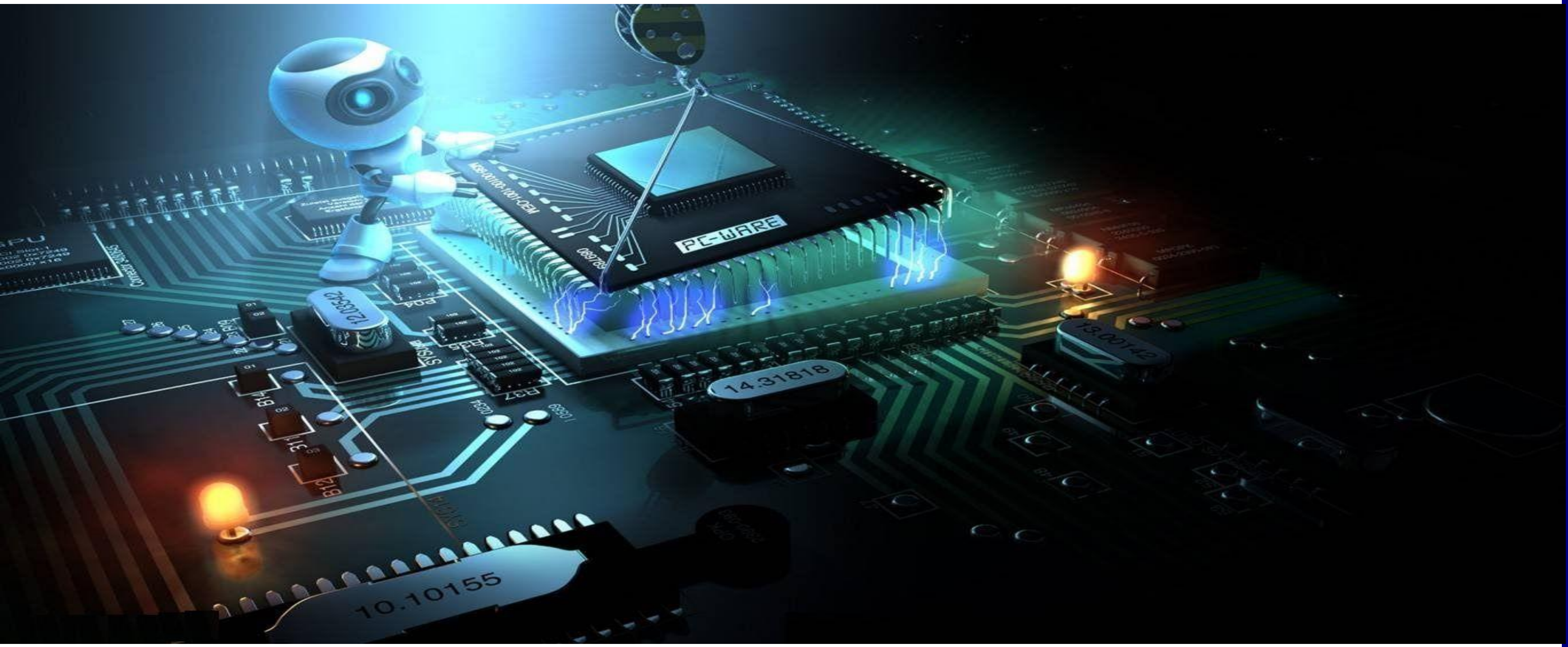
جامعة القاهرة الجديدة
التكنولوجية

Fall 2023

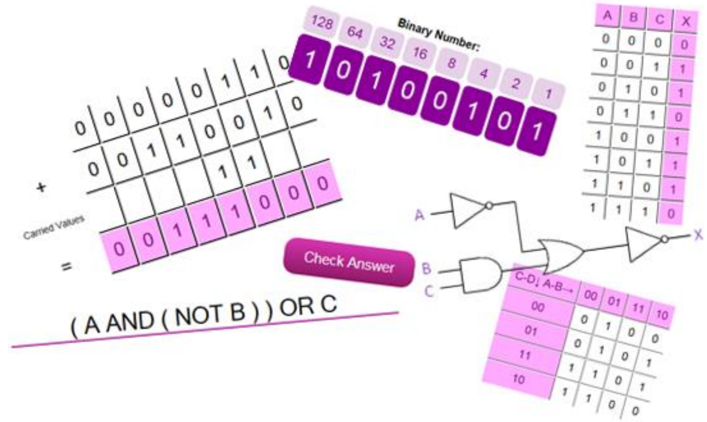
(2)



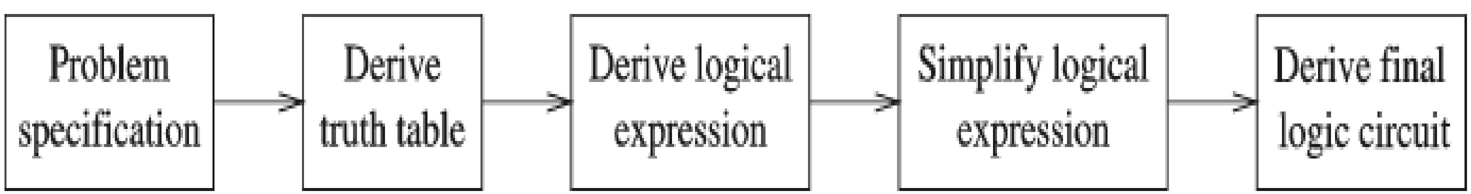
Digital Engineering

Dr. Hatem Yousry

Agenda

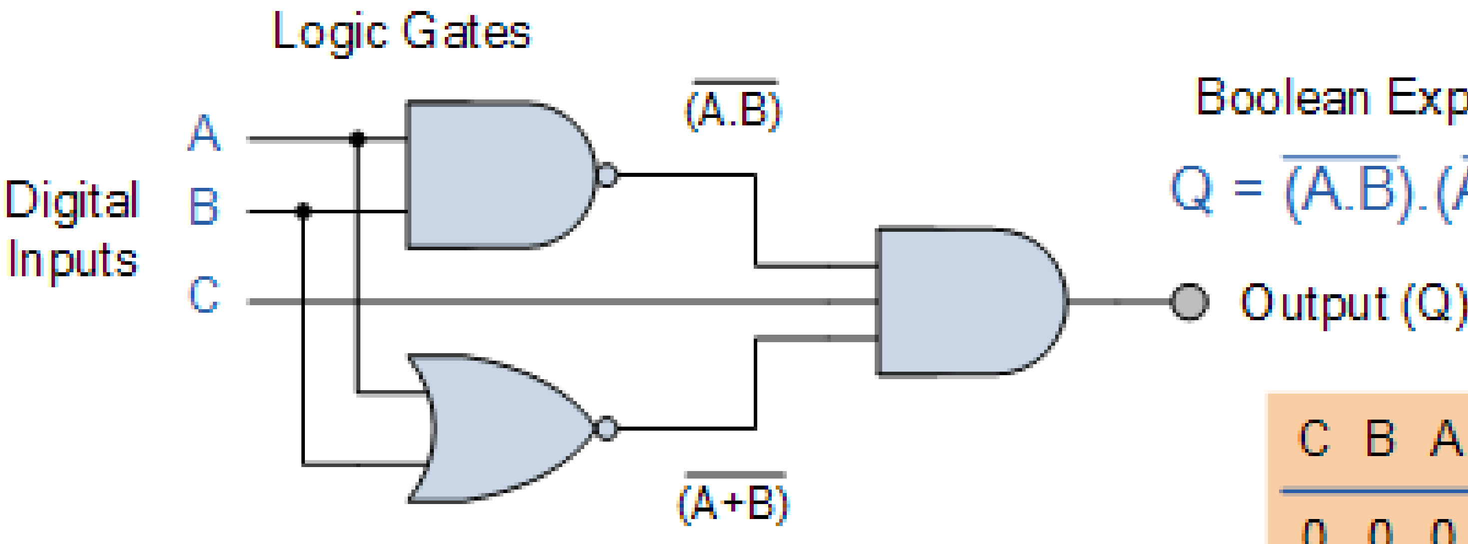


- Conversions.
- Complements.
- Boolean Algebra.



Digital Engineering (Logic)

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



Logic Diagram

Boolean Expression

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

Typical Truth Table

C	B	A	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Complements

- There are two types of complements for each **base-*r*** system: the radix (base) complement and diminished radix complement.
- **Diminished Radix Complement - (r-1)'s Complement**
  - Given a number *N* in base *r* having *n* digits, the (*r*–*1*)'s complement *of N* is defined as:

$$\boldsymbol{(r^n - 1) - N}$$

- **Example for 6-digit decimal numbers**:
  - 9's complement is $\boldsymbol{(r^n - 1) - N = (10^6 - 1) - N} = 999999 - N$
  - 9's complement of 546700 is 999999–546700 = 453299
- **Example for 7-digit binary numbers:**
  - 1's complement is $\boldsymbol{(r^n - 1) - N = (2^7 - 1) - N} = 1111111 - N$
  - 1's complement of 1011000 is 1111111–1011000 = 0100111
- **Observation:**
  - Subtraction from (r^n – 1) will never require a borrow
  - Diminished radix complement can be computed digit-by-digit
  - For binary: 1 – 0 = 1 and 1 – 1 = 0

Complements

- 1's Complement (*Diminished Radix* Complement)
  - All '0's become '1's
  - All '1's become '0's

Example $(10110000)_2$

⇨ $(01001111)_2$

If you add a number and its 1's complement …

Binary Logic

- Definition of Binary Logic
  - Binary logic consists of binary variables and a set of logical operations.
  - The variables are designated by letters of the alphabet, such as *A*, *B*, *C*, *x*, *y*, *z*, etc, with each variable having two and only two distinct possible values: 1 and 0,
  - Three basic logical operations: AND, OR, and NOT.

1. AND: This operation is represented by a dot or by the absence of an operator. For example, $x \cdot y = z$ or $xy = z$ is read "*x* AND *y* is equal to *z*," The logical operation AND is interpreted to mean that $z = 1$ if only $x = 1$ and $y = 1$; otherwise $z = 0$. (Remember that *x*, *y*, and *z* are binary variables and can be equal either to 1 or 0, and nothing else.)
2. OR: This operation is represented by a plus sign. For example, $x + y = z$ is read "*x* OR *y* is equal to *z*," meaning that $z = 1$ if $x = 1$ or $y = 1$ or if both $x = 1$ and $y = 1$. If both $x = 0$ and $y = 0$, then $z = 0$.
3. NOT: This operation is represented by a prime (sometimes by an overbar). For example, $x' = z$ (or $\bar{x} = z$) is read "not *x* is equal to *z*," meaning that *z* is what *z* is not. In other words, if $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$, The NOT operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1.

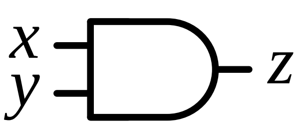
Binary Logic

- Truth Tables, Boolean Expressions, and Logic Gates

AND

x	*y*	*z*
0	0	0
0	1	0
1	0	0
1	1	1

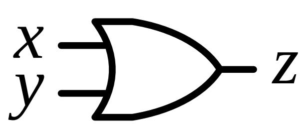
$$z = x \bullet y = x\,y$$



OR

x	*y*	*z*
0	0	0
0	1	1
1	0	1
1	1	1

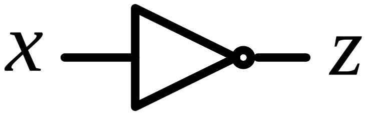
$$z = x + y$$



NOT

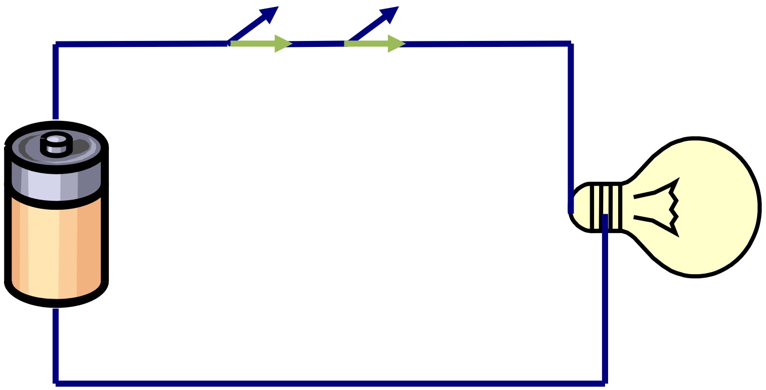
x	*z*
0	1
1	0

$$z = \overline{x} = x'$$

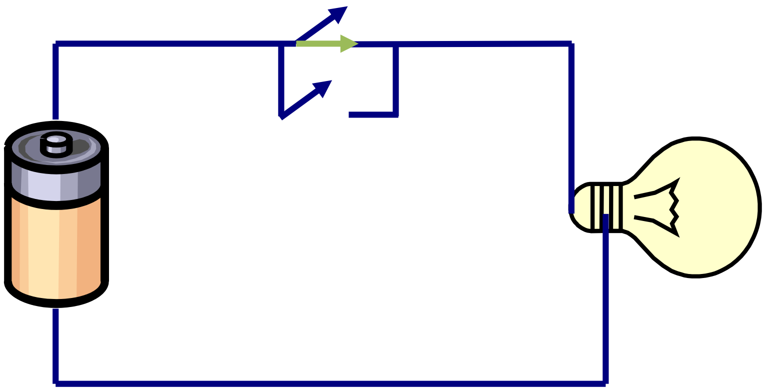


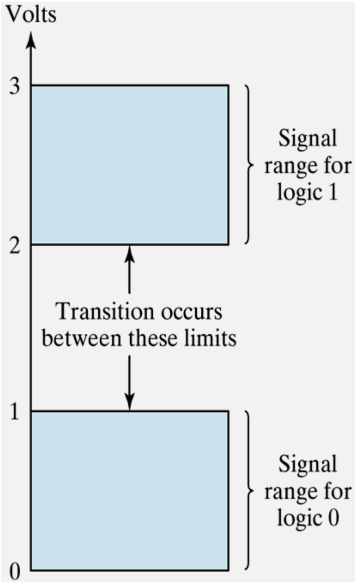
Switching Circuits

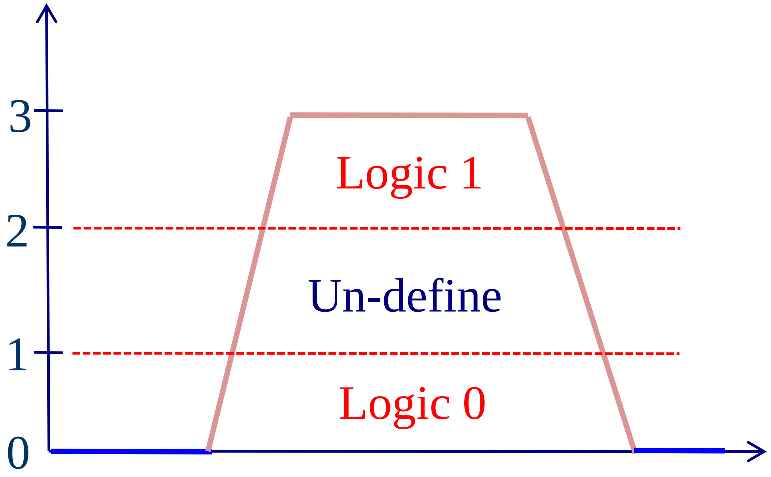
AND



OR

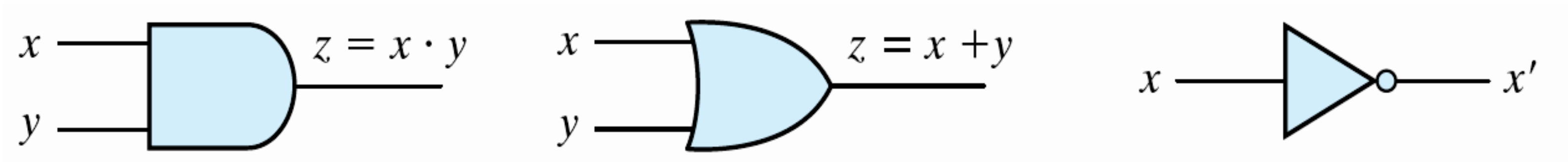




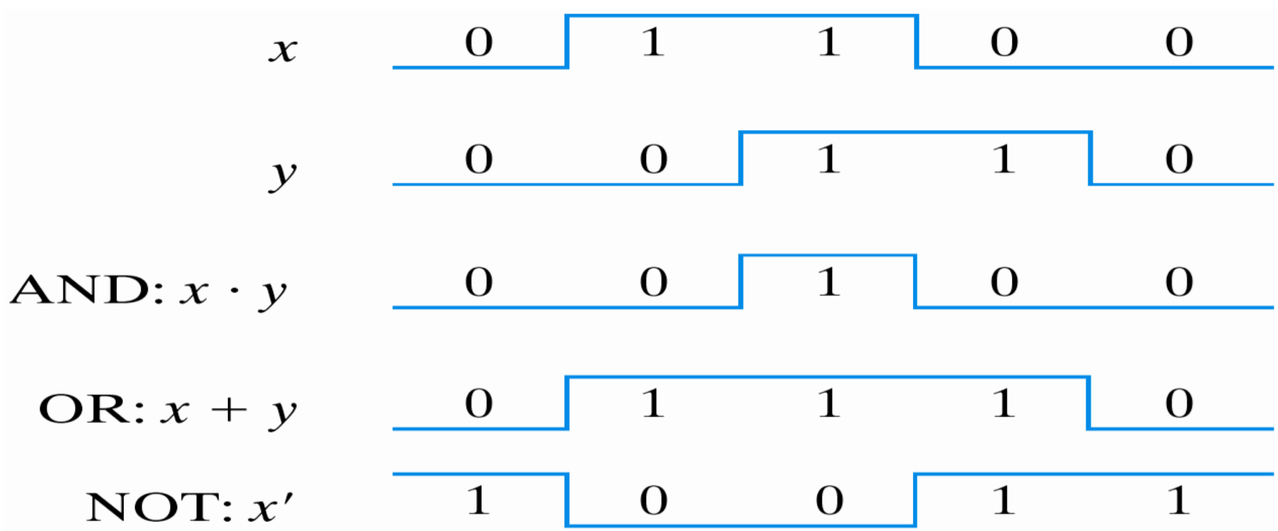


Binary Logic

- Logic gates
  - Graphic Symbols and Input-Output Signals for Logic gates:

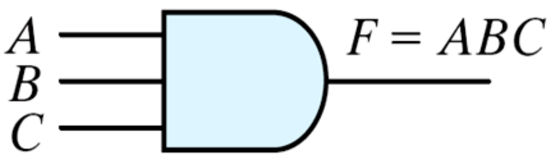


(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

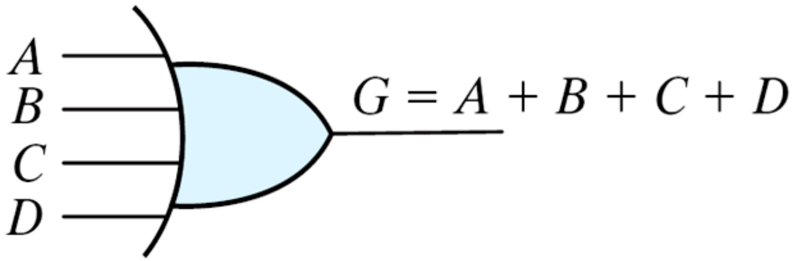


Binary Logic

- Logic gates
  - Graphic Symbols and Input-Output Signals for Logic gates:



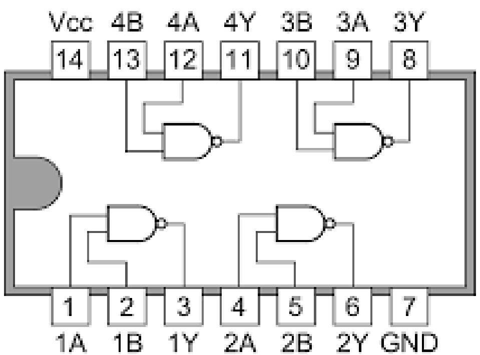
(a) Three-input AND gate



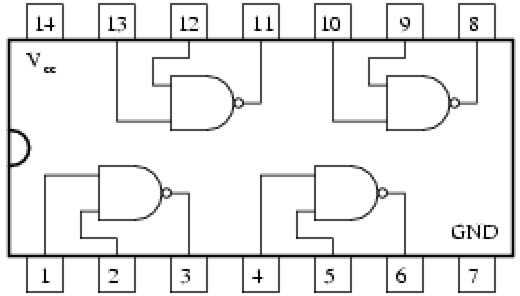
(b) Four-input OR gate

Logic Chips

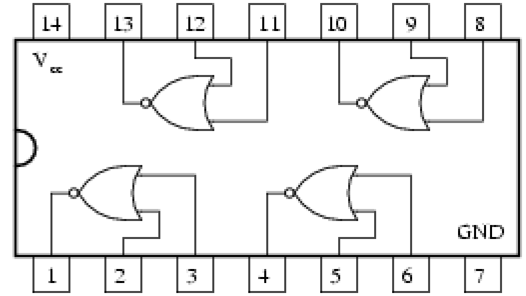
7400 Quad 2-input NAND Gates



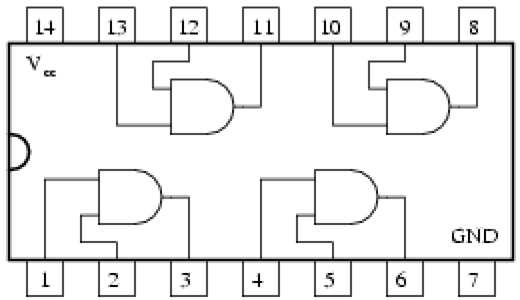
5400/7400
Quad NAND gate



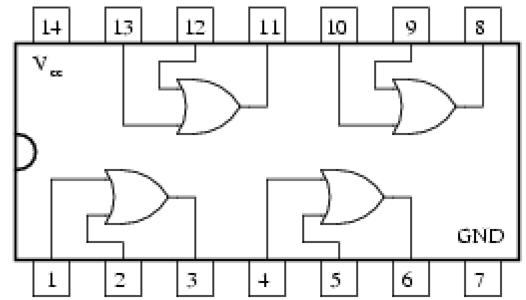
5402/7402
Quad NOR gate



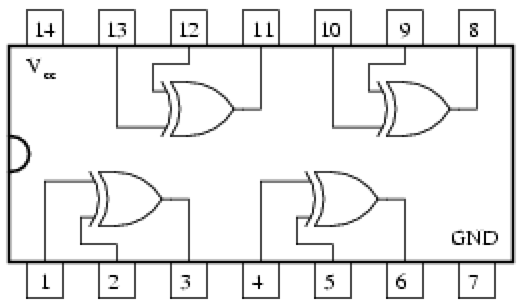
5408/7408
Quad AND gate



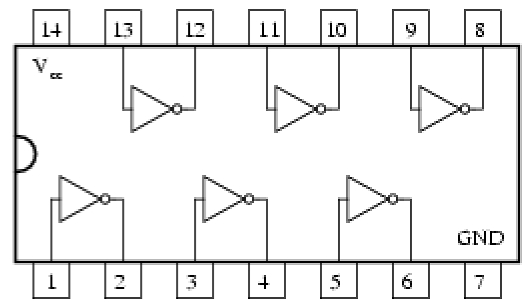
5432/7432
Quad OR gate



5486/7486
Quad XOR gate



5404/7404
Hex inverter



Integration levels

- SSI (small scale integration)
  - Introduced in late 1960s
  - 1-10 gates (previous examples)
- MSI (medium scale integration)
  - Introduced in late 1960s
  - 10-100 gates
- LSI (large scale integration)
  - Introduced in early 1970s
  - 100-10,000 gates
- VLSI (very large scale integration)
  - Introduced in late 1970s
  - More than 10,000 gates

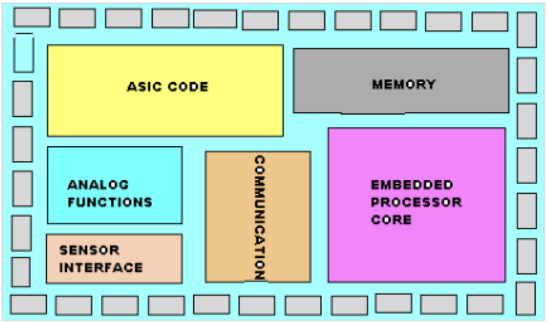
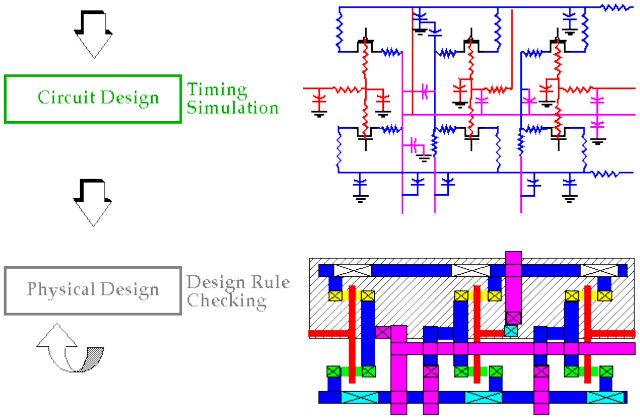
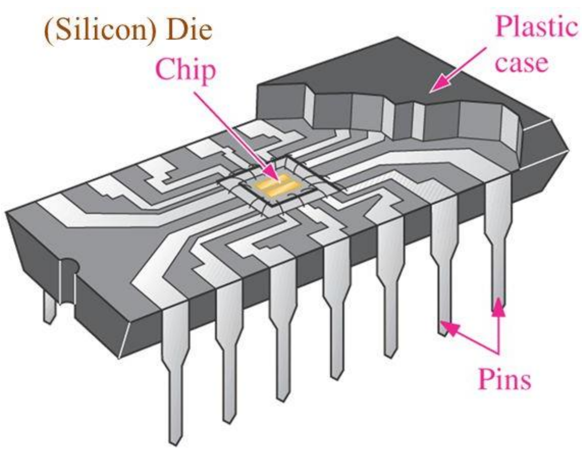


fig 4.1 **A SOC DEVICE**





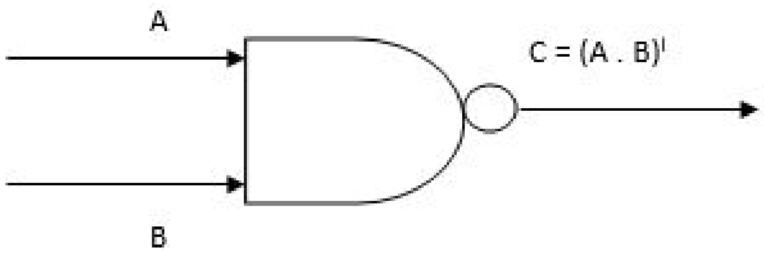
Logic Functions

- Logical functions can be expressed in several ways:
  - **Truth table**
  - **Logical expressions**
  - **Graphical form**

Truth table:

Input		Output
A	B	C=(A . B)'
0	0	1
0	1	1
1	0	1
1	1	0

Graphical Symbol:



Algebraic Expression is, C = (A . B)'

Example: Majority function

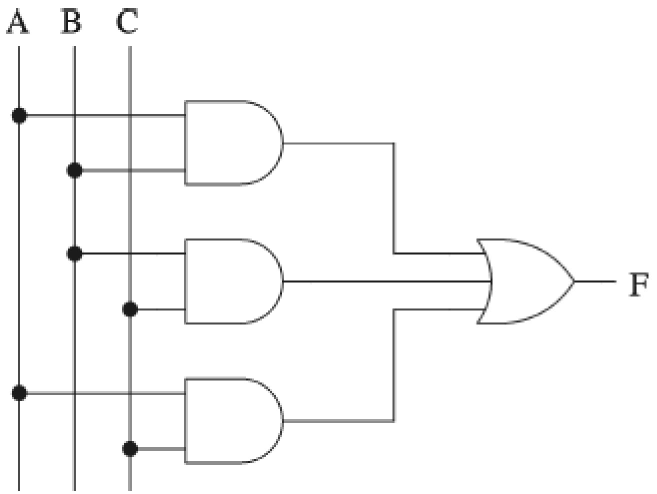
- Output is 1 whenever majority of inputs is 1
- We use 3-input majority function

3-input majority function

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

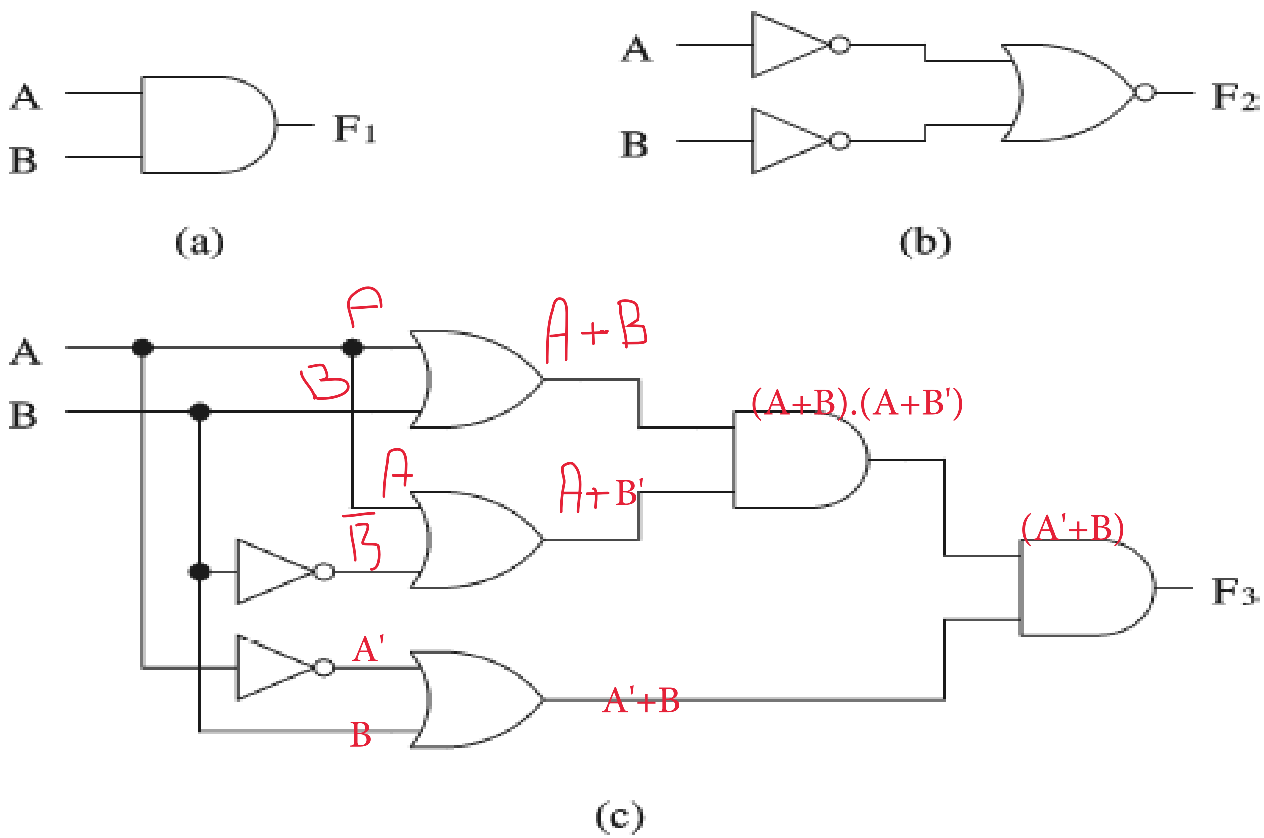
Logical expression form

$$F = A\,B + B\,C + A\,C$$



Logical Equivalence

- All three circuits implement ***F* = *A* . *B*** function

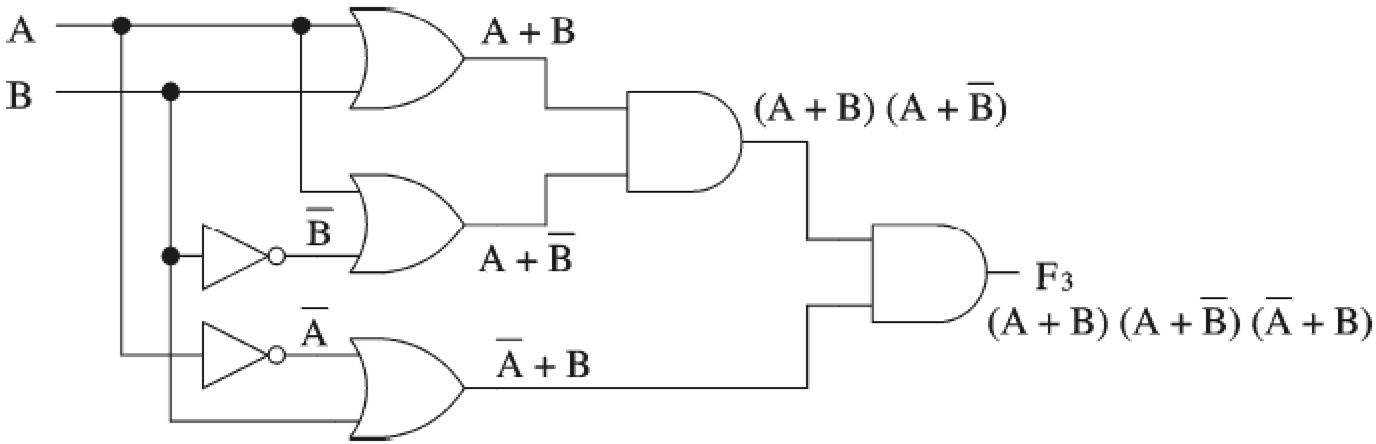


Logical Equivalence

- Proving **logical equivalence** of two circuits
- Derive the logical expression for the output of each circuit
- Show that these two expressions are equivalent, Two ways:
  - You can use the truth table method
    - For every combination of inputs, if both expressions yield the same output, they are equivalent
    - Good for logical expressions with small number of variables
  - You can also use algebraic manipulation
    - Need Boolean identities

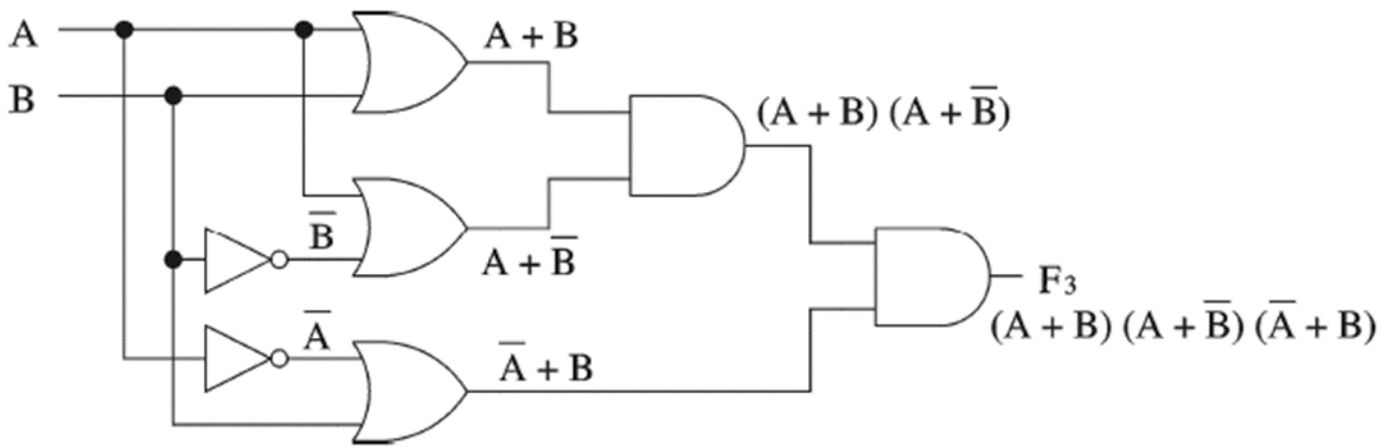
Derivation of logical expression from a circuit

- **Trace from the input to output**
- Write down intermediate logical expressions along the path



Truth Table Method

A	B	F1 = A B	F3 = $(A + B)(A + \overline{B})(\overline{A} + B)$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1



Boolean Algebra

Name	AND version	OR version
Identity	$x \cdot 1 = x$	$x + 0 = x$
Complement	$x \cdot \overline{x} = 0$	$x + \overline{x} = 1$
Commutative	$x \cdot y = y \cdot x$	$x + y = y + x$
Distribution	$x \cdot (y+z) = xy+xz$	$x + (y \cdot z) = (x+y)(x+z)$
Idempotent	$x \cdot x = x$	$x + x = x$
Null	$x \cdot 0 = 0$	$x + 1 = 1$

Boolean Algebra

Name	AND version	OR version
Involution	$\overline{\overline{x}} = x$	---
Absorption	$x \cdot (x+y) = x$	$x + (x \cdot y) = x$
Associative	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	$x + (y + z) = (x + y) + z$
de Morgan	$\overline{x \cdot y} = \overline{x} + \overline{y}$	$\overline{x + y} = \overline{x} \cdot \overline{y}$

Boolean Algebra

- Complement
  - x+x'=1 → 0+0'=0+1=1; 1+1'=1+0=1
  - x.x'=0 → 0. 0'=0. 1=0; 1. 1'=1. 0=0
- Duality
  - Form the dual of the expression
  - a + (bc) = (a + b)(a + c)
  - Following the replacement rules…
  - a(b + c) = ab + ac
- Absorption Property (Covering)
  - x + xy = x

x	y	xy	$x+xy$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

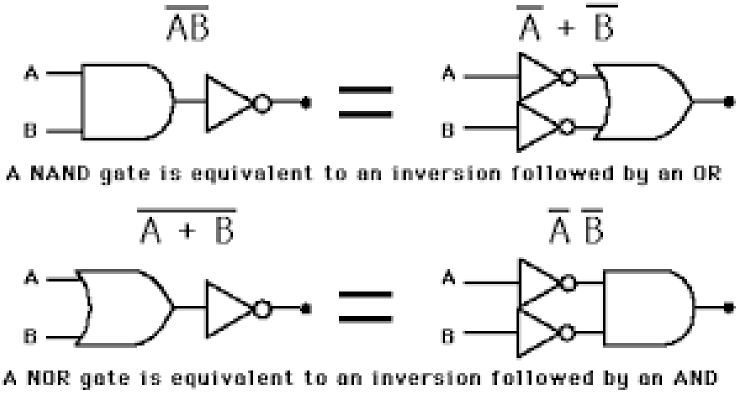
Basic Theorems

Postulates and Theorems of Boolean Algebra

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

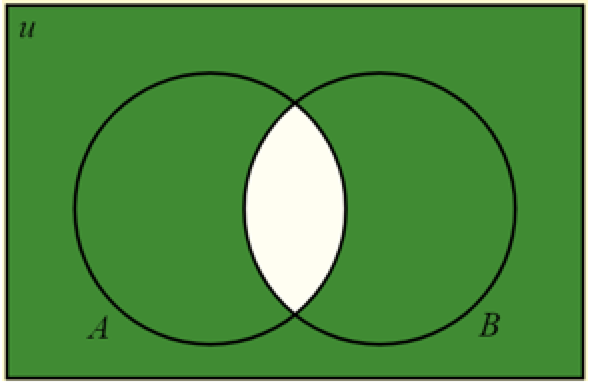
DeMorgan’s Theorem

- $(x + y)' = x'y'$
- $(xy)' = x' + y'$
- By means of truth table

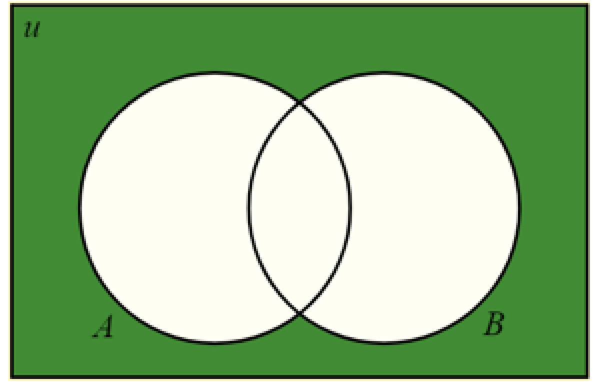


x	y	x'	y'	$x+y$	$(x+y)'$	$x'y'$	xy	$x'+y'$	$(xy)'$
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

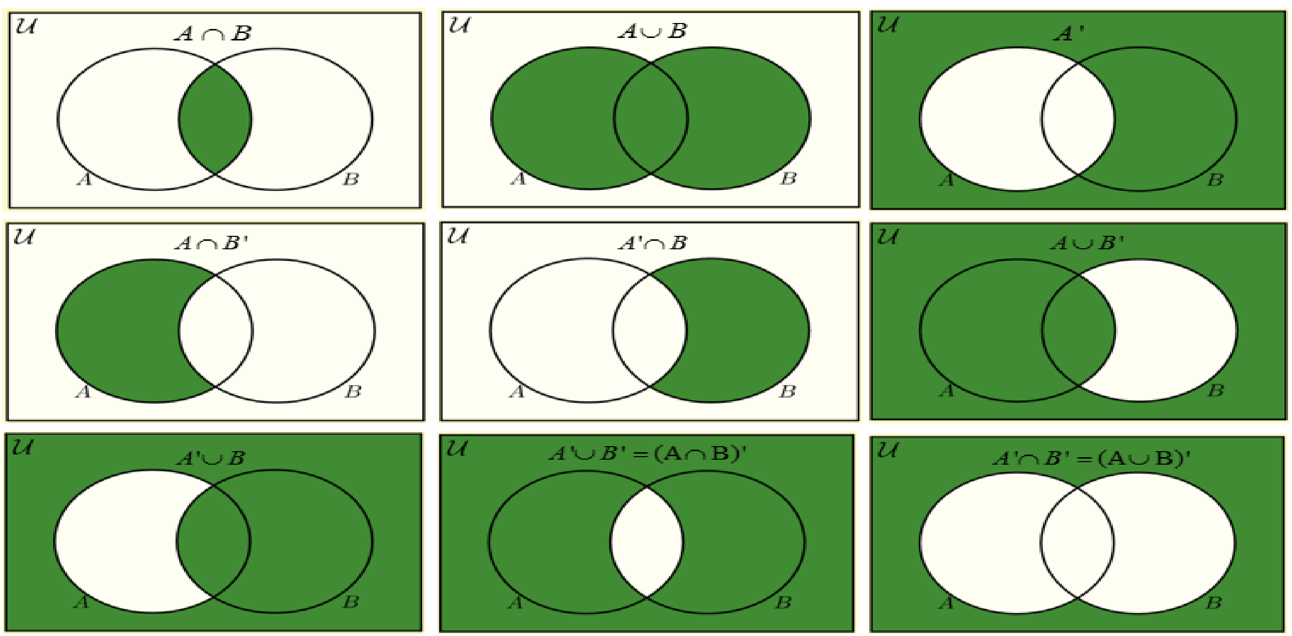
De Morgan’s Theorem



$$(A \cap B)' = A' \cup B'$$



$$(A \cup B)' = A' \cap B'$$



Consensus Theorem

1. $xy + x'z + yz = xy + x'z$
2. $(x+y) \cdot (x'+z) \cdot (y+z) = (x+y) \cdot (x'+z)$ -- (dual)

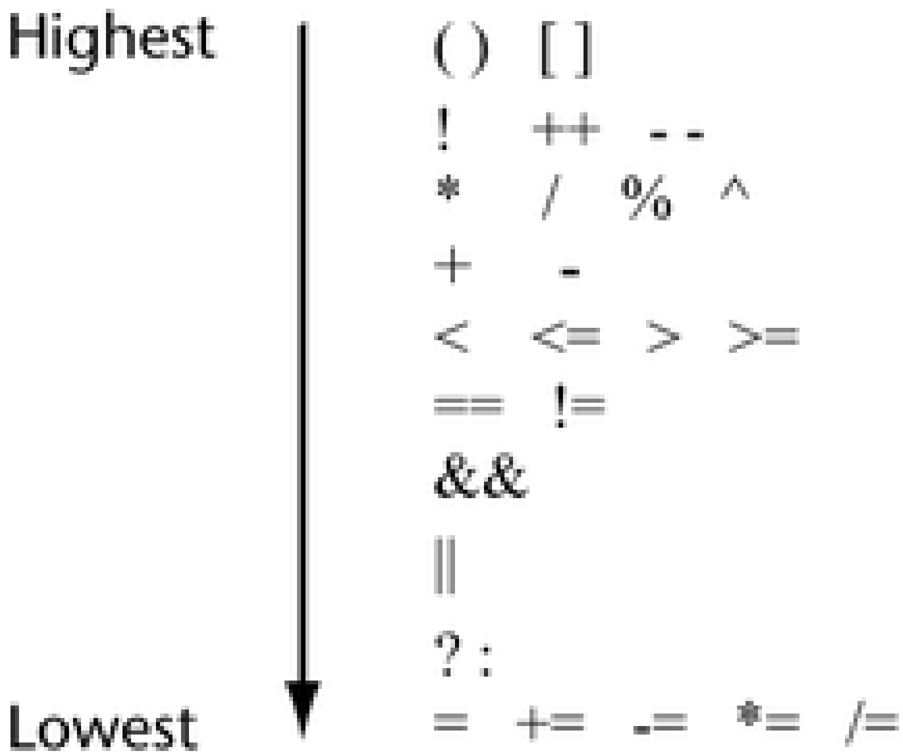
$$F = (A + B).(A' + C).(B + C)$$

Redundancy term

Complemented variable

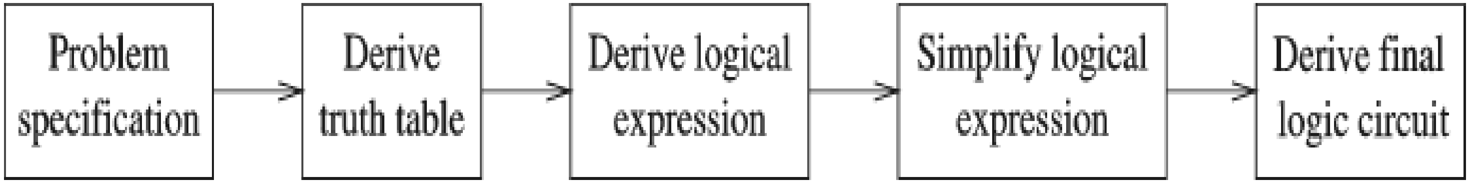
Operator Precedence

- The operator precedence for evaluating Boolean Expression is
    - **Parentheses**
    - **NOT**
    - **AND**
    - **OR**



Logic Circuit Design Process

- Problem specification.
- Truth table derivation.
- Derivation of logical expression.
- Simplification of logical expression.
- Implementation.

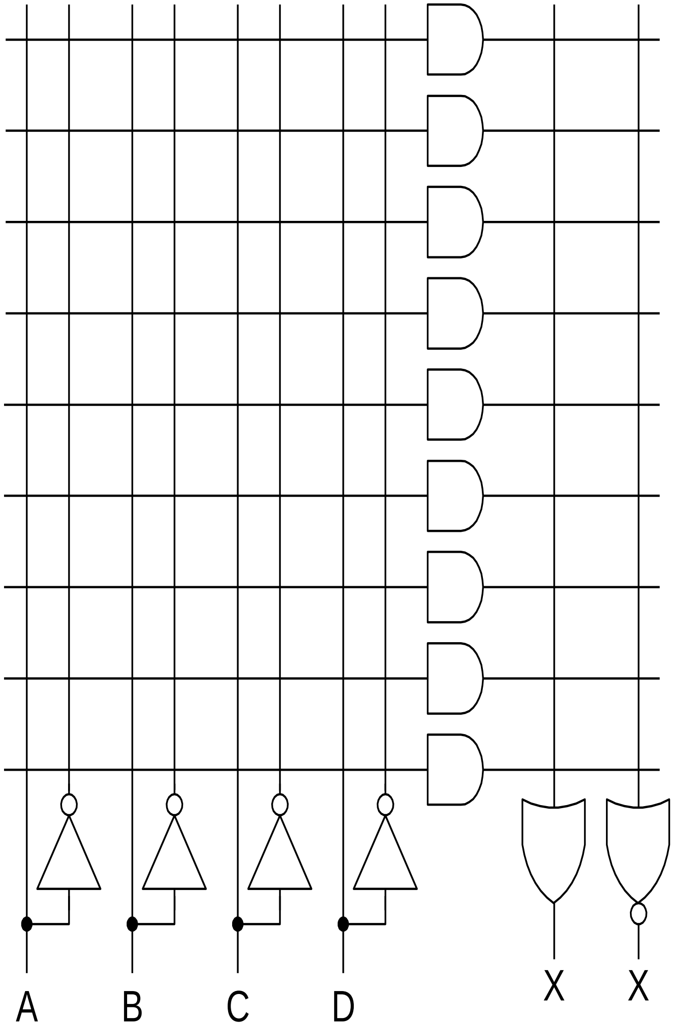


	Inputs				Output	
	A	B	C	D		
0	0	0	0	0		
1	0	0	0	1		
2	0	0	1	0		
3	0	0	1	1		
4	0	1	0	0		
5	0	1	0	1		
6	0	1	1	0		
7	0	1	1	1		
8	1	0	0	0		
9	1	0	0	1		
10	1	0	1	0		
11	1	0	1	1		
12	1	1	0	0		
13	1	1	0	1		
14	1	1	1	0		
15	1	1	1	1		

X =

Logic Diagram

X =

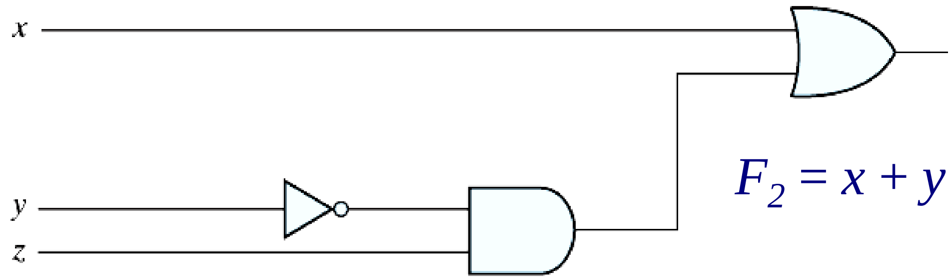


Examples

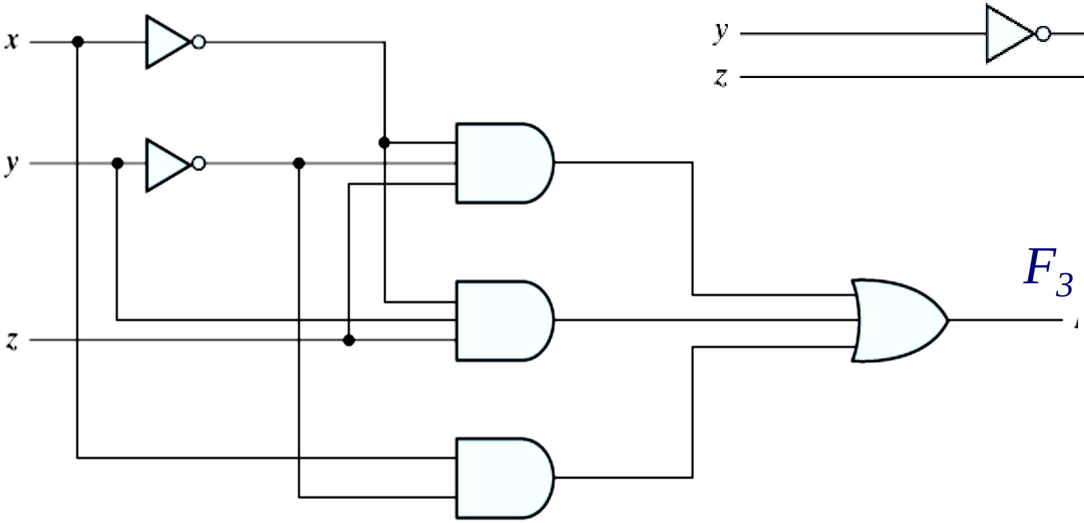
- $F_1 = x\,y\,z'$
- $F_2 = x + y'z$
- $F_3 = x'\,y'\,z + x'\,y\,z + x\,y'$
- $F_4 = x\,y' + x'\,z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

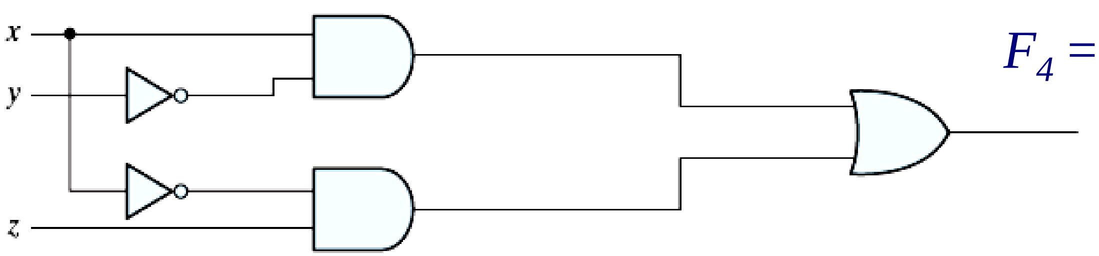
Boolean Functions



$F_2 = x + y'z$



$F_3 = x'y'z + x'yz + xy'$



$F_4 = xy' + x'z$

Example

1. $x(x'+y) = xx' + xy = 0+xy = xy$

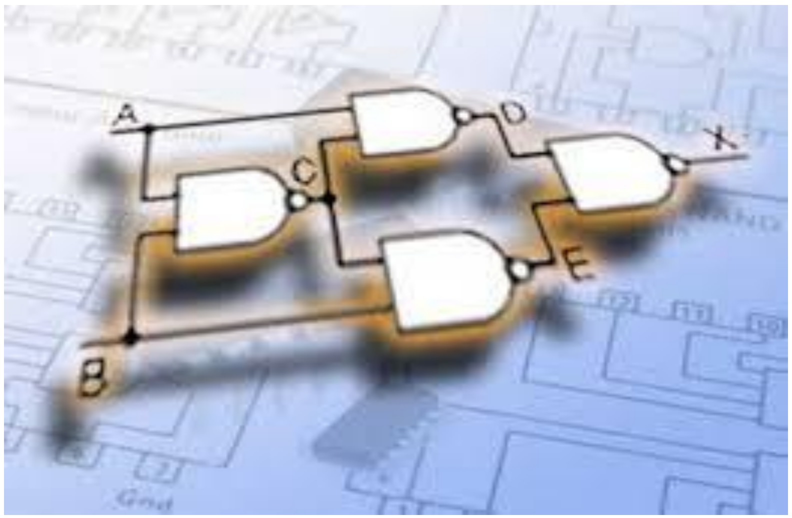
2. $x+x'y = (x+x')(x+y) = 1\,(x+y) = x+y$

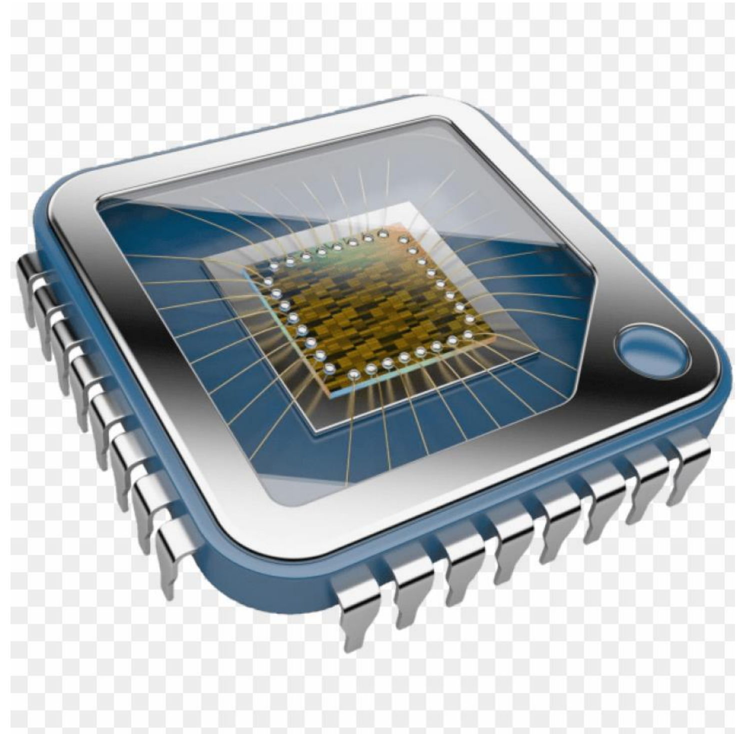
3. $(x+y)(x+y') = x+xy+xy'+yy' = x(1+y+y') = x$

4. $xy + x'z + yz = xy + x'z + yz(x+x') = xy + x'z + yzx + yzx' = xy(1+z) + x'z(1+y) = xy + x'z$

5. $(x+y)(x'+z)(y+z) = (x+y)(x'+z)$, by duality from function 4. (*consensus theorem* with duality)

Thank You





Dr. Hatem Yousry
E-mail: Hyousry@nctu.edu.eg