

Cheatsheet - Logic Gates

Fabio Lama – fabio.lama@pm.me

1. Intro

(NOTE: Reading the *Postulates of Boolean Algebra* cheatsheet is recommended here)

Logic gates are basic elements of circuits implementing Boolean operations. The most basic circuits are **OR** gates, **AND** gates and invertors (**NOT** gates). All boolean functions can be written in terms of these three logic operations.









- **AND** operation is represented as $f = x \cdot y$ or $f = xy$.
- **OR** operation is represented as $f = x + y$.
- **NOT** operation is represented as $f = \bar{x}$.

Other gates:

- **XOR** operation is *true* only when the value of the inputs differ.
- **NAND** operations is equivalent to "not AND".
- **NOR** operation is equivalent to "not OR".
- **XNOR** operation is equivalent to a "not XOR".

AND, OR, XOR and XNOR are **commutative** (e.g. $a + b = b + a$) and **associative** (e.g. $a + (b + c) = (a + b) + c$). NAND and NOR are commutative but not associative.

Logic Gates - Symbols and Truth Tables

<div>BUF (Buffer)</div> <div></div>	<div>In</div> <div>Out</div> <div>00</div> <div>11</div>
<div>AND</div> <div></div>	<div>In1In2Out</div> <div>000</div> <div>010</div> <div>100</div> <div>111</div>
<div>OR</div> <div></div>	<div>In1In2Out</div> <div>000</div> <div>011</div> <div>101</div> <div>111</div>
<div>XOR (Exclusive Or)</div> <div></div>	<div>In1In2Out</div> <div>000</div> <div>011</div> <div>101</div> <div>110</div>
<div>NOT (Inverter)</div> <div></div>	<div>In</div> <div>Out</div> <div>01</div> <div>10</div>
<div>NAND (NOT AND)</div> <div></div>	<div>In1In2Out</div> <div>001</div> <div>011</div> <div>101</div> <div>110</div>
<div>NOR (NOT OR)</div> <div></div>	<div>In1In2Out</div> <div>001</div> <div>010</div> <div>100</div> <div>110</div>
<div>XNOR (NOT XOR)</div> <div></div>	<div>In1In2Out</div> <div>001</div> <div>010</div> <div>100</div> <div>111</div>

A circle behind a symbol indicates that the output signal is inverted.

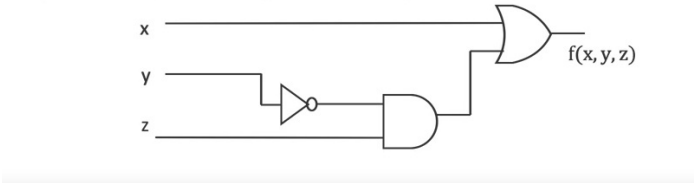
Figure 1. Source: http://www.exclusivearchitecture.com/?page_id=2425

2. Circuits

We describe the combination of logic gates as a **circuit**.

- Let's consider the Boolean function f defined as:
$$f(x,y,z) = x + y'z$$

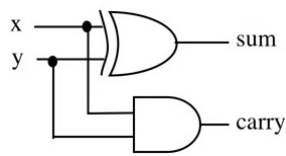
- f can be represented by the following circuit:



A circuit that's used for the **addition** of inputs is called an **adder**. A **half adder** takes two inputs and generates a **carry** and a **sum**. A **full adder** takes three inputs and generates a carry and a sum.

For example, an **half adder**:

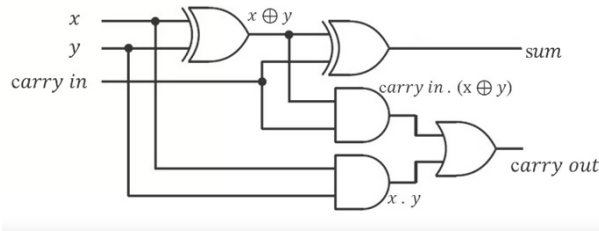
$$\begin{aligned} \text{sum} &= xy' + x'y = x \oplus y \\ \text{carry} &= xy \end{aligned}$$



And a **full adder**:

$$\text{sum} = x \oplus y \oplus \text{carry in}$$

$$\text{carry out} = xy + \text{carry in} \cdot (x \oplus y)$$



2.1. Simplification of Circuits (example)

Let's consider the following boolean expression:

$$E = ((xy)'z)'((x' + y)(y' + z'))'$$

Using the **De Morgan's laws** and **involution**:

$$\begin{aligned} E &= ((xy)'' + z')((x' + z)' + (y' + z')') \\ &= (xy + z')((x'' \cdot z') + y'' \cdot z'') \\ &= (xy + z')(xz' + yz) \end{aligned}$$

Using the **distributive laws**:

$$E = xyz' + xyyz + z'xz' + z'yz$$

Using **commutative**, **idempotent** and **complement** laws:

$$E = xyz' + xyz + xz' + 0$$

Using **absorption** law:

$$E = xyz + xz'$$

Last updated 2022-12-10 16:45:22 UTC