**Combined gates**

Sometimes, it is practical to combine functions of the basic gates into more complex gates, for instance in order to save space in circuit diagrams. In this section, we show some such combined gates together with their truth tables.

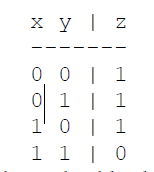
- **The *nand*-gate**

The *nand*-gate is an *and*-gate with an inverter on the output. So instead of drawing several gates like this:





The truth table for the *nand*-gate is like the one for the *and*-gate, except that all output values have been inverted:

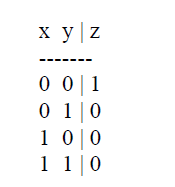


**The *nor*-gate**

The *nor*-gate is an *or*-gate with an inverter on the output. So instead of drawing several gates like this:



The truth table for the *nor*-gate is like the one for the *or*-gate, except that all output values have been inverted:

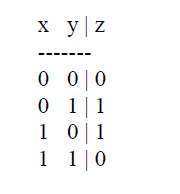


- **The *exclusive-or*-gate**

The *exclusive-or*-gate is similar to an *or*-gate. It can have an arbitrary number of inputs, and its output value is 1 if and only if *exactly one input* is 1 (and thus the others 0). Otherwise, the output is 0. We draw an *exclusive-or*-gate like this:



The truth table for an *exclusive-or*-gate with two inputs looks like this:



**Boolean Algebra**

One of the primary requirements when dealing with digital circuits is to find ways to make them as simple as possible. This constantly requires that complex logical expressions be reduced to simpler expressions that nevertheless produce the same results under all possible conditions. The simpler expression can then be implemented with a smaller, simpler circuit, which in turn saves the price of the unnecessary gates, reduces the number of gates needed, and reduces the power and the amount of space required by those gates. One tool to reduce logical expressions is the mathematics of logical expressions, introduced by George Boole in 1854 and known today as *Boolean Algebra*. The rules of Boolean Algebra are simple and straight-forward, and can be applied to any logical expression. The resulting reduced expression can then be readily tested with a Truth Table, to verify that the reduction was valid.

Boolean algebra is an algebraic structure defined on a set of elements B, together with two binary operators(+, **.**) provided the following postulates are satisfied.

1. Closure with respect to operator + and Closure with respect to operator **.**

2. An identity element with respect to + designated by 0: X+0= 0+X=X

An identity element with respect to . designated by 1: X.1= 1.X=X

3. Commutative with respect to +: X=Y=Y+X

Commutative with respect to **.**: X.Y=Y.X

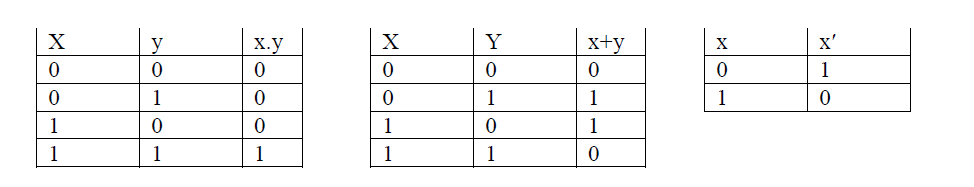
4. . distributive over +: X.(Y+Z)=X.Y+X.Z

+ distributive over .: X+(Y.Z)=(X+Y).(X+z)

5. For every element x belonging to B, there exist an element x′ or called the complement of x such that x. x′=0 and x+ x′=1 *x*

6. There exists at least two elements x,y belonging to B such that x≠y

The two valued Boolean algebra is defined on a set B={0,1} with two binary operators + and**.**



**Laws of Boolean Algebra**

**Postulate 2 :** (a) *0 + A = A* (b) *1.A = A* **Postulate 5 :** (a) *A + A*′ *=1* (b) *A. A*′=0

**Theorem1 : Identity Law** (a) *A + A = A* (b) *A A = A*

**Theorem2** (a) *1 + A = 1* (b) *0. A = 0* **Theorem3: involution** *A*′′=A

**Postulate 3 : Commutative Law** (a) *A + B = B + A* (b) *A B = B A*

**Theorem4: Associate Law** (a) *(A + B) + C = A + (B + C)* (b) *(A B) C = A (B C)*

**Postulate4: Distributive Law** (a) *A (B + C) = A B + A C* (b) *A + (B C) = (A + B) (A + C)*

**Theorem5 : De Morgan's Theorem** (a) (A+B)′= A′B′ (b) (AB)′= A′+ B′

**Theorem6 : Absorption** (a) *A + A B = A* (b) *A (A + B) = A* Prove Theorem 1 : (a) X+X=X x+x=(X+X).1 by postulate 2b =(x+x)(x+x′) 5a =x+xx′ 4b =x+0 5b =x 2a Prove Theorem 1 : (b) X.X=X xx=(X.X)+0 by postulate 2a =x.x+x.x′ 5b =x(x+x′) 4a =x.1 5a =x 2b