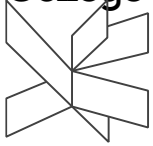


Gør tanke til handling

**VIA University
College**



Boolean algebra

Operations on bits

Boolean algebra

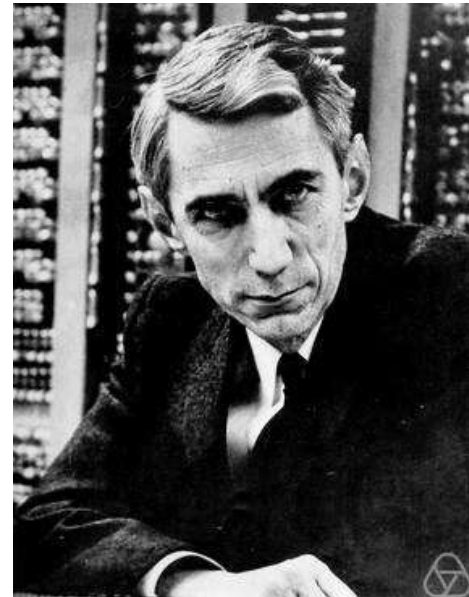
The study of the kinds of operations that can be carried out on bits is called **Boolean algebra**.

George Boole (1815-1864)



Developed the rules of logic.

Claude Shannon (1916-2001)



Showed how to use this for designing computer circuits.

Basic Boolean operators

So what can we do with the 0's and 1's? The most basic operations is to add them, multiply them and take the complement ("the opposite").

These operations are referred to as "OR", "AND" and "NOT":

OR

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 1 = 1$$

AND

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 1 = 1$$

NOT

$$\overline{0} = 1$$

$$\overline{1} = 0$$

Example:

Use the rules above to calculate $1 \cdot 0 + \overline{(0 + 1)}$:

$$1 \cdot 0 + \overline{(0 + 1)} = 1 \cdot 0 + \overline{1} = 0 + 0 = 0$$

Additional Boolean operators

In addition to the basic operators from the last slide, we will also occasionally meet the “Not AND (NAND)”, “Not OR (NOR)” and “Exclusive OR (XOR)” operators:

NAND

$$0 \uparrow 0 = 1$$

$$0 \uparrow 1 = 1$$

$$1 \uparrow 1 = 0$$

NOR

$$0 \downarrow 0 = 1$$

$$0 \downarrow 1 = 0$$

$$1 \downarrow 1 = 0$$

XOR

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 1 = 0$$

Boolean variables

A Boolean variable is a variable which can only take the values 0 or 1. We will typically use x , y and z as our Boolean variables.

Order of Boolean operators

The basic Boolean operators are applied in the order:

1. NOT (\bar{x})
2. AND ($x \cdot y$)
3. OR ($x + y$)

So, for example, $x \cdot y + z$ means $(x \cdot y) + z$.

Most of the time, however, we will write the parenthesis to avoid misunderstandings.

Boolean functions

A **Boolean function** is a function that takes 0's and 1's as inputs and whose output is also 0 or 1. It is composed of the Boolean operators introduced in the previous slides, and expressed using Boolean variables.

Example

$F(x) = \bar{x}$ is an example of a Boolean function.

We can display the function in a table, which shows the value of $F(x)$ for all values of x :


x	$F(x)$
0	1
1	0

Example: A Boolean function of two variables

$$F(x, y) = x\bar{y}$$

Below, the function is displayed in a table, which shows the value of F for all combinations of values of the inputs x and y :

x	y	\bar{y}	$F(x, y)$
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0


 “helping”-column – this is not strictly necessary, but it assists the calculation of F

Example: A Boolean function of 3 variables

$$F(x, y, z) = xy + \bar{z}$$

The function is displayed in the table below, which shows the value of F for all combinations of the input values x , y , and z :

x	y	z	xy	\bar{z}	$F(x, y, z)$
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1


“helping”-columns – these are not strictly necessary, but they assist the calculation of F

Simplifying a Boolean expression

The table below shows the output for the functions $F(x, y) = (x\bar{y} + xy) \cdot y$ and $G(x, y) = xy$ for all values of x and y .

x	y	xy	\bar{y}	$x\bar{y}$	$xy + x\bar{y}$	$F(x, y)$	$G(x, y)$
0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0	1	1	1	0	0
1	1	1	0	0	1	1	1

These functions produce the same output for all values of x and y .

That is, these two functions are the same!

It is often possible to reduce Boolean expressions significantly. In the next slide, you will see some rules for how to do that.

Boolean identities

- Double complement:

$$\bar{\bar{x}} = x$$

- Idempotent laws:

$$x + x = x, \quad x \cdot x = x$$

- Identity laws:

$$x + 0 = x, \quad x \cdot 1 = x$$

- Domination laws:

$$x + 1 = 1, \quad x \cdot 0 = 0$$

- Unit property:

$$\bar{x} + x = 1$$

- Zero property:

$$\bar{x} \cdot x = 0$$

- Commutative laws:

$$x + y = y + x, \quad x \cdot y = y \cdot x$$

- Absorption laws:

$$x + x \cdot y = x$$

$$x \cdot (x + y) = x$$

- De Morgan's laws:

$$\overline{(x \cdot y)} = \bar{x} + \bar{y}$$

$$\overline{(x + y)} = \bar{x} \cdot \bar{y}$$

- Associative laws:

$$x + (y + z) = (x + y) + z$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

- Distributive laws:

$$x + y \cdot z = (x + y) \cdot (x + z)$$

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

Example

Show that $x + x \cdot y = x$.

1) Using Boolean identities:

$$x + x \cdot y = x \cdot 1 + x \cdot y = x(1 + y) = x$$

2) Using a table:

x	y	$x \cdot y$	$x + x \cdot y$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Since the columns for x and $x + x \cdot y$ are identical, $x = x + x \cdot y$ for all values of x and y .

Creating circuits for Boolean functions

Logic gates

Each of the Boolean operations we have studied so far are represented in circuits using the symbols below.



NOT gate



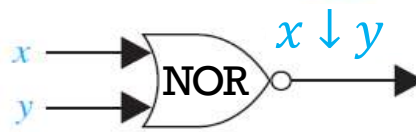
OR gate



AND gate



NAND gate



NOR gate

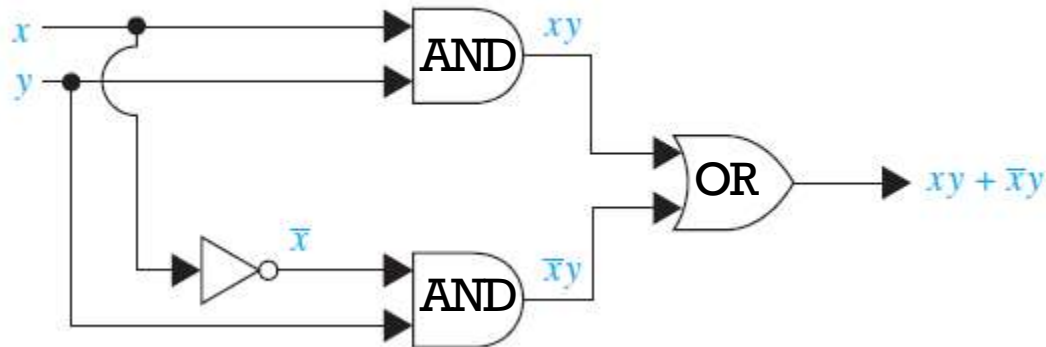


XOR gate

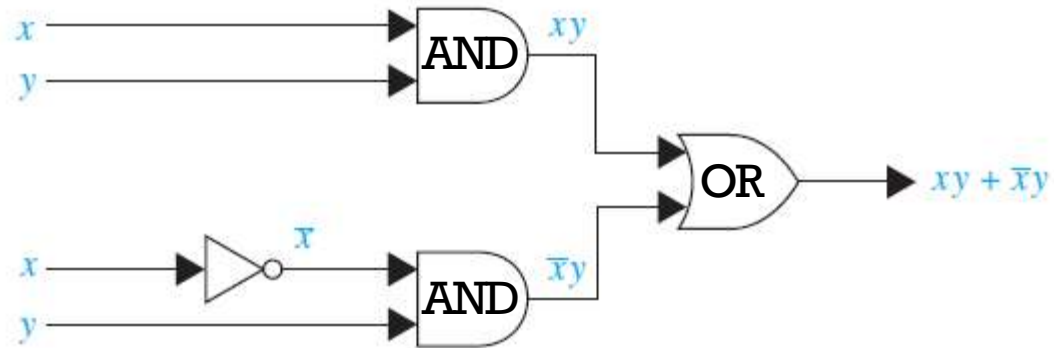
Example: Logic circuits

Using the logic gates from the previous slide, we can build circuits which represent any given Boolean function.

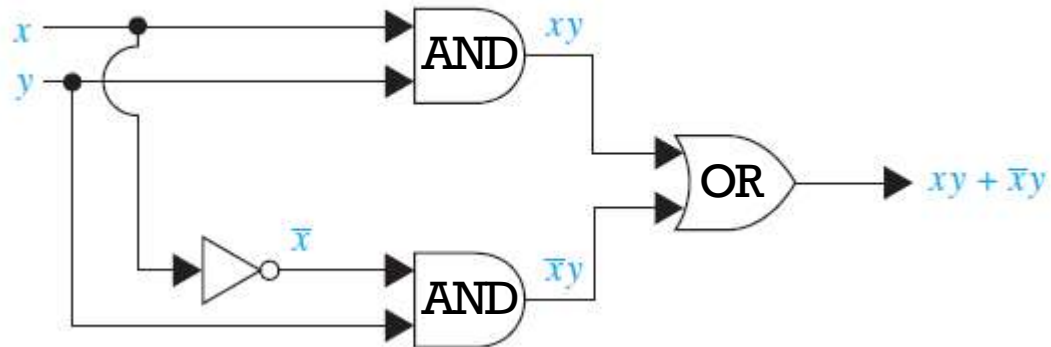
For example, the Boolean function $F(x, y) = xy + \bar{x}y$ can be represented with the circuit below.



Logic circuits

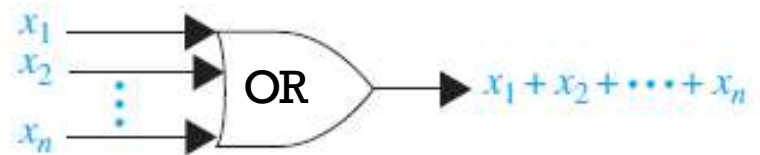
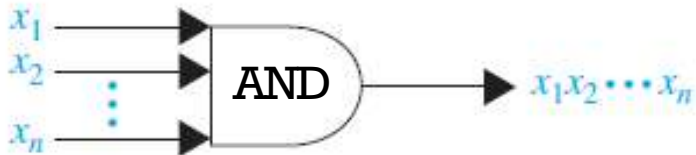


Two ways of drawing the same circuit!



Logic circuits

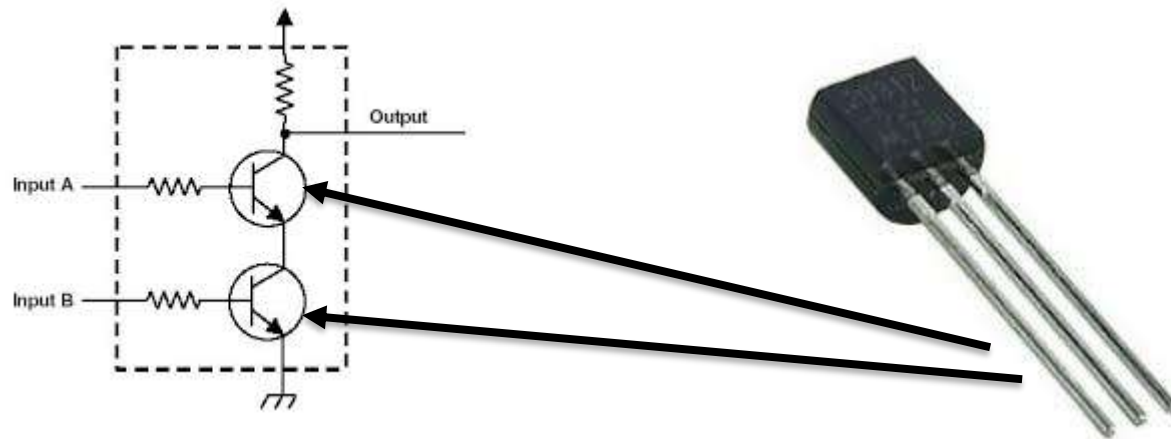
We can also have more than two inputs to a gate:



Logic gates in computers

Logic gates in computers

A NAND-gate can be constructed from two transistors in the following way:



- If the input to a transistor is 1, this corresponds to a closed switch.
- If both input *A* and input *B* are 1, there is a connection all the way from the bottom to the top, and the current flow that way – that means the output is 0.
- If both input *A* and input *B* are 0, there is no connection at all, so the output is 0.
- If either input *A* or input *B*, both not both, are 1, current can only flow from that input to the output, so the output is 1.

Functional completeness

The table below shows that all the basic Boolean operations can actually be represented using only NAND-gates (and so can NOR and XOR) – therefore, we say that the NAND-gate is **functionally complete**.

Since we have just seen how to build a NAND-gate using transistors, you now know how all the functions we have considered could be represented with transistors in a computer!

x	y	$x \uparrow x$	$(x \uparrow y) \uparrow (x \uparrow y)$	$(x \uparrow x) \uparrow (y \uparrow y)$
0	0	1	0	0
0	1	1	0	1
1	0	0	1	1
1	1	0	1	0

↑

NOT: \bar{x}

↑

AND: $x \cdot y$

↑

OR: $x + y$