DIGITAL TECHNICS I

Dr. Bálint Pődör

Óbuda University, Microelectronics and Technology Institute

3. LECTURE: LOGIC (BOOLEAN) FUNCTIONS AND APPLICATIONS

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

1

This is the

3. LECTURE…

But before starting with the new material, let's repeat a few important points from the previous lecture…

…because as the Latin proverb says

Repetitio est mater studiorum

2

AXIOMS OF BOOLEAN ALGEBRA

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)

Repetitio est mater studiorum

3

BOOLEAN THEOREMS

commutative law

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

associative law

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

distributive law

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

Repetitio est mater studiorum

4

DE MORGAN'S THEOREM

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

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

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

The negation of a logic sum (OR) or a logic product (AND) can be performed by negating (complementing) the logic operands and interchanging the logic operations (in place of OR put AND, in place of AND put OR).

Repetitio est mater studiorum

5

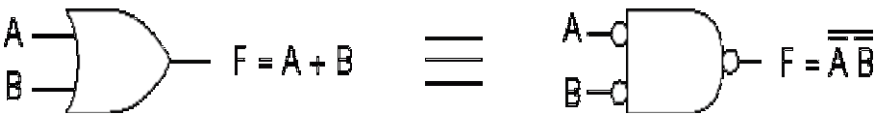
KEY APPLICATION OF DE MORGAN THEOREM

AND operation using OR and NOT

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

OR operation using AND and NOT

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



Repetitio est mater studiorum

6

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).

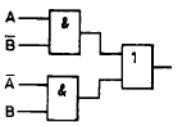
Repetitio est mater studiorum

7

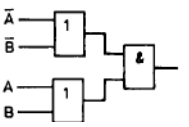
COMBINATIONAL CIRCUITS IN PRACTICE

From De Morgan's theorem it follows:

Any two-level AND-OR (sum-of-products) network can be implemented with two-level NAND-NAND network.

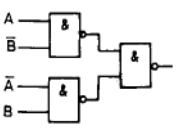


AND-OR
sum-of-products

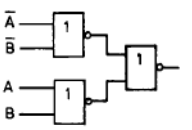


OR-AND
product-of-sums

Any two-level OR-AND (product-of-sums) network can be implemented with two-level NOR-NOR network.



NAND-NAND



NOR-NOR

Repetitio est mater studiorum

8

Some addenda …

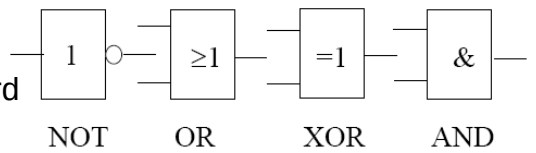
9

NOTE ON SYMBOLS

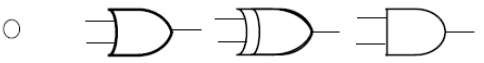
Standards (some examples)

German
DIN Norm

Hungarian standard



American
Standard



In connection
with gate

ALGEBRAIC TRANSFORMATION OF LOGIC EXPRESSIONS: PRACTICE

Bring to a simpler form the following expressions:

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

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

$Y = \overline{C}BA + C\overline{B}A + CB\overline{A} + CBA$ Answer: ?

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 TRANSFORMATION OF LOGIC EXPRESSIONS: PRACTICE

Bring to a simpler form the following expressions:

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

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

$Y = \overline{C}BA + C\overline{B}A + CB\overline{A} + CBA$ 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):

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

Using appropriate factorings

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

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

It can be seen that several equivalent algebraic forms exist! 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

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

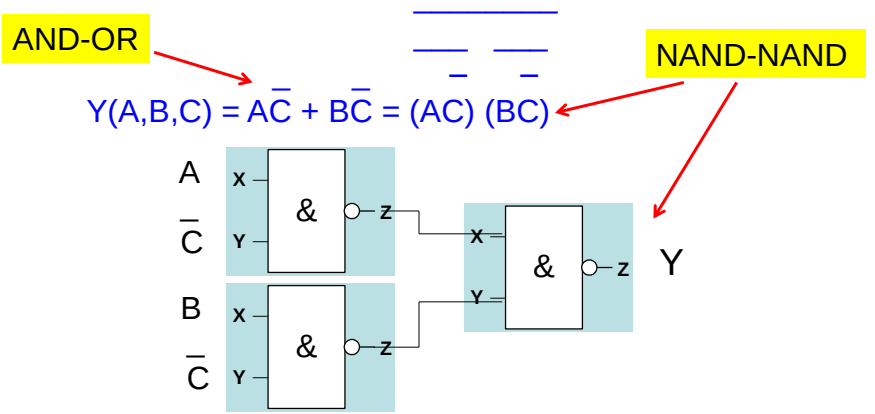
and the reduced (minimized) form

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

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.

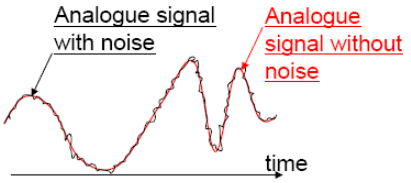


TECHNICAL ASPECTS: MAPPING FROM PHYSIACL WORLD TO BINARY WORLD

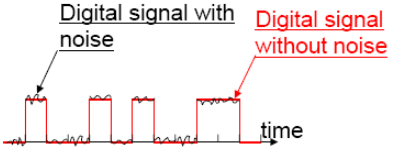
Technology	State 0	State 1
Relay logic	Circuit Open	Circuit Closed
CMOS logic	0.0-1.0 volts	2.0-3.0 volts
Transistor transistor logic (TTL)	0.0-0.8 volts	2.0-5.0 volts
Fiber Optics	Light off	Light on
Dynamic RAM	Discharged capacitor	Charged capacitor
Nonvolatile memory (erasable)	Trapped electrons	No trapped electrons
Programmable ROM	Fuse blown	Fuse intact
Bubble memory	No magnetic bubble	Bubble present
Magnetic disk	No flux reversal	Flux reversal
Compact disc	No pit	Pit

16

TECHNICAL ASPECTS: NOISE IN ANALOG AND DIGITAL SYSTEMS



Difficult to distinguish between signal with noise, and wanted signal without noise



Easy to distinguish between signal with noise, and wanted signal without noise

17

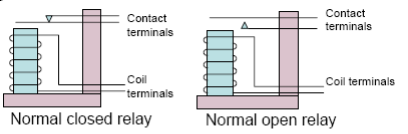
TECNICAL: ELEMENTARY SWITCHES

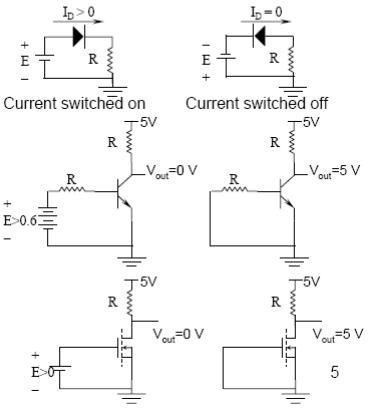
Mechanical switches:

- Relays

Electronical switches:

- Diodes:
- Bipolar transistor
- Field effect Transistor (NMOS-FET)





18

LOGIC (BOOLEAN) FUNCTIONS

1. Logic functions: fundamental concepts, general properties
2. Two-variable Boolean functions: a summary
3. Fully specified and incompletely specified logic functions and logic problems
4. Canonic algebraic forms of logic functions (disjunctive and conjunctive canonic forms).
5. The concept of minterm and maxterm, their properties

19

BOOLEAN FUNCTIONS

The one- and two-variable operations of Boolean algebra can be considered as functions of one and two variables, respectively.
In the case of generalized functions the number of variables is extended only.
n-variable Boolean or logic function

$$Z = f(X_1, X_2, ...X_n)$$

The particular truth value of Z is defined by the f function.

20

LOGIC (BOOLEAN) FUNCTIONS AND COMBINATIONAL NETWORKS

To each logic function a combinational network can be given, and *vice versa* to each logic problem and combinational network an appropriate logic function can connected.

With the help of logic function the operation of combinational networks can be unequivocally described.

For this reason it is necessary to get acquainted in detail with the properties of logic functions.

What follows is detailed survey of one- and two-variable logic functions.

21

ONE-VARIABLE LOGIC FUNCTIONS

In the case of one variable four Boolean functions are possible.

A	$f_o^1(A)$	$f_1^1(A)$	$f_2^1(A)$	$f_3^1(A)$
0	0	0	1	1
1	0	1	0	1

In the f_i^n notation the index n denotes the number of variables, the index i denotes the decimal value of the binary number represented in the corresponding column.

Two logic constant 0, 1, and two ”real” functions A, $\overline{A}$.

22

TWO-VARIABLE LOGIC FUNCTIONS

Classification and properties of two-variable Boolean functions.

In the case of two variables the number of possible input combinations is $2^2 = 4$, therefore the number of possible two-variable functions is $2^4 = 16$. Each function describes a single or complex logic operation.

Generalization: n variable, possible input combinations $k = 2^n$, the number of possible n-variable functions is 2^k, (exponential growth!).

E.g. for n = 3 256,
for n = 4 65 536,
for n = 5 4 294 967 296, …, etc.

23

TWO-VARIABLE BOOLEAN FUNCTIONS

A	B	f_o^2	f_1^2	f_2^2.....	f_8^2	f_{14}^2	f_{15}^2
0	0	0	1	0	0	0	1
0	1	0	0	1	0	1	1
1	0	0	0	0	0	1	1
1	1	0	0	0	1	1	1

Notation: f_i^n .
Index i: decimal value of the binary number read from the column (LSB upper row).
Functions with indices i and 15-i are complements of each other.

Cf. Rőmer's text p. 9, or Zsom's text (I) p. 71.

24

CLASSIFICATION OF BOOLEAN FUNCTIONS OF TWO VARIABLES

Function name	f(A,B)
Logical constants	0, 1
Functions of one variable	$A, \overline{A}, B, \overline{B}$
AND, OR, NAND, NOR	$A \bullet B, A+B, \overline{A \bullet B}, \overline{A+B}$
XOR ($A \oplus B$), XNOR ($A \odot B$)	$\overline{A}\,B+A\,\overline{B}, \overline{A}\,\overline{B}+A\,B$
INHIBITION	$A \supset B, B \supset A$
IMPLICATION	$A \rightarrow B, B \rightarrow A$

25

LOGIC CONSTANTS

A	B	f_o^2	f_1^2	f_2^2....	f_{14}^2	f_{15}^2
0	0	0	1	0	0	1
0	1	0	0	1	1	1
1	0	0	0	0	1	1
1	1	0	0	0	1	1

f_o^2 zero-function, its value is always 0, independently of the variables.
f_{15}^2 unity-function, its value is always 1, independently of the variables.
These are the logic constants.

26

ONE-VARIABLE FUNCTIONS

$f_{12}^2(A,B) = A$

$f_3^2(A,B) = \overline{A}$

$f_{10}^2(A,B) = B$

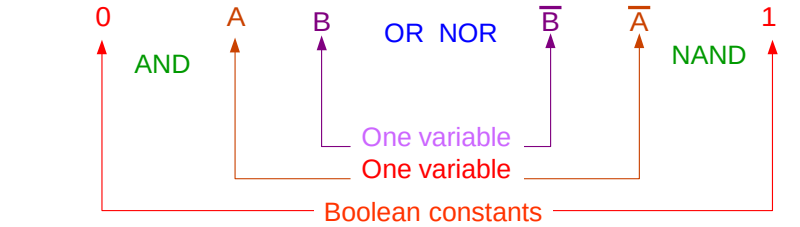
$f_5^2(A,B) = \overline{B}$

These are not true two-variable functions but one-variable functions. They represent the true and negated values of the variables.

27

BOOLEAN FUNCTIONS OF TWO VARIABLE

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1



28

LOGIC FUNCTIONS: AND, OR, NAND, NOR

A	B	f_1^2	f_7^2	f_8^2	f_{14}^2
0	0	1	1	0	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	0	0	1	1

$\overline{A+B}$	$\overline{AB}$	AB	A+B
NOR	NAND	AND	OR

29

TWO-VARAIBLE FUNCTIONS: ANTIVALENCY AND EQUVALENCY

Function name	f(A,B)
Logical constants	0, 1
Functions of one variable	A, $\overline{A}$, B, $\overline{B}$
AND, OR, NAND, NOR	A•B, A+B, $\overline{A \cdot B}$, $\overline{A+B}$
XOR ($A \oplus B$), XNOR ($A \odot B$)	$\overline{A}\,B+A\,\overline{B}$, $\overline{A}\,\overline{B}+A\,B$
INHIBITION	$A \supset B$, $B \supset A$
IMPLICATION	$A \rightarrow B$, $B \rightarrow A$

30

ANTIVALENCY (XOR)
EQUIVALENCY (XNOR)

A B	f_6^2	f_9^2
0 0	0	1
0 1	1	0
1 0	1	0
1 1	0	1

Antivalency (exclusive OR, XOR, $A \oplus B$)

Equivalency (coincidence, XNOR, $A \odot B$)

XOR $f_6^2 = A \oplus B = \overline{A}\,B + A\,\overline{B}$,

XNOR $f_9^2 = A \odot B = \overline{A}\,\overline{B} + A\,B$

The two functions are complementary: $A \oplus B = \overline{A \odot B}$

31

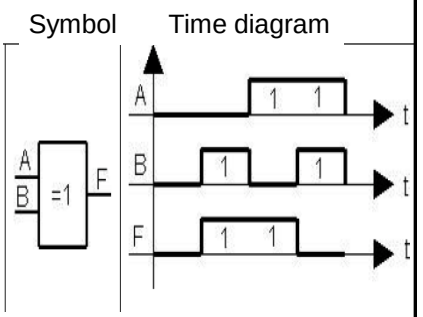
ANTIVALENCY, EXCLUSIVE-OR

A B	f_6^2
0 0	1
0 1	1
1 0	1
1 1	0

$f_6^2 = \overline{A}\,B + A\,\overline{B}$

usual notation:

$f_6^2 = A \oplus B$



ANTIVALENCY or EXCLUSIVE-OR, the function is1, if ane of the variables 1 while the other is 0, and is 0, if both variables are simultaneously 0 or 1.

32

EQUIVALENCY, EXCLUSIVE-NOR

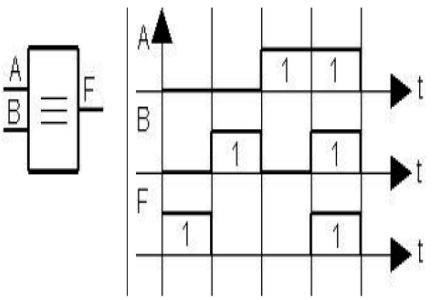
A	B	f_9^2
0	0	1
0	1	0
1	0	0
1	1	1

$f_9^2 = AB + \overline{A}\,\overline{B}$

usual symbol:

$f_9^2 = A \odot B$

Symbol Time diagram



EQUIVALENCY (EXCLUSIVE-NOR), the function is1, if both variables ar simultaneously 0 or 1, and is 0 if one of the variables is 0 and the other is 1.

33

ANTIVALENCY

A	B	f_6^2
0	0	1
0	1	1
1	0	1
1	1	0

The $f_6^2 = A \oplus B$ operation realizes also mathematical operation of binary addition of two bits without carry (binary half-adder).

The antivalency operation or gate can also be considered as a digital 1-bit comparator.
By appropriate chaining n-bit comparators can be constructed.

34

EXLUSICE-OR: IMPLEMENTATION

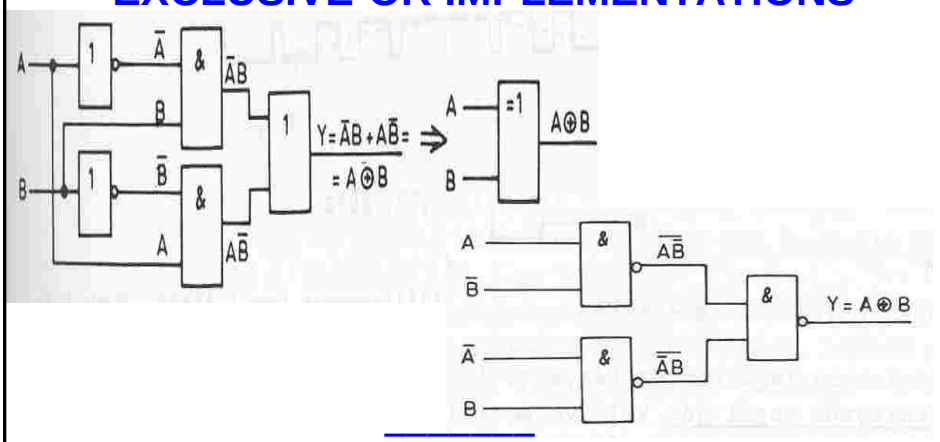
The EXCLUSIVE-OR operation can be implemented ousing AND, OR gates and inverters (NOT) based on the Boolean equation.

Or it can be implemented, after appropriate transformation using the universal NAND or NOR elements.

In the TTL and CMOS modular logic families EXCLUSIVE-OR gates are also available.

35

EXCLUSIVE-OR IMPLEMENTATIONS



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

It is supposed that the negated imput variables are available.

36

EXCLUSIVE-NOR: EQUIVALENCY

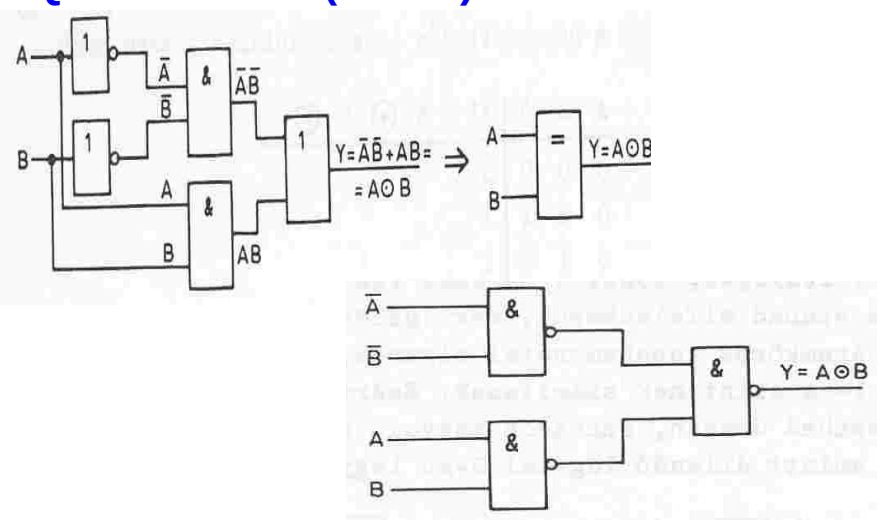
$$A \odot B = \bar{A}\,B + A\,\bar{B}$$

According to the truth table XNOR is the complemenet (negated) of XOR.

When extending to more the two variables, several different definitions are used! E.g. According to MSz (Hungarian Standards) for three variables the value of XNOR is 1 only for the input combinations 000 and 111, and is 0 for all other combination.
The other usual definition assigns a value of 1 to the output if even number of inputs is 1.

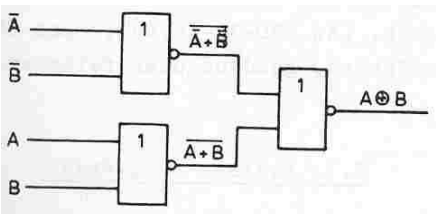
37

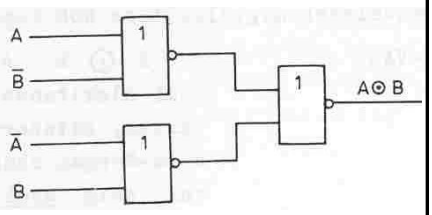
EQUIVALENCY (XNOR) IMPLEMENTATIONS



38

XOR, XNOR REALIZATIONS WITH NOR





ANTIVALENCY (XOR) and EQQUIVALENCY (XNOR) realizations with homogeneous NOR gate circuitry.

39

TWO-VARAIBLE FUNCTIONS: INHIBITION AND IMPLICATION

Function name	f(A,B)
Logical constants	0, 1
One-variable functions	$A, \overline{A}, B, \overline{B}$
AND, OR, NAND, NOR	$A \bullet B, A+B, \overline{A \bullet B}, \overline{A+B}$
XOR ($A \oplus B$), XNOR ($A \odot B$)	$\overline{A}\,B+A\,\overline{B}$, $\overline{A}\,\overline{B}+A\,B$
INHIBITION	$A \supset B, B \supset A$
IMPLICATIION	$A \rightarrow B, B \rightarrow B$

40

TWO-VARAIBLE FUNCTIONS: INHIBITION AND IMPLICATION

Function name	f(A,B)
Logical constants	0, 1
Functions of one variable	A, $\overline{A}$, B, $\overline{B}$
AND, OR, NAND, NOR	A•B, A+B, $\overline{A \cdot B}$, $\overline{A+B}$
XOR (A⊕B), XNOR (A⊙B)	$\overline{A}$ B+A $\overline{B}$, $\overline{A}$ $\overline{B}$+A B
INHIBITION	A ⊃ B, B ⊃ A
IMPLICATION	A → B, B → A

41

INHIBITION AND IMPLICATION

Details on these four functions can be found in the texts. They play a role in formal logic, and less in digital technics.

The four last functions:

A ⊃ B	A INHIBITS B
B ⊃ A	B INHIBITS A
A → B	IF A THEN B TOO
B → B	IF B THEN A TOO

42

TWO-VARIABLE FUNCTIONS: RECAPITULATION AND SUMMARY

Function name	f(A,B)
Logical constants	0, 1
Functions of one variable	A, $\overline{A}$, B, $\overline{B}$
AND, OR, NAND, NOR	A•B, A+B, $\overline{A \cdot B}$, $\overline{A+B}$
XOR (A⊕B), XNOR (A⊙B)	$\overline{A}$ B+A $\overline{B}$, $\overline{A}$ $\overline{B}$+A B
INHIBITION	$A \supset B$, $B \supset A$
IMPLICATION	$A \rightarrow B$, $B \rightarrow A$

43

BOOLEAN FUNCTIOS AND OPERATIONS OF TWO VARIABLES: A SUMMARY

From the 16 possible two-variable Boolean functions

6 can be considered as trivial
(2 of them are constants, 4 of them are in fact one-variable functions)

From the 10 non-trivial functions
2 (AND and OR) and their complementary (AND-NOT and OR-NOT)
as well as the EXCLUSIVE-OR (antivalency) and the EXCLUSIVE-NOR (equivalency)
are of significance for the practice.

44

OUTLOOK: IMPLEMENTATION 74HC/HCT181

Total 16 arithmetic operations (add, subtract, plus, shift, plus 12 others)
- Total 16 logic operations (XOR, AND, NAND, NOR, OR, plus 11 others)
- Capable of active-high and active-low operation

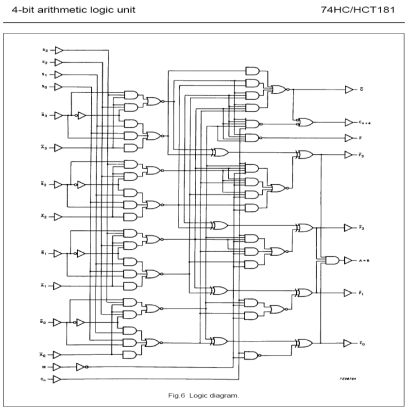
FUNCTION TABLES

MODE SELECT INPUTS				ACTIVE HIGH INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M=H)	ARITHMETIC(2) (M=L; C_n=H)
L	L	L	L	$\overline{A}$	A
L	L	L	H	$\overline{A+B}$	A + B
L	L	H	L	$\overline{AB}$	A + $\overline{B}$
L	L	H	H	logical 0	minus 1
L	H	L	L	$\overline{AB}$	A plus $A\overline{B}$
L	H	L	H	$\overline{B}$	(A + B) plus $A\overline{B}$
L	H	H	L	$A \oplus B$	A minus B minus 1
L	H	H	H	$A\overline{B}$	$A\overline{B}$ minus 1
H	L	L	L	$\overline{A}+B$	A plus AB
H	L	L	H	$\overline{A \oplus B}$	A plus B
H	L	H	L	B	(A + $\overline{B}$) plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	logical 1	A plus A(1)
H	H	L	H	A + $\overline{B}$	(A + B) plus A
H	H	H	L	A + B	(A + $\overline{B}$) plus A
H	H	H	H	A	A minus 1

MODE SELECT INPUTS				ACTIVE LOW INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M=H)	ARITHMETIC(2) (M=L; C_n=L)
L	L	L	L	$\overline{A}$	A minus 1
L	L	L	H	$\overline{AB}$	AB minus 1
L	L	H	L	$\overline{A}+B$	$A\overline{B}$ minus 1
L	L	H	H	logical 1	minus 1
L	H	L	L	$\overline{A+B}$	A plus (A + $\overline{B}$)
L	H	L	H	$\overline{B}$	AB plus (A + $\overline{B}$)
L	H	H	L	$\overline{A \oplus B}$	A minus B minus 1
L	H	H	H	A + $\overline{B}$	A + $\overline{B}$
H	L	L	L	$\overline{AB}$	A plus (A + B)
H	L	L	H	$A \oplus B$	A plus B
H	L	H	L	B	$A\overline{B}$ plus (A + B)
H	L	H	H	A + B	A + B
H	H	L	L	logical 0	A plus A(1)
H	H	L	H	AB	AB plus A
H	H	H	L	AB	$A\overline{B}$ plus A
H	H	H	H	A	A

Notes to the function tables

1. Each bit is shifted to the next more significant position.
2. Arithmetic operations expressed in 2s complement notation.



74HC/HCT181 ARITHMETIC LOGIC UNIT

FUNCTION TABLES

MODE SELECT INPUTS				ACTIVE HIGH INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M=H)	ARITHMETIC(2) (M=L; C_n=H)
L	L	L	L	$\overline{A}$	A
L	L	L	H	$\overline{A+B}$	A + B
L	L	H	L	$\overline{AB}$	A + $\overline{B}$
L	L	H	H	logical 0	minus 1
L	H	L	L	$\overline{AB}$	A plus $A\overline{B}$
L	H	L	H	$\overline{B}$	(A + B) plus $A\overline{B}$
L	H	H	L	$A \oplus B$	A minus B minus 1
L	H	H	H	$A\overline{B}$	$A\overline{B}$ minus 1
H	L	L	L	$\overline{A}+B$	A plus AB
H	L	L	H	$\overline{A \oplus B}$	A plus B
H	L	H	L	B	(A + $\overline{B}$) plus AB
H	L	H	H	AB	AB minus 1
H	H	L	L	logical 1	A plus A(1)
H	H	L	H	A + $\overline{B}$	(A + B) plus A
H	H	H	L	A + B	(A + $\overline{B}$) plus A
H	H	H	H	A	A minus 1

MODE SELECT INPUTS				ACTIVE LOW INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC (M=H)	ARITHMETIC(2) (M=L; C_n=L)
L	L	L	L	$\overline{A}$	A minus 1
L	L	L	H	$\overline{AB}$	AB minus 1
L	L	H	L	$\overline{A}+B$	$A\overline{B}$ minus 1
L	L	H	H	logical 1	minus 1
L	H	L	L	$\overline{A+B}$	A plus (A + $\overline{B}$)
L	H	L	H	$\overline{B}$	AB plus (A + $\overline{B}$)
L	H	H	L	$\overline{A \oplus B}$	A minus B minus 1
L	H	H	H	A + $\overline{B}$	A + $\overline{B}$
H	L	L	L	$\overline{AB}$	A plus (A + B)
H	L	L	H	$A \oplus B$	A plus B
H	L	H	L	B	$A\overline{B}$ plus (A + B)
H	L	H	H	A + B	A + B
H	H	L	L	logical 0	A plus A(1)
H	H	L	H	AB	AB plus A
H	H	H	L	AB	$A\overline{B}$ plus A
H	H	H	H	A	A

Notes to the function tables

1. Each bit is shifted to the next more significant position.
2. Arithmetic operations expressed in 2s complement notation.

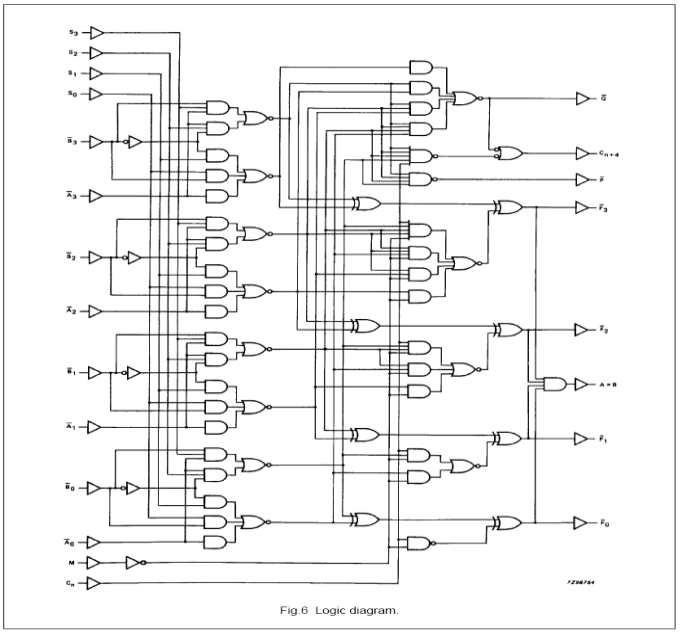


Fig.6 Logic diagram.

47

ARITHMETIC LOGIC UNIT (ALU): MAIN TECHNICAL CHARCTERISTICS

Typical characteristics (e.g. 74181 type):

4-bit word length (input: A, B, output: F)
16 arithmetic and 16 logic operations
4 selector input, 1 mode change input
Carry-in and carry out
Reltion (A=B) output
Carry look-ahead (G, P) output

48

WAYS TO SPECIFY LOGIC FUNCTIONS

A logic function can be specified in various different ways.
The possibilities listed below will be discussed here.

1. Truth table
2. Algebraic form
3. Graphic representation
4. Symbolic representation

49

SPECIFICATION OF LOGIC FUNCTION: TRUTH TABLE AND ALGEBRAIC FORM

i	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

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

Unambiguous definition of a logic function: Prescription of the value of the function for each and all possible combinations of the independent variables. This is the thruth table.
This kind of definition of a function is unambigouos.

50

SPECIFICATION OF FUNCTION IN ALGEBRAIC FORM: THE CANONIC FORM

A logic function can specified using algebra in such a way that it is described by the symbols of logic operations (AND, OR and NOT).

The fist step of synthesis of combinational networks is the formulation of the relevant logic function on the basis of the problem/task to be solved. Usually the algebraic form is used.

A logic function can be specified in several algebraic forms. Among the algebraic forms the normalized or canonic forms have specific relevance.

51

GRAPHIC REPRESENTATION

Based on the truth table the values of logic functions can be graphically represented (mapped) in various forms of tables or maps (Karnaugh maps or Veitch diagrams).

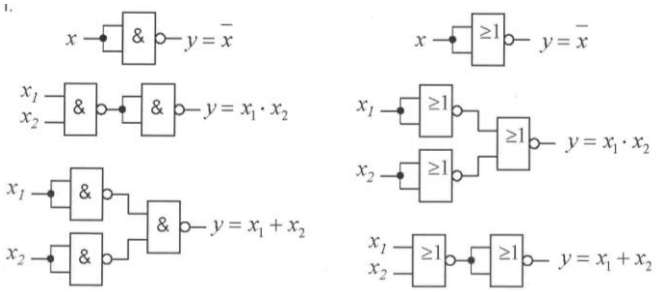
This kind of representation is very useful for the cases when the number of variables is limited (say less than five or six).

52

SYMBOLIC REPRESENTATION

The logic operations can be designated by symbols (e.g. symbols on a drawing). In electronic logic systems each symbol represents a circuit performing a given logic operation.

Therefore the such logic symbols give also information on the circuit implementation too.



53

CANONIC FORMS OF LOGIC FUNCTIONS

The following presentation and discussion of canonic (algebraic) forms of logic functions is based on Arató's text.

Disjunctive canonic form or extended sum-of-product form.

Conjunctive canonic form or extended product-of sum form.

54

COMPLETELY SPECIFIED LOGIC FUNCTION (LOGIC PROBLEM)

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

Completely determined logic function can be specified by listing all those input combinations for which the value of the function is F = 1,
or listing all those for which F = 0.

$$F(ABC) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + AB\bar{C}$$

or

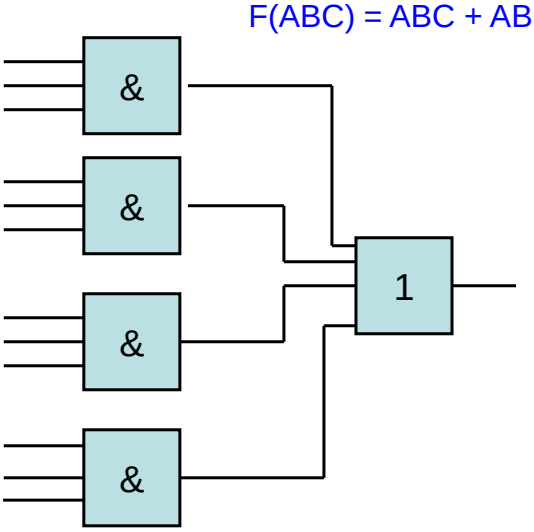
$$F(ABC) = \overline{A\,B\,C} + \bar{A}\,BC + A\bar{B}C + ABC$$

$\bar{F}$ and F both can be represented as sums of logic products (SOP).

55

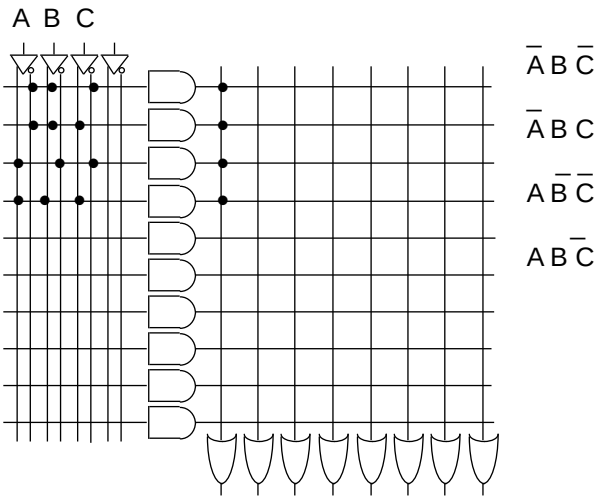
TWO-LEVEL AND-OR REALIZATION (SSI MODULAR LOGIC)

$$F(ABC) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + AB\bar{C}$$



56

PLA IMPEMENTATION



PLA: a plane of AND gates followed with a plane of OR gates, interconnections can be established by the user.

57

INCOMPLETELY SPECIFIED LOGIC FUNCTION

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

Incompletely specified logic function, there are such input combinations to which no value of the function is specified.

Her e.g. the truth table specifies four different but fully specified logic function, each of which solves the logic problem specified by the table. Assigning values to the two unspecified (don't care) terms the function can be made completely specified in four different ways.

The selection of the "best" or most appropriate network should be made during the synthesis process.

58

INCOMPLETELY SPECIFIED LOGIC FUNCTION

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

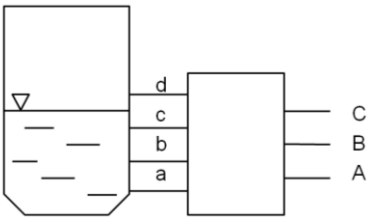
Algebraic notation: the don't care terms are in parantheses:

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

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

59

INCOMPLETELY SPECIFIED LOGIC FUNCTION: EXAMPLE



Liquid level indicator using four sensors:

Level indication:	below a
	between a and b
	between b and c
	between c and d
	above b

E.g. if dry = 0, wet = 1, then abcd = 0 0 0 1 cannot occur!

60

CANONIC FORMS OF LOGIC FUNCTIONS

In the synthesis of combinational networks it is expedient to start from the algebraic form. Because a logic function can have various different but equivalent algebraic form, it is necessary to use a special algebraic form which has the property which cannot be attributed to any other equivalent form. Such algebraic form is called normal or canonic form of the logic function.

There exist two such forms:

the disjunctive canonic form or minterm form

and

the conjuctive canonic form or maxterm form.

61

DISJUNCTIVE CANONIC FORM (EXTENDED SUM-OF-PRODUCTS)

The algebraic form constructed from the truth table as logic sum of logic products (AND-OR) is (disjunctive) canonic form.

Previous example, properties of the function:

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

- Each product term represents an input variable combination for which the function is F = 1;
- Each product term contains each and all variables either in asserted or in negated form.

A completely specified logic function has only one (unique) such algebraic form, the disjunctive canonic form.

62

MINTERM (DEFINITION)

The terms of the disjunctive canonic form are called minterm

m_i^n here n is the number of independent variables,

i (minterm index) is the decimal value of the binary number corresponding to the given combination of the independent variables.

$$F(ABC) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C}$$

(*remember*: (2) (3) (4) (6))

$$F(ABC) = m_2^3 + m_3^3 + m_4^3 + m_6^3$$

Other notation:

$$F = \sum^3 (2,3,4,6)$$

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

CONJUNCTIVE CANONIC FORM (1)

To find the conjunctive canonic form, at first consider the negated function from the truth table

$$\overline{F(ABC)} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$\overline{F(ABC)} = m_0^3 + m_1^3 + m_5^3 + m_7^3$$

The negated function consists of those minterms, which are not contained in the function. (Note that this is true only for completely specified functions!)

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

64

CONJUNCTIVE CANONIC FORM (1)

Based on De Morgan's law the conjunctive canonic form of F can be obtained from the negated function by appropriate transformation resulting in a product-of-sums (POS) i.e. in a product of maxterms.

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

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

$$F(ABC) = M_7{}^3 M_6{}^3 M_2{}^3 M_0{}^3$$

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

65

CONNECTION BETWEEN MINTERMS AND MAXTERMS

Original function, disjunctive canonic form (SOP)

$$F(ABC) = m_2{}^3 + m_3{}^3 + m_4{}^3 + m_6{}^3$$

Negated function, disjunctive form (index i)

$$\overline{F(ABC)} = m_0{}^3 + m_1{}^3 + m_5{}^3 + m_7{}^3$$

Original function, conjunctive canonic form (POS), (index $I = 2^3 - i$)

$$F(ABC) = M_7{}^3 M_6{}^3 M_2{}^3 M_0{}^3$$

66

MINTERMS AND MAXTERMS

Relationship between the minterm indexes i of the negated function and the maxterm I of the asserted function (written for the case of three variables)

$$i + I = 7 = 2^3 - 1$$

For n-variable function

$$i + I = 2^n - 1$$

67

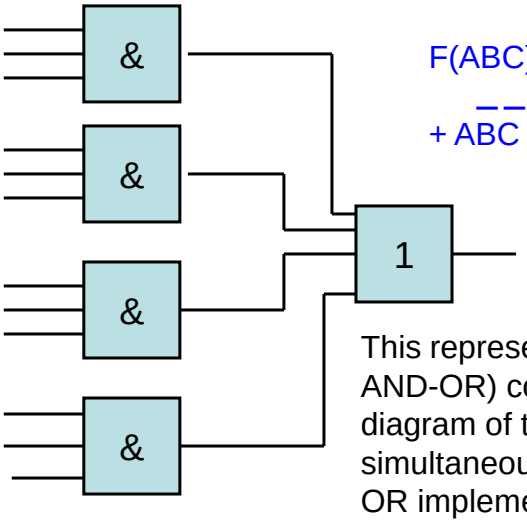
CONCEPTUAL LOGIC DIAGRAM AND TWO-LEVEL REALIZATION

All logic functions can be specified using AND, OR and NOT (inverter) operations. Not including the inverters (NOT) to obtain the negated values of the input variables, both canonic forms (extended SOP and POS) can specified and implemented by two-level AND-OR or OR-AND gate networks respectively (conceptual logic diagram).

Because the AND and OR (and the NOT too) can be implemented using either only NAND or only NOR gates, then based on the respective canonic forms all logic functions can be implemented with homogeneous two-level NAND gate or NOR gate networks.

68

TWO-LEVEL AND-OR IMPLEMENTATION OF F(A,B,C)

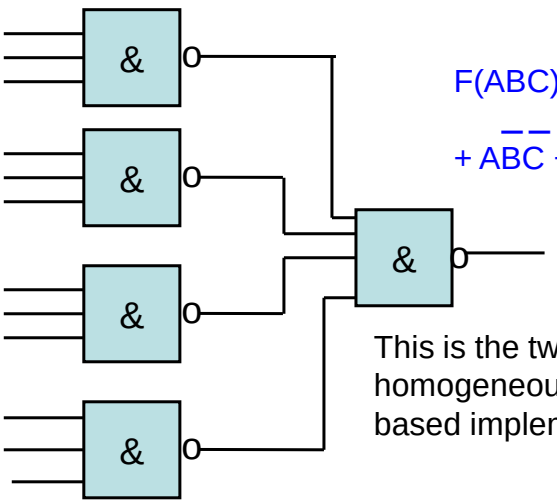


$F(ABC) = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C$

This represents a (two-level AND-OR) conceptual logic diagram of the function, and simultaneously a two level AND-OR implementation.

69

TWO-LEVEL HOMOGENEOUS NAND GATE IMPLEMENTATION OF F(A,B,C)



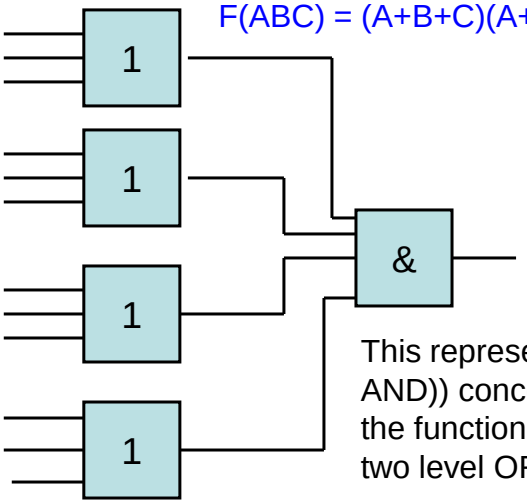
$F(ABC) = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C$

This is the two-level homogeneous NAND gate based implementation.

70

TWO-LEVEL OR-AND IMPLEMENTATION OF F(A,B,C)

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

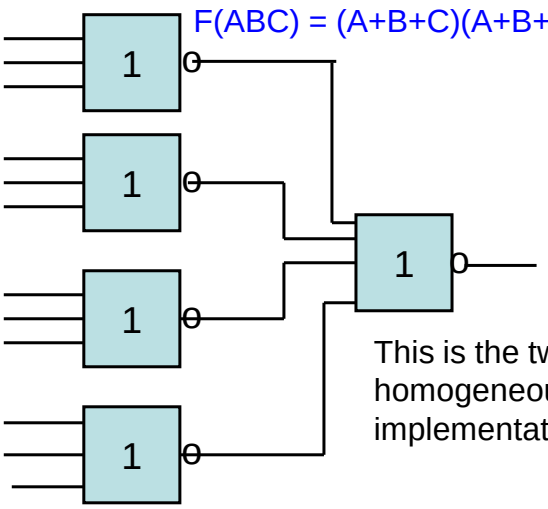


This represents a (two-level OR-AND)) conceptual logic diagram of the function, and simultaneously a two level OR-AND implementation.

71

TWO LEVEL HOMOGENEOUS NOR GATE IMPLEMENTATION OF F(A,B,C)

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



This is the two-level homogeneous NOR gate based implementation.

72

CONCEPTUAL LOGIC DIAGRAM AND TWO-LEVEL REALIZATION: EMPHASIS

All logic functions can be specified using AND, OR and NOT (inverter) operations. Not including the inverters (NOT) to obtain the negated values of the input variables, both canonic forms (extended SOP and POS) can specified and implemented by two-level AND-OR or OR-AND gate networks respectively (conceptual logic diagram).

Because the AND and OR (and the NOT too) can be implemented using either only NAND or only NOR gates, then based on the respective canonic forms all logic functions can be implemented with homogeneous two-level NAND gate or NOR gate networks.

73

REVIEW QUESTIONS

1. 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.)

74

REVIEW QUESTIONS

2. Write both sum-of-products and product-of-sums Boolean expressions for (a) a two-input AND gate, (b) a two-input NAND-gate, (c) a two-input EX-OR gate and (d) a two-input NOR gate from their respective truth tables.

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

75

REVIEW QUESTIONS

4. 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.)

76

PROBLEMS AND EXERCISES

1. Transform and simplify the Boolean expressions given below using algebraic methods. The final result(s) should be in the form of sum of products of logical variables.

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

(X + Y Z) (X Y + X Y) (ANSWER : **X Y Z**)

($\overline{X}$ + Z) (X Y + X $\overline{Z}$) (ANSWER: **X Z + X Y Z**)

($\overline{X\,Y}$ + A) ($\overline{X\,Z}$ + Y) (ANSWER: **X + $\overline{Y}$ Z + $\overline{A}$ $\overline{Z}$ + A $\overline{Y}$**)

77

PROBLEMS AND EXERCISES

2. Design a logic circuit with two inputs, A and B, and two outputs, X and Y, so that it operates as follows:

(1) X and Y are both HIGH as long as A is HIGH, regardless of the level of B.
(2) If A pulses LOW, the LOW will appear at X if B=0 or at Y if B=1.

3. Find the canonic algebraic forms of the logic function

F(A,B,C) = A B + A C

4. List the minterm and maxterm indices of the logic function below

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

78

PROBLEMS AND EXERCISES

5. Four large tanks at a chemical plant contain different liquids being heated. Liquid-level sensors are being used to detect whenever the level in tank A and B rise above a predetermined level. Temperature sensors in tanks C and D detect when the temperature in these tanks drop below a prescribed temperature limit. Assume that the liquid-level sensor outputs A and B are *low* when the level is satisfactory and *high* when the level is to high. Also, the temperature-sensor outputs C and D are low when the temperature is satisfactory and high when the temperature is too low.
Design a logic circuit that will detect whenever the level in tank A or tank B is too high at the same time that the temperature in either tank C or tank is too low.

(Adapted from R. J. Tocci: *Digital Systems, Principles and Applications*)

79

END

80