Chapter 3 Notes | Digital Logic Design | Logic Gates & Boolean Algebra

Logic Gates

- The most basic digital devices are called gates.
- A gate has one or more inputs and produces an output that is a function of the current input values.
- The relationship between the input and the output is based on a **certain logic**

All Logic Gates

- NOT
- AND
- OR
- XOR
- XNOR
- NAND
- NOR

Truth Table

- Provides a listing of every possible combination of inputs and its corresponding outputs.

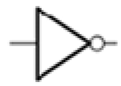
Basic Gates:

- AND, OR & NOT

Universal Gates:

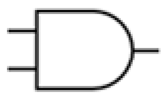
- A **universal gate** is a **gate** which can implement any other **gate**
- NAND & NOR

NOT



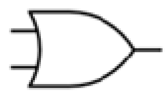
INPUT	OUTPUT
A	
0	1
1	0

AND



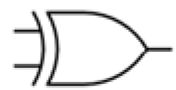
INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

OR



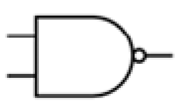
INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR



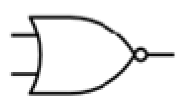
INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

NAND



INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR



INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

XNOR

INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	1

Boolean Algebra

- Like any other deductive mathematical system, defined with a set of elements, a set of operators and a number of axioms or postulates.
- In Boolean algebra, set consists at least 2 variables say x & y, with 2 binary operations {+} and {.} and 1 unary operation {‘}

Boolean Theorems & Postulates:

TABLE 2-1
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) $x(x + y) = x$

Duality

- **Duality Principle** – every valid Boolean expression (equality) remains valid if the operators and identity elements are interchanged, as follows:

  + ↔ **.**

  1 ↔ 0

- Example: Given the expression

  a + (b.c) = (a+b)**.**(a+c)

  then its dual expression is

  a **.** (b+c) = (a.b) + (a.c)

Complementing a function

1. Take dual of the function
2. Complement each literals

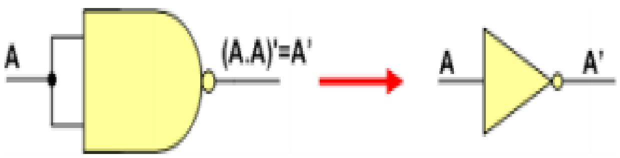
Example: F1= x’yz’+x’y’z

1. Dual of the function F1 is (x’+y+z’)(x’+y’+z)
2. Complement each literal= (x+y’+z)(x+y+z’)

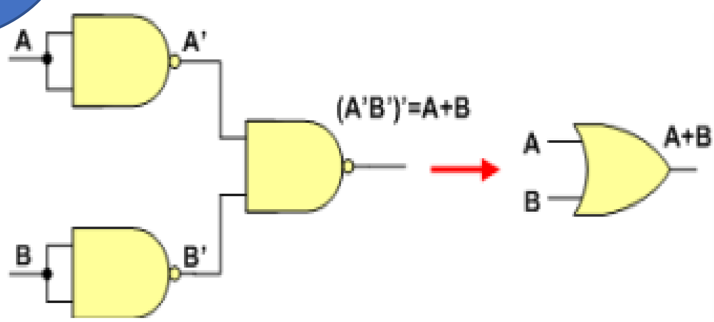
   Therefore, F1’= (x+y’+z)(x+y+z’)

Using NAND to represent NOT, OR & AND Gate:

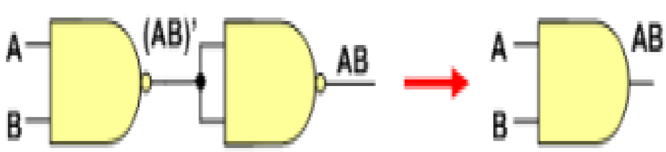
NOT Gate



OR Gate

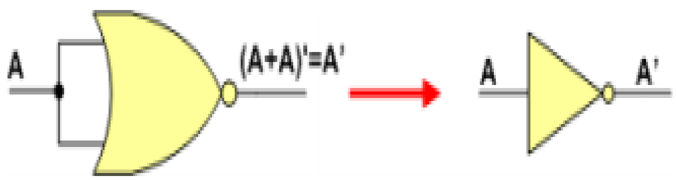


AND Gate

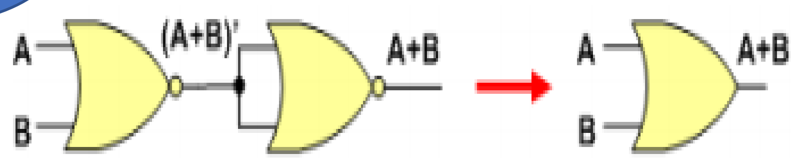


Using NOR to represent NOT, OR & AND Gate:

NOT Gate



OR Gate



AND Gate