


Digital Circuits



License: <http://creativecommons.org/licenses/by-nc-nd/3.0/>

Logic Gates

Logic gates are physical devices which implement simple Boolean functions.


Some of the simple gates:

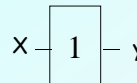
ANSI/IEEE-1973

ANSI/IEEE-1984

BUFFER

$Y=X$

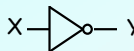


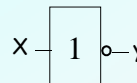


X	Y
0	0
1	1

INVERTER (NOT)

X'




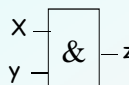


X	Y
0	1
1	0

AND

$X \cdot Y$




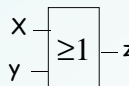


X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR

$X + Y$





X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

<http://www.faculty.itu.edu.tr/buzluca>

<http://www.buzluca.info>

© 2011 Dr. Feza BUZLUCA

3.1

Digital Circuits

NAND
(NOT AND)

$(xy)'$

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR
(NOT OR)

$(x+y)'$

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

XOR (Difference)
 $xy' + x'y$

$X \oplus Y$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

XNOR (Equality)
 $xy + x'y'$

$X \odot Y$

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

<http://www.faculty.itu.edu.tr/buzluca>

<http://www.buzluca.info>

© 2011 Dr. Feza BUZLUCA

3.2

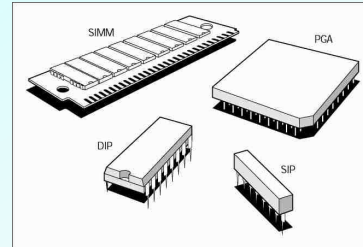
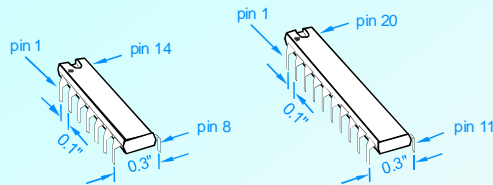
Integrated Circuits - IC

Logic gates are manufactured in a form of integrated circuits (chips).

Recently, a large number of mixed logic gates are packed into a single integrated circuit.

ICs are manufactured in different shapes.

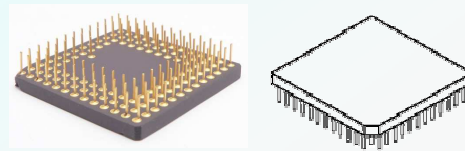
Dual in-line Package (DIP) ICs



Quad Flat Pack (QFP)



Pin Grid Array (PGA)

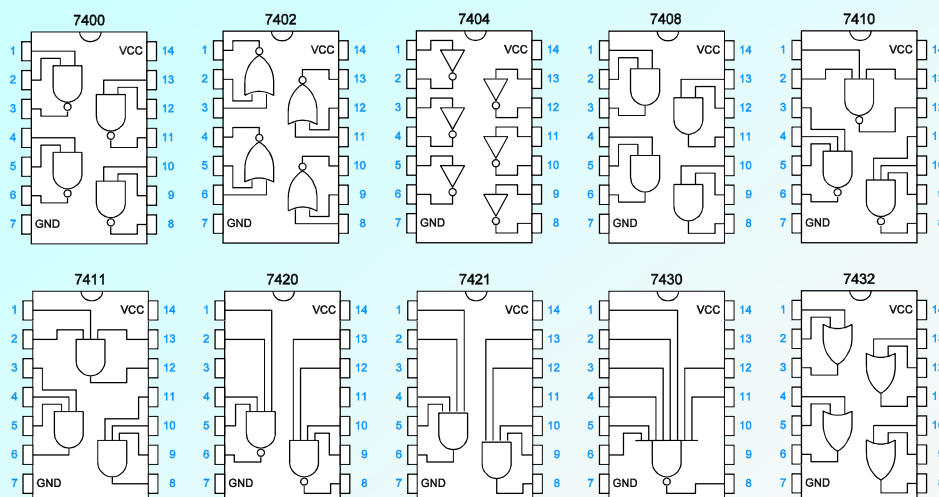


<http://www.faculty.itu.edu.tr/buzluca>
<http://www.buzluca.info>

© 2011 Dr. Feza BUZLUCA

3.3

Examples of 74xx Series



You may find necessary information about ICs in their datasheet catalogs.

<http://www.faculty.itu.edu.tr/buzluca>
<http://www.buzluca.info>

© 2011 Dr. Feza BUZLUCA

3.4

Positive and Negative Logic

Boolean values zero and one represent physical quantities such as voltage or state of an entity (door is open, light is off).

Assigning "1" to high value, and "0" to low value is called **positive logic**, and assigning "0" to high value, and "1" to low value is called **negative logic**.

Example:

Function table of a physical device with 2 inputs and one output is shown below.

If we use the positive logic the device can be implemented with an AND gate.

In negative logic system the device is implemented with an OR gate.

Physical Device			Positive Logic			Negative Logic		
Inputs:		Output:	Inputs:		Output:	Inputs:		Output:
x1	x2	z	x1	x2	z	x1	x2	z
L	L	L	0	0	0	1	1	1
L	H	L	0	1	0	1	0	1
H	L	L	1	0	0	0	1	1
H	H	H	1	1	1	0	0	0

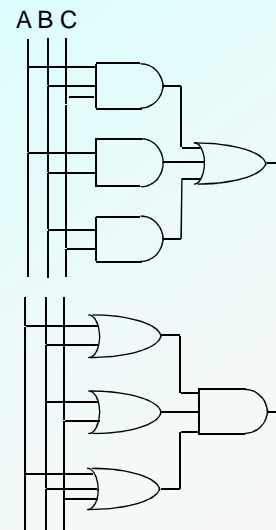
Implementation of Boolean Functions Using Logic Gates

Sum of Products (SoP)

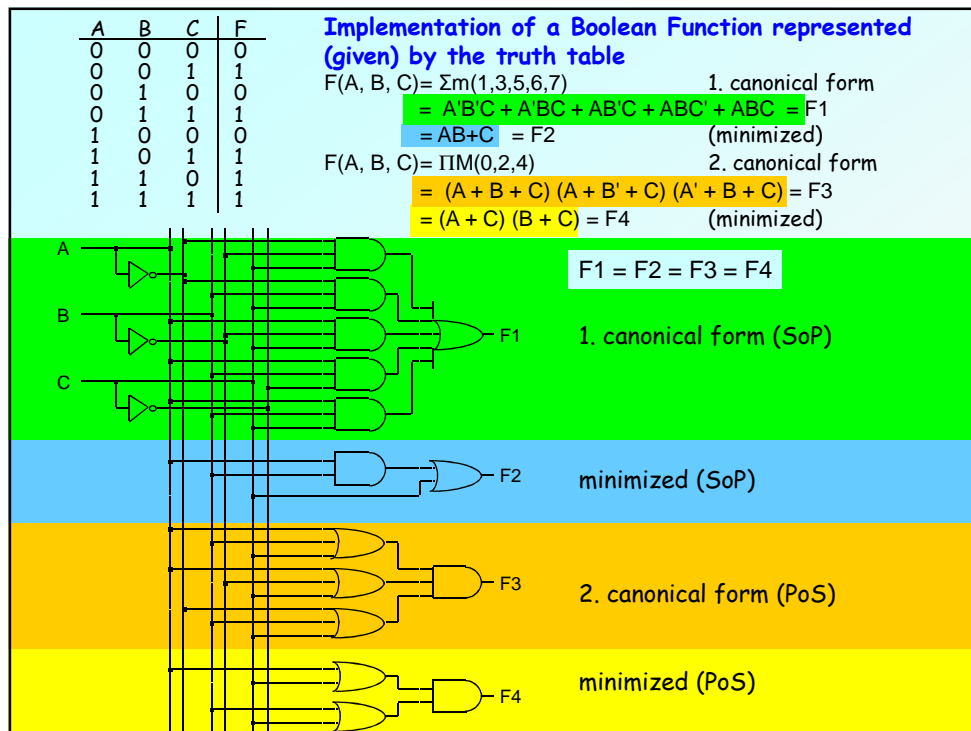
- AND gates implement the products.
- OR gate implements the sum.

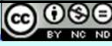
Product of Sums (PoS)

- OR gates implement the sums.
- AND gate implements the product.



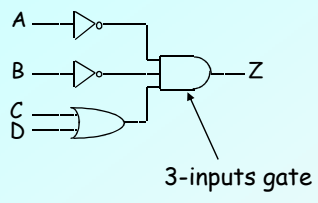
NOT gates can be also used, where they are necessary.



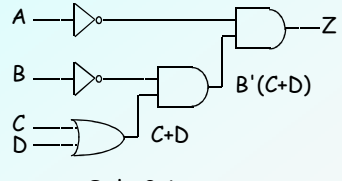
Digital Circuits  License: <http://creativecommons.org/licenses/by-nc-nd/3.0/>

As a Boolean function has many different expressions it can be implemented in different ways using different logic gates.

Example: $Z = A' \cdot B' \cdot (C + D) = (A' \cdot (B' \cdot (C + D)))$



3-inputs gate



Only 2-input gates

Sometimes it would be necessary to manipulate logical expressions of functions according to currently available gates.

<http://www.faculty.itu.edu.tr/buzluca>
<http://www.buzluca.info>

© 2011 Dr. Feza BUZLUCA 3.8

Functional completeness and Universal Gates

A **functionally complete set** of logical operators is one, which can be used to express all possible truth tables by combining members of the set.

From the definition of the Boolean algebra AND, OR, NOT is a complete set.

- The set consisting only of the binary operator NAND is also functionally complete.
- NAND gates alone can be used to reproduce the functions of all the other logic gates.
- By using only NAND gates all logic functions can be implemented.
- Similarly, the set consisting only of the binary operator NOR is also functionally complete.
- NOR gates alone can be used to reproduce the functions of all the other logic gates.

NAND (and also OR) gates are also called **universal logic gates**.

Proof of completeness

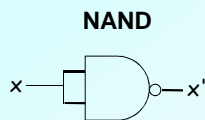
To prove that NAND and NOR operators are functionally complete, we have to show that AND, OR, NOT operations can be implemented by using only NAND (or alternatively NOR) gates.

NAND is denoted by symbol $|$

NOR is denoted by symbol \downarrow

• NOT:

$$\begin{aligned} x' &= x | x \\ &= (x \cdot x)' \\ &= x' \end{aligned}$$



• AND:

$$x \cdot y = (x | y)'$$

• OR:

$$x + y = (x' | y') \text{ de Morgan}$$

NOR

$$x' = x \downarrow x$$



$$x \cdot y = (x' \downarrow y') \text{ de Morgan}$$

$$x + y = (x \downarrow y)'$$

Relation between NAND and NOR

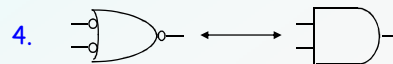
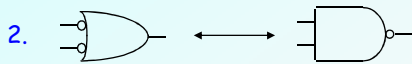
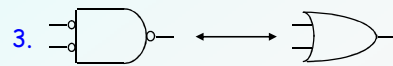
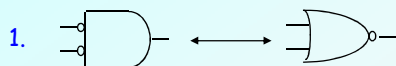
■ NAND - NOR Conversions

■ de Morgan:

1. $(A + B)' = A' \cdot B'$
2. $(A \cdot B)' = A' + B'$
3. $(A' \cdot B')' = A + B$
4. $(A' + B')' = (A \cdot B)$

■ These expressions show that:

1. An AND gate with inverted inputs is the equivalent of the NOR gate.
2. An OR gate with inverted inputs is the equivalent of the NAND gate.
3. A NAND gate with inverted inputs is the equivalent of the OR gate.
4. A NOR gate with inverted inputs is the equivalent of the AND gate.



Implementation of Boolean functions using only NAND (NOR) gates

As the binary operator NAND is functionally complete all Boolean functions can be implemented by using only NAND gates.

The same property is also valid for NOR gates.

Implementation of Boolean functions in the SOP form using only NAND gates

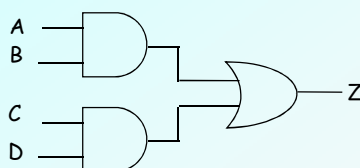
Shortcut: In circuits in the SOP form all AND gates and OR gates can be replaced with NAND gates. These changes do not affect the output.

Details:

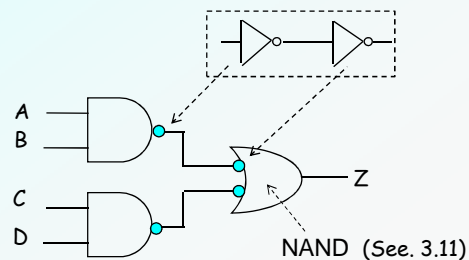
If we put NOT gates to the outputs of AND gates, and to the inputs of the OR gates we obtain NAND gates. (See 3.11)

If we connect two NOT gates one after the other the truth table of the function does not change. $(a')' = a$ (Involution)

Example: $Z = (A \cdot B) + (C \cdot D)$



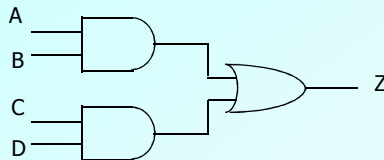
≡



Algebraic conversion:

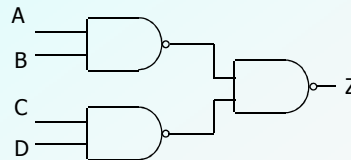
Expression is two times inverted. $(Z')' = Z$ (Involution)

$$\begin{aligned}
 Z &= (A \cdot B) + (C \cdot D) && \text{(SoP form)} \\
 &= [(A \cdot B) + (C \cdot D)]' && \\
 &= [(A \cdot B)' \cdot (C \cdot D)']' && \text{(De Morgan)} \\
 &= (A \mid B) \mid (C \mid D) && \text{(only NAND gates)}
 \end{aligned}$$

Algebraic verification:

?

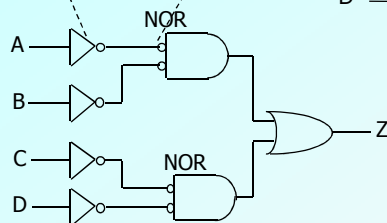
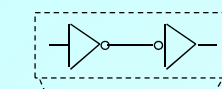
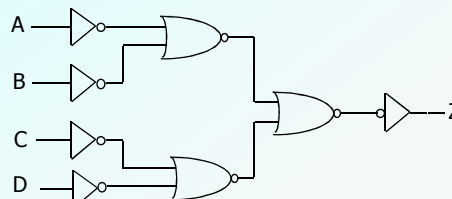
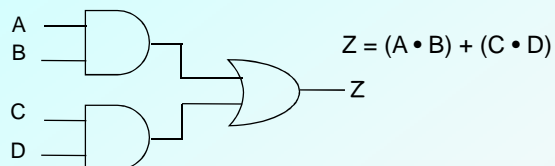
≡



$$\begin{aligned}
 Z &= [(A \cdot B)' \cdot (C \cdot D)']' \\
 &= [(A' + B') \cdot (C' + D')] \\
 &= [(A' + B') + (C' + D')] \\
 &= (A \cdot B) + (C \cdot D) \checkmark
 \end{aligned}$$

Implementation of Boolean functions in the SOP form using only NOR gates

In this case extra NOT gates must be connected to the inputs and outputs of the circuit.

1. Step2. Step

Remember: NOT gates can be implanted using NOR gates. $x \rightarrow x'$



Implementation of Boolean functions in the POS form using only NOR gates

Shortcut: In circuits in the POS form all AND gate and OR gates can be replaced with NOR gates.

Details:

If we put NOT gates to the outputs of OR gates, and to the inputs of the AND gates we obtain NOR gates. (See 3.11)

Remember: If we connect two NOT gates one after the other, the truth table of the function does not change. $(a')' = a$ (Involution)

Example: $Z = (A + B) \cdot (C + D)$

