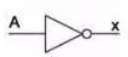
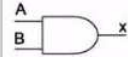
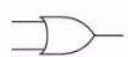
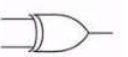


Logic Circuits

The basic instructions that our computer machines are able to perform can be represented by *logic circuits*, which are basically graphs whose nodes are logic “gates”, and edges represent the way these gates are connected. We’ll suppose in the following that the flow of information (electric power) travels from the left-handed side of the circuits, to its right-handed side. We’ll consider the following set of logic gates:

NOT		AND			OR			XOR		
										
A	X	B	A	X	B	A	X	B	A	X
0	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1	0	1	1
		1	0	0	1	0	1	1	0	1
		1	1	1	1	1	1	1	1	0

If you have your lecture notes with you, it is recommended that you give a look before starting your exercises.

Exercise 1 – warming up

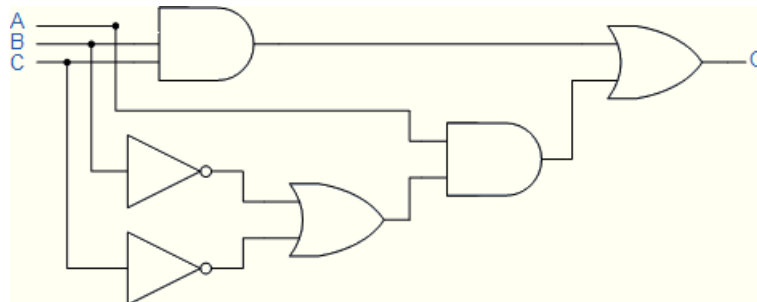
Our very first exercise will consist in designing a 1-bit multiplier:

- how many logic gates do we need?
- should we take into consideration any carry-over?

Even if it looks too simple to you, please draw the circuit on paper.

Exercise 2

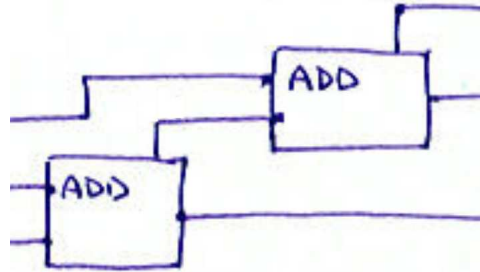
Develop the truth table for the circuit below:



How many rows and columns your truth table has? Can you find a general formula capable to give the number of rows and columns on the basis of some circuit selected features? Pay particular attention to the last column of the table: the result is very close to the input column A. Can you design a smaller circuit, still involving A, B and C and that gives exactly the same result, but by using no more than 2 logic gates?

Exercise 3

Do you remember how we were used to make products at elementary school? Design a logic circuit that implements exactly the same procedure, but only on 2-bit unsigned integers. Your circuit needs to use *adders*: you can represent them in a compact form, such as a box accepting bits in input and in output. It is recommended to use half-adders. To make your life easier, the detail of the solution where a half-hadder is used twice is given below.



Exercise 4 – Exam December 2019

The half-adder and the full-adder are two very well-known and simple circuits to perform the sum of a subset of bits. The full-adder is able to take three bits in entry (one of them represents the carry-in from a previous sum), and to provide in output two bits, consisting of the result of the sum, with its carry-out.

We are now interested in extending a little further the capacity of the full-adder circuit. Instead of taking three bits in input, we want our new circuit to take four entries, and to compute the sum of these four bits. Notice that the half-adder will part of our extended circuit: you can represent it as a box, just as we did in the previous exercise. For simplicity, we will not distinguish between result and carry-overs, but the order of the bits forming your result will need to be clear in the drawing representing your solution.