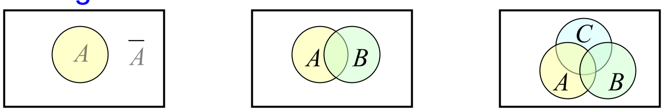


$x \wedge y$ $x \vee y$ $\neg x$

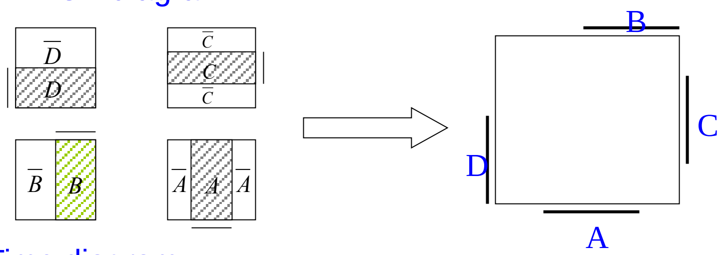
x AND y   x OR y   NOT x

LOGIC VARIABLES VISUALIZATION

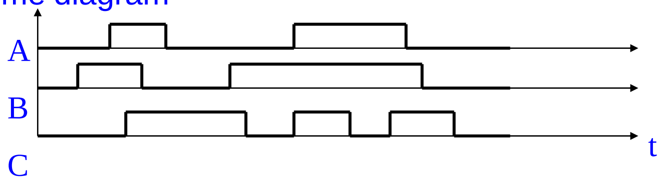
VENN diagram



VEITCH diagram



Time diagram



AXIOMS OF BOOELAN ALGEBRA

The **axioms** are fixed fundamental statements which are always valid in a given algebraic system, however the themselves cannot be proved or cannot be deduced from other theorems. The basic statements define the elements of the given set, the operation, their properties, etc.

The **theorems** however can be proved based on the axioms, they are direct consequences of the fundamental statements.

The Boolean algebra is based on the five axioms/postulates listed below:

BOOLEAN POSTULATES

1. Boolean algebra is defined on a set of two-valued elements
2. Each element of the set has its complementary also belonging to the set
3. Logic operations: conjunction (logic AND) and disjunction (logic OR)
4. Logic operations are commutative, associative, and distributive
5. Special elements of the set are the unity (its value is always 1) and the zero (its value is always 0)

15

COMPLEMENTATION: NEGATION

Logic or Boolean algebra is based on the axioms/postulations listed above.

For the technical realizations of logic problems an operation which gives the complement of an element of the set is also necessary. This is the LOGIC NEGATION.

$A \bullet \overline{A} = 0$ and $\overline{A} + A = 1$

LOGIC OPERATIONS

Three fundamental operations

- **AND** (conjunction) – logic multiplication;
- **OR** (disjunction) – logic addition;
- **NOT** (negation, inversion, complementation) - logical negation.

The AND and OR logic operations are two- or multi-variables. They define logic connections between at least two variables or between at least two groups of variables. The NOT (logic negation) is a one-variable operation, which can act on any of the variables, or any of the group of variables.

.

FUNDAMENTAL BOOLEAN THEOREMS

commutative law

$$A \bullet B = B \bullet A$$
$$A + B = B + A$$

associative law

$$A \bullet (B \bullet C) = (A \bullet B) \bullet C = A \bullet B \bullet C$$
$$A + (B + C) = (A + B) + C = A + B + C$$

distributive law

$$A \bullet (B + C) = A \bullet B + A \bullet C$$
$$A + (B \bullet C) = (A + B) \bullet (A + C)$$

Logic operations are commutative, associative, and distributive

18

UNITY AND ZERO ELEMENT

Specific elements of the set are

The **unity** element
(its value within the set is always TRUE, i.e. 1),

$$A \bullet 1 = 1 \bullet A = A$$

and

the **zero** element
(its value within the set is always FALSE, i.e. 0)

$$A + 0 = 0 + A = A$$

THE AND OPERATION (CONJUNCTION, LOGIC MULTIPLICATION)

The result of AND operation is TRUE if and only if both input operands are TRUE.
Truth table of the AND operation:

B	A	K=A•B
0	0	0
1	0	0
0	1	0
1	1	1

In logic algebra the AND operation is also called logic multiplication and is denoted by dot (•), however the dot is usually not written explicitly:

$$K = A \bullet B = A\,B$$

$0 \cdot 0 = 0$
$0 \cdot 1 = 1$
$1 \cdot 0 = 1$
$1 \cdot 1 = 1$

The truth table of the logic AND operation formally agrees with that of the common (numeric, binary) algebra.

THE TRUTH TABLE

Every logic problem can be phrased in terms of the three basic logic operations (AND, OR and INV/NOT). The more complicated the problem, the larger the number of logic variable and gates becomes, until we have difficulty in stating the problem precisely in words, i.e. taking into account all the possibilities.

Fortunately organized /systematic procedures exist which can tabulate or list all the possibilities that can arise.

The most basic procedure employed, and also one of the best is the truth table.

THE TRUTH TABLE

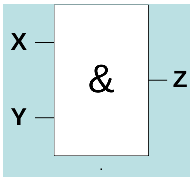
The truth table makes use o the fact that if we have N two-state variables, there are 2^N different ways of combining them. These 2^N possible combinations are set forth in the truth table.
E.g. for the two variable AND operation with N = 2,
$2^N = 4$, and as was shown above:

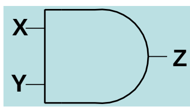
B	A	K=A•B
0	0	0
1	0	0
0	1	0
1	1	1

As the number of variables increases, so does the complexity of the truth table.

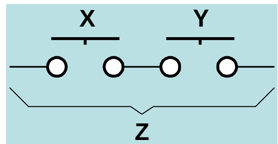
AND CIRCUIT SYMBOLS

Logic gate circuits





Switching circuits



Series connected and in ground state open (=MAKE) switches

THE OR OPERATION (DISJUNCTION, LOGIC ADDITION)

The result of OR operation is TRUE if any input operands are TRUE.

Truth table of the OR operation:

B	A	K = A + B
0	0	0
1	0	1
0	1	1
1	1	1

In logic algebra the OR operation is also called logic addition and is denoted by add symbol (+)

K = A + B

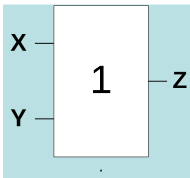
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1

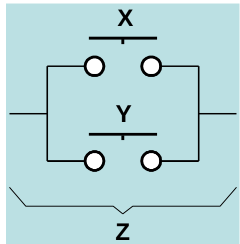
Except the last row the truth table of the logic OR operation formally agrees with that of the common (numeric, binary) algebra. It is equivalent with the mod2 addition.

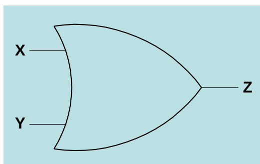
OR CIRCUIT SYMBOLS

Logic gate circuits

Switching circuits







Parallel connected and in ground state open (= MAKE) switches

LOGIC MULTIPLICATION AND ADDITION WITH MORE THAN TWO VARIABLES

The definition of both logic operation can be naturally extended two more than two operands.

The order of performing of the operations should be properly denoted by parentheses as in common algebra.

AND, OR OPERATIONS AND OPEN/CLOSED CONTACTS

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

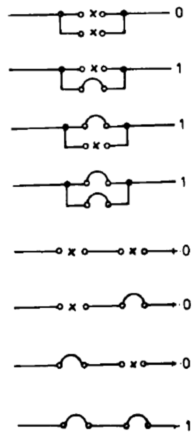
$1 + 1 = 1$

$0 \cdot 0 = 0$

$0 \cdot 1 = 0$

$1 \cdot 0 = 0$

$1 \cdot 1 = 1$



Electrical implementation of AND and OR: parallel and series connection of switching elements like electromechanical relays or n-and p-channel FETs in CMOS circuitry.

THE NOT OPERATION (LOGIC NEGATION, INVERSION, COMPLEMENTATION)

The result of NOT operation is TRUE if the single input value is FALSE, and vice versa.
Truth table:

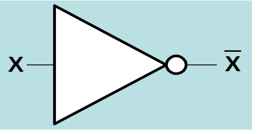
K	$K = \bar{A}$
0	1
1	0

In logic algebra the NOT operation is denoted by overline (bar)

$$K = \bar{A}$$

(pronounce: *a bar*)

NOT operation symbol
(note the bubble!)



THE NOT OPERATION (LOGIC NEGATION, INVERSION, COMPLEMENTATION)

Some properties:

$\overline{0} = 1$ and $\overline{1} = 0$

The NOT operation gives the opposite value of the operand in logic sense. Applying the NOT operation in an even number of times, the value of the operand remains unchanged:

$\overline{\overline{0}} = 0$ and $\overline{\overline{1}} = 1$

If applied in an odd number of times, the negated value is obtained.

FUNDAMENTAL PROPERTIES OF LOGIC OPERATIONS

Commutativity: the order of operands can be reversed

$$A \bullet B = B \bullet A$$

$$A + B = B + A$$

Associativity: the operands can be regrouped

$$A \bullet (B \bullet C) = (A \bullet B) \bullet C = A \bullet B \bullet C$$

$$A + (B + C) = (A + B) + C = A + B + C$$

The order of operations is given by the parantheses

Distributivity: the operands can be reordered

$$A \bullet (B + C) = A \bullet B + A \bullet C$$

$$A + (B \bullet C) = (A + B) \bullet (A + C)$$

MANIPULATION AND TRANSFORMATION OF LOGIC EXPRESSIONS

Based on the properties of logic operations the logic expressions can be transformed, and in this way it is possible to find the most simple equivalent forms.

This has a tremendous practical importance when realizing a logic expression using appropriate hardware.

The designer can then choose among the various forms to satisfy the requirements of the problem at hand. Typical practical reqirements involve choosing the circuit using the minimum number of gates, the minimum number of interconnection, or isung only one type of gate.

THEOREMS OF LOGIC ALGEBRA

Important **theorems**, without detailed proofs.
Their truth can be simply proved by substituting all possible combinations of variables.

Operations performed with the special elements (1 or 0):

1 • 1 = 1	**0 • 0 = 0**
1 • A = A	**0 • A = 0**
1 + 1 = 1	**0 + 0 = 0**
1 + A = 1	**0 + A =A**

THEOREMS OF LOGIC ALGEBRA: IDEMPOTENT LAW

Logic operations on the same variable:

Idempotent law:

$$A \bullet A = A \qquad A + A = A$$

Rules of negation:

$$A \bullet \bar{A} = 0 \qquad A + \bar{A} = 1$$

A can be a logic variable, or a result of a complex logic operation.

THEOREMS OF LOGIC ALGEBRA

Theorem of logic negation (twofold negation):

$$\bar{\bar{A}} = A$$

Generally: even number of negations does not change the value of a variable or expression, while odd number of negations changes it to the opposite.

UNITING THEOREM

Uniting theorem (absorption law)

$$A \cdot (A + B) = A$$

$$A + A \cdot B = A$$

These theorems are only valid in logic algebra, and they are not valid in the common algebra!

DE MORGAN'S THEOREM

De Morgan's theorem occupies an important place in the logic (Boolean) algebra

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

In logic, De Morgan's laws (or theorem) are rules in formal logic relating pairs of dual logic operators in a systematic manner expressed in terms of negation. De Morgan's theorem may be applied to the negation of a disjunction or to the negation of a conjunction in all part of a formula.

DE MORGAN'S THEOREM

Negation of a disjunction

$$\overline{A + B} = \overline{A} \bullet \overline{B}$$

Since two things are false, it's also false that either of them is true.

Negation of conjunction

$$\overline{A \bullet B} = \overline{A} + \overline{B}$$

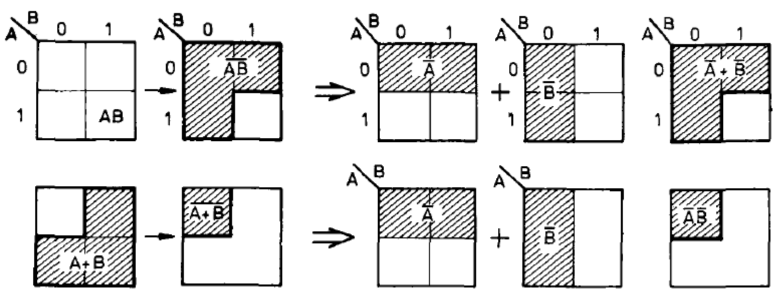
Since it is false that two things together are true, at last one of them should be false.

37

DE MORGAN'S THEOREM ON THE VEITCH DIAGRAM/KARNAUGH MAP

$$\overline{A + B} = \overline{A} \bullet \overline{B}$$

$$\overline{A \bullet B} = \overline{A} + \overline{B}$$



Representation of De Morgan's theorem on the Karnaugh map or Veitch diagram.

38

BOOLEAN THEOREMS: DUALITY

Note that each of the theorems is given in two different forms, called duals. In general, the dual of theorem is obtained by interchanging the AND and OR operations and also interchanging 1s and 0s if they are present.

39

DE MORGAN'S THEOREM

De Morgan's formulation of his theorem influenced the algebraization of logic undertaken by Boole, which cemented De Morgan's claim to the find, although a similar observation was made by Aristotle and was known to Greek and Medieval logicians, e.g. to William Ockham (1325), the great medieval scholastic philosopher.

In electrical engineering context the negation operator can be written as an overline (bar) above the terms to be negated.

There is a the *mnemonic* to help to memorize De Morgan's law

"break the line, change the operation"

40

GENERALIZED DE MORGAN'S THEOREMS

The De Morgan's theorem is an important tool in the analysis and synthesis of digital and logic circuits. Its generalization to several variables is stated below

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

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

41

KEY APPLICATION OF DE MORGAN THEOREM

AND operation using OR and NOT

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

OR operation using AND and NOT

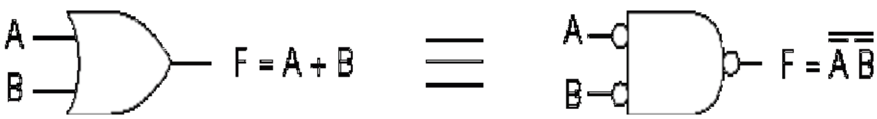
$$A + B = \overline{(\overline{A} \bullet \overline{B})}$$

42

DE MORGAN'S THEOREM

A B	$\overline{AB} = \bar{A} + \bar{B}$		$\overline{A+B} = \bar{A}\,\bar{B}$	
0 0	1	1	1	1
0 1	1	1	0	0
1 0	1	1	0	0
1 1	0	0	0	0

DeMorgan's theorem: $A + B = \overline{\overline{A+B}} = \overline{\bar{A}\,\bar{B}}$



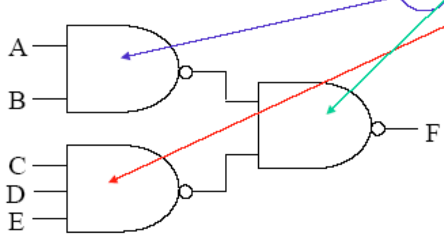
LOGIC SYNTHESIS GUIDED BY DE MORGAN'S THEOREM

DeMorgan's Theorem :

$A + B + C = \overline{[\bar{A}\,\bar{B}\,\bar{C}]}$ or $\bar{A} + \bar{B} + \bar{C} = \overline{[A\,B\,C]}$

Example of Using DeMorgan's Theorem:

$$F = A \bullet B + C \bullet D \bullet E = \overline{\overline{AB} \bullet \overline{CDE}}$$



Thus any sum of products expression can be immediately synthesized from NAND gates alone

IMPLEMENTATION OF COMBINATIONAL CIRCUITS

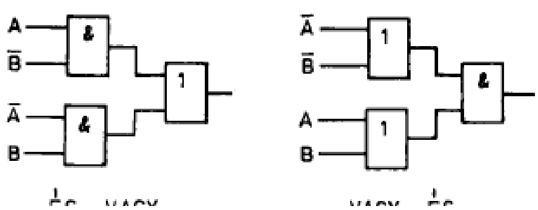
Any two-level AND-OR circuit (sum-of-products, SOP) can be implemented using a two-level NAND-NAND circuit.

Any two-level OR-AND circuit (product-of-sums, POS) can be implemented using a two-level NOR-NOR circuit.

All these are based on De Morgan's theorem.

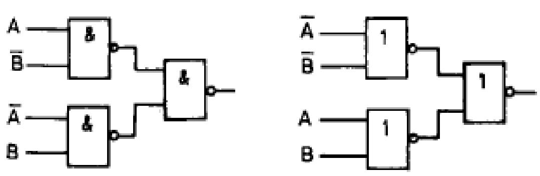
45

TWO LEVEL IMPLEMENTATIONS



AND-OR
From sum-of- product
(SOP) algebraic form

OR-AND
from product-of-sum
(POS) algebraic form



NAND

NOR

SHANNON'S GENERALIZATION OF DE MORGAN'S THEOREMS

The De Morgan-Shannon's theorem refers to the logic or Boolean functions constructed using logic multiplications and additions

$$\overline{f(A, B, C, ..., +, \bullet)} = f(\overline{A}, \overline{B}, \overline{C}, ..., \bullet, +)$$

The negation of the function can be performed by negating each variable and replacing all logic summations (ORs) with logic multiplications (ANDs) and replacing all logic multiplications (ANDs) with logic summations (ORs).

SHANNON'S EXPANSION THEOREMS

$$F(X_1, X_2,\ldots X_n) = X_1\ F(1,X_2,\ldots X_n) + \overline{X}_1\ F(0,X_2,\ldots X_n)$$

$$F(X_1, X_2,\ldots X_n) = [X_1 + F(0,X_2,\ldots X_n)]\ [\overline{X}_1 + F(1,X_2,\ldots X_n)]$$

Based on this theorem larger logic fucntions can be broken down to an appropriate combination of smaller logic fucntions. The basic step is to break down an N variable function to an AND-OR connection of two N-1 variable functions. The AND-OR connection can also be implemented with a 2-to-1 multiplexer.

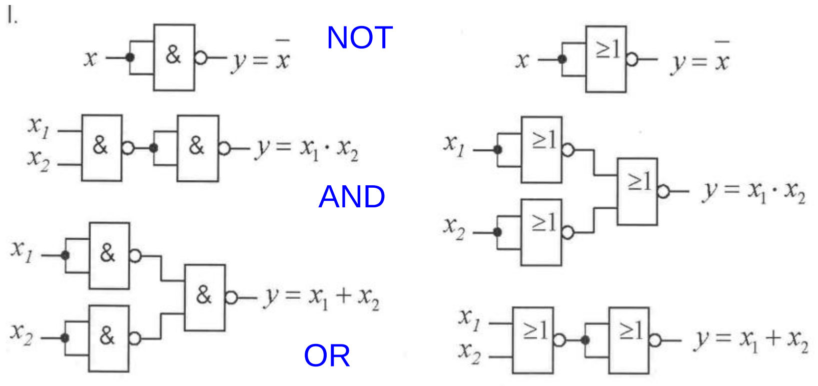
LOGIC GATES

- Elementary building blocks of logic circuits.
- They implement a basic logic operations.
- Complex logic networks can be implemented by appropriate interconnection of logic gates.
- However nowadays: use more complex building blocks and functional elements (complexity: few tens to few hundred gates), or some kind of programmable logic devices (complexity up to ten thousand gates or even more). E. g. the complexity of a 1 digit BCD-to-seven segment display decoder is about 30 gates, of a 4-bit ALU is less than 100 gates. Both are available in MSI in one package.

IMPLEMENTATION OF LOGIC CIRCUITS

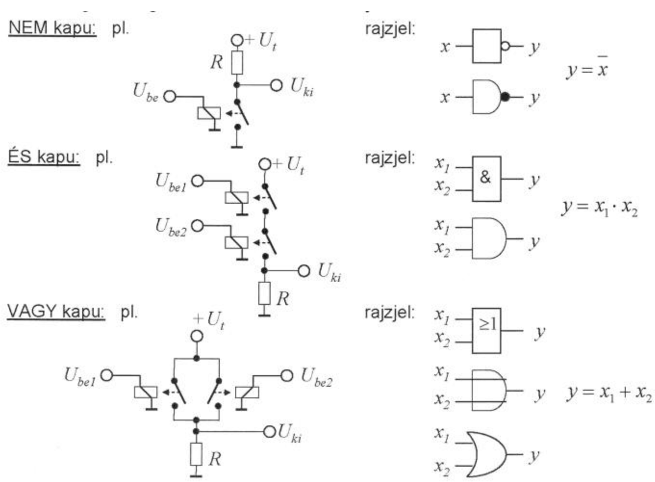
All logic functions and circuits can be described in terms of the three fundamental elements.
While the NOT, AND, OR functions have been designed as individual circuits in many circuit families, by far the most common functions realized as individual circuits are the NAND and NOR circuits. A NAND can be described as equivalent to an AND element driving a NOT element. Similarly, a NOR is equivalent to an OR element driving a NOT element.
The reason for this strong bias favouring inverting outputs is that the transistor, and the vacuum tube which preceded it, are by nature inverters or NOT-type devices when used as signal amplifiers. Electric and electronic switches (gates) do not readily perform the OR and AND logic operations, but most commercially available gates do perform the combined operations AND-NOT (NAND) and OR-NOT (NOR).

REALIZATTIONS BASED ON NOT-AND (NAND) AND NOT-OR (NOR) GATES



Realization of the three fundamental operation (NOT, AND and OR) with only NOT-AND (NAND) and NOT-OR (NOR) elements (gates).

FUNDAMENTAL LOGIC GATES: CIRCUITRY



Principles of circuit realization . Positive logic::
high voltage level ⇒1, low voltage level⇒0

PRACTICAL REALIZATION OF LOGIC GATES

In the two most common electronic realization of the logic circuits the switching element is:

- n- and p-channel silicon (Si) field effect transistor pair (CMOS FET, complementary metal-oxide-semiconductor field effect transistor),

and

- silicon (Si) diode and bipolar transistor (TTL circuits).

respectively.

The amplifying element in both cases is the corresponding transistor.

In TTL the NOR gate is the basic element, in CMOS mostly the NOR gate is the basic element, the because of electronic circuitry reasons.

GATE SYMBOLS

ANSI & IEEE has developed a standard set of logic symbols according to which the use of both rectangular and distinctive-shape logic gate symbols are allowed.

Notwithstandig several gate symbol systems are used, and various other standards are used too.

In Hungary the Hungarian Standard MSz 9200/33-73 gives the mandatory prescriptions with respect of logic gate symbols. All the textbooks which are compulsory for this course, use this.

LOGIC GATES AND SYMBOLS (1)

HUNG   USA   symbol

•Negation $\overline{A}$

A	F
0	1
1	0

• AND) A*B

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

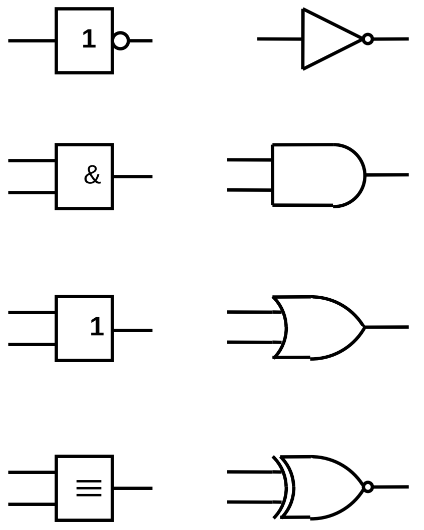
•OR A+B

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

•EQUIVALENCY

$A \otimes B = \overline{A} * \overline{B} + A * B$

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



Hungarian standard MSZ 9200/33-73

LOGIC GATES AND SYMBOLS (2)

MSZ   USA   symbol

•NOT-AND (NAND) $\overline{A*B}$

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

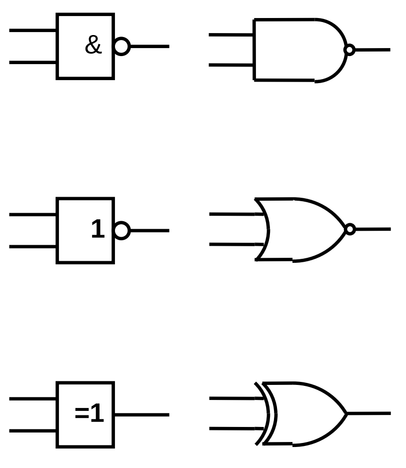
•NOT-OR (NOR) $\overline{A+B}$

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

•ANTIVALENCY

$A \oplus B = \overline{A} * B + A * \overline{B}$

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



ALGEBRAIC TRANSFORMATION OF LOGIC EXPRESSIONS: PRACTICE

Bring to a simpler form the following expressions:

$Y = AB + A\bar{B} + ABC\bar{D}$ Answer: $Y = A$

$Y = \overline{ABC} + \overline{A + B + C}$ Answer: $Y = \bar{A} + \bar{B} + \bar{C}$

$Y = \bar{C}BA + C\bar{B}A + CB\bar{A} + CBA$ Answer: $= AB + BC + CA$

Note: the last function is called the (3-variable) majority function, it also gives the carry-out in a 1-bit binary full adder (operands A and B, carry-in C).
It is to emphasized that the rules of manipulations in logic algebra are not the same as in the common algebra!

ALGEBRAIC FORM OF LOGIC FUNCTION GIVEN BY ITS TRUTH TABLE: EXAMPLE

ROW	A	B	C	Y
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	0
6	1	1	0	1
7	1	1	1	0

The algebraic form of the logic function Y(A,B,C) can be read off from the column Y, and can be written as the disjunction of three conjunctions (where Y = 1):

$$Y(A,B,C) = \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C}$$

Using appropriate factorings

$$Y = (\bar{A}B + A(\bar{B} + B))\bar{C} = (\bar{A}B + A)\bar{C}$$

$$Y(A,B,C) = (A + \bar{A})(A + B)\bar{C} = (A + B)\bar{C} = A\bar{C} + B\bar{C}$$

It can be seen that several equivalent algebraic form exists!
The last form cannot be reduced further, and is the same which can be obtained by using Karnaugh map minimization.

REALIZATION OF LOGIC FUNCTION

Both the original function

$$Y(A,B,C) = \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C}$$

and the reduced (minimized) form

$$Y(A,B,C) = A\bar{C} + B\bar{C}$$

can be realized with two-level AND-OR gate network, or based on De Morgan's theorem, with two-level NAND-NAND gate network. The first version requires 4 gates and 12 pins (gate inputs), the second version requires 3 gate and 6 pins (gate inputs).

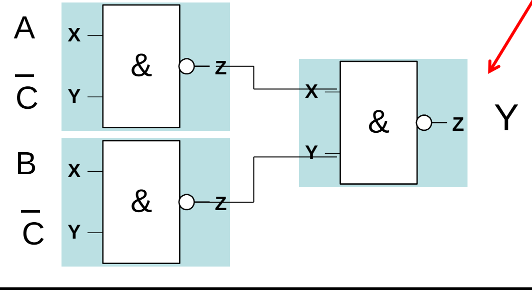
HOMOGENOUS NAND GATE CIRCUIT

The two-level homogeneous NAND gate realization follows from a transformation from the AND-OR expression based on De Morgans's theorem.

AND-OR

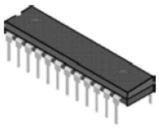
NAND-NAND

$$Y(A,B,C) = A\bar{C} + B\bar{C} = \overline{\overline{(A\bar{C})}\;\overline{(B\bar{C})}}$$

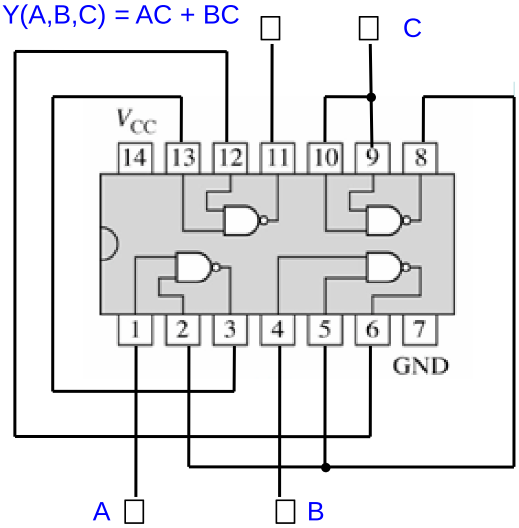


PHYSICAL REALIZATION IN THE LAB

74LS00
4 x 2-input NAND



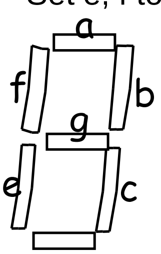
$Y(A,B,C) = A\bar{C} + B\bar{C}$

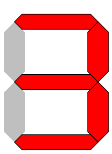


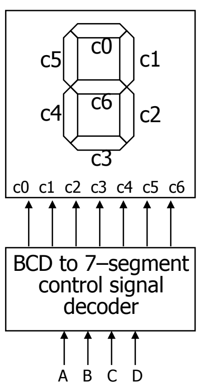
PROPOSED EXERCISE

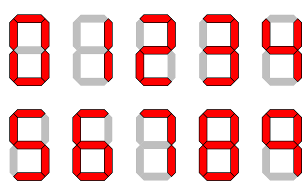
Construct the truth table of the BCD-to-seven segment display logic network!
(4 columns for the input variables, 7 columns for the output variables. 10 ”valid” rows, 6 rows ”free” (what to do with them?))

Display digit “3”?
Set a, b, c, d, g to 1
Set e, f to 0









PROPOSED EXERCISE

Exercise: construct the truth table of the BCD-to-seven segment display logic network!
(4 columns for the input variables, 7 columns for the output variables. 10 ”valid” rows, 6 rows ”free” (what to do with them?))

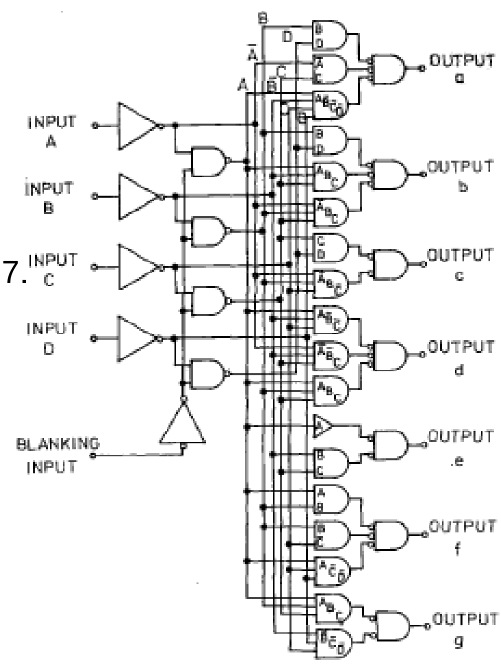
Exercise (for ambitious students): Draw also the logic diagram of a possible version of the BCD-to-seven-segment decoder!

Possible solution

c.f. *Zsom*, Vol I. Fig. 4.7.

or

54/74LS47 datasheet



REVIEW QUESTIONS

Consider each of the following statements and for each indicate for which logic gate or gates (AND, OR, NAND, NOR) the statement is true:

(a) Output is high only if all inputs are low.
(b) Output will be low if the inputs are at different levels.
(c) Output is high when both inputs are high.
(d) Output is low only if all inputs are high.
(e) All low inputs produce a high output.

(Adapted from: Ronald J. Tocci, Digital Systems, Prentice Hall, London,1980.)

REVISION QUESTIONS

1. State and interpret DeMorgan's theorem.

2. State and interpret Shannon's extension of DeMorgan's theorem.

3. Interpret and explain the following concepts:

(standard/extended) sum-of-product form, also known as (minterm/disjunctive) canonical form
(standard/extended) product-of-sum form, also known as (maxterm/conjunctive) canonical form

PROBLEMS AND EXERCISES

1. Show the validity of the following relation

$$(A + B)(A + C) = A + B\,C$$

2. Prove that

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

3. Simplify the following terms using DeMorgan's theorems:

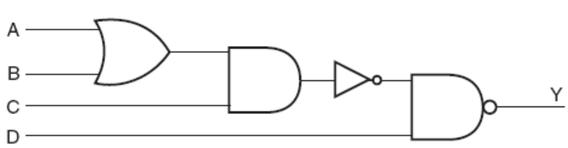
(a) $\overline{\overline{A}\,B\,\overline{C}}$ (b) $\overline{A + B\,\overline{C}}$ (c) $\overline{A\,B\,\overline{C\,D}}$

(ANS: (a) $A + \overline{B} + C$ (b) $\overline{A}\,\overline{B} + \overline{A}\,C$ (d) $\overline{A} + \overline{B} + C\,D$)

PROBLEMS AND EXERCISES

4. What is the only input combination that:
(a) Will produce a logic '1' at the output of an eight-input AND gate?
(b) Will produce a logic '0' at the output of a four-input NAND gate?
(c) Will produce a logic '1' at the output of an eight-input NOR gate?
(d) Will produce a logic '0' at the output of a four-input OR gate?

5. Draw the truth table of the logic circuit shown below:



6. It is proposed to construct an eight-input NAND gate using only two-input AND gates and two-input NAND gates. Draw the logic arrangement that uses the minimum number of logic gates.