MATH 8
HANDOUT 8: LOGIC 2: LOGIC GATES

Reminder: Basic logic operations and laws

NOT (for example, NOT A): true if A is false, and false if A is true. Commonly denoted by $\neg A$

AND (for example A AND B): true if both A, B are true, and false otherwise (i.e., if at least one of them is false). Commonly denoted by $A \wedge B$

OR (for example A OR B): true if at least one of A, B is true, and false otherwise. Sometimes also called "inclusive or" to distinguish it from the "exclusive or" below. Commonly denoted by $A \vee B$.

XOR (for example, A XOR B): exclusive or; true if exactly one of A, B is true and false otherwise

NAND: not and:A NAND $B = \neg(A$ AND $B)$. True if at least one of A, B is false; false if both A, B are true.

NOR: not or:A NOR $B = \neg(A$ OR $B)$. True if both of A, B are false; false if at least one of A, B is true.

Some logic laws:

Double negation:

$$\neg(\neg A) \iff A$$

De Morgan's law:

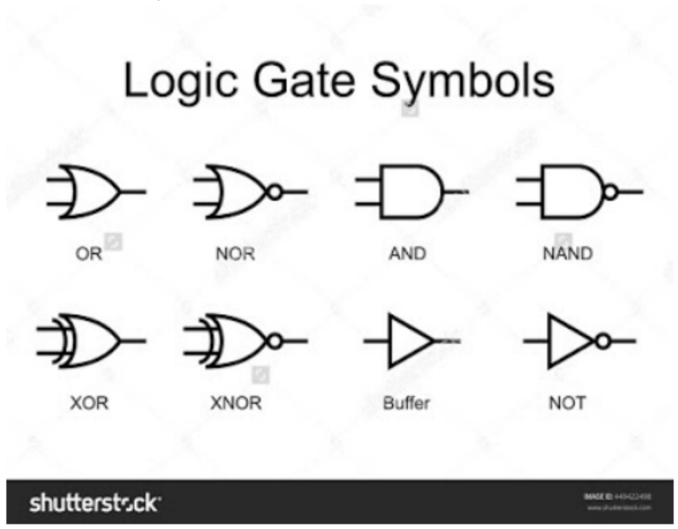
$$\neg(A \wedge B) \iff (\neg A) \vee (\neg B)$$
$$\neg(A \vee B) \iff (\neg A) \wedge (\neg B)$$

Logic gates in electronics

In electronics, it is common to have digital signals which take two values: LOW and HIGH (e.g., in certain microcontrolers, voltage can range between 0–5V, and anything over 2.5V is considered HIGH, and anything below, LOW). These two values can also be considered as two binary digits: LOW= 0, HIGH= 1, or as boolena values: LOW= 0 =false, HIGH= 1 =true.

The basis of all modern computers are "logic gates", chips that take two (or more) such inputs and produce an output described by some truth table. (These chips contain transistors and diodes, but this is irrelevant for us). For example, below is a typical such chip, containing four AND gates, each with two inputs

In electronics, there are standard notations for different kinds of gates (AND, OR, NAND...):



Combining such simple gates, one can create more complicated ones — and use that to create circuits which take as input a collection of binary digits and produce as output some function such as sum or product of inputs (interpeting n binary inputs as an n-digit binary number).

There is a number of online simulators that allow you to create and test such circuits virtually; in particular, you could use `http://logic.ly/demo`.

Problems

1. Half adder. Adding two one-digit binary numbers is an operation which has two inputs, A and B, and produces two outputs, X_0, X_1, which are just two digits of the sum $A + B$ written in binary. E.g., $1_b + 1_b = 10_b$ (we use subscript b to indicate that this is a binary number, not a decimal one); thus, when $A = 1, B = 1$, we have $X_0 = 0$ (last digit of sum) and $X_1 = 1$.

(a) Write a table of values for this operation, listing for each possible combination of A, B the values of outputs X_0, X_1.

(b) Can you get X_0 from A, B by using one of the standard logic gates listed above?

(c) Same question for X_1

(d) Construct a circiut with two inputs and two outputs, consisting of the basic logic gates, which implements this addition operation. [This is commonly known as half-adder.] Test your circuit in `logic.ly`.

***2. Full adder.** Can you construct a circuit which does binary addition of three inputs A, B, C (each input being a 1-digit binary number)? This circuit is known as full adder.

Hint: use the half-adder you constucted in the previous problem as one of the building blocks — you can use it to first add A and B.

Test your final circuit in `logic.ly`

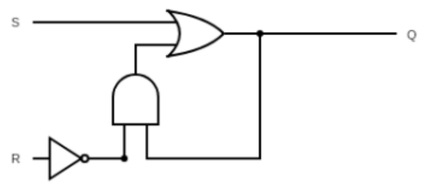
3. Using the full adder constucted in the previous problem as a buiding block, construct a circuit which adds two 3-digit binary numbers. Your circuit must have six inputs A_0, A_1, A_2, B_0, B_1, B_2 (representing digits of A, B respectively) and four outputs $X_0, \dots, X_3$, representing digits of the sum.

This is known as `ripple-carry adder` (carry because it implements usual addition with carry digits; word "ripple" can be ignored for now).

4. Consider the circuit shown below.

Can you make a table of values, describing for each combination of values of the inputs S, R the value of the output? [Hint: there is a catch there...]

You could test it on `logic.ly`



5. Show that for any value of A, expression $A \vee (\neg A)$ is true (such expressions, which are true for all values of variables involved, are called *tautologies*). This particular tautology is sometimes called "law of excluded middle" (meaning there is no middle ground — A must be true or false).

Similarly, show that $A \wedge (\neg A)$ is always false.