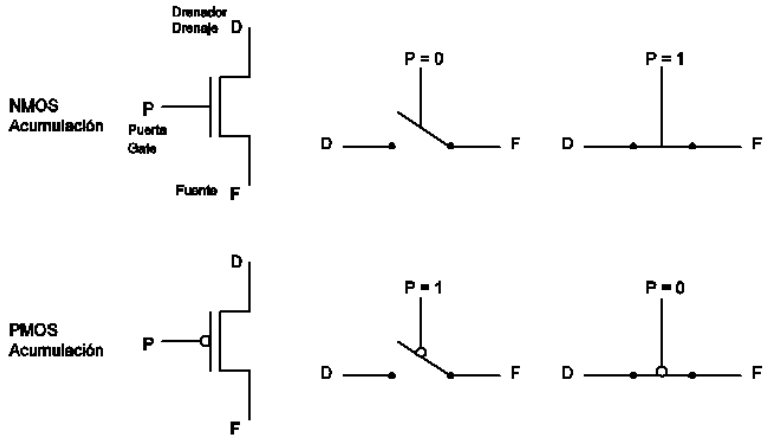
Basic CMOS concepts

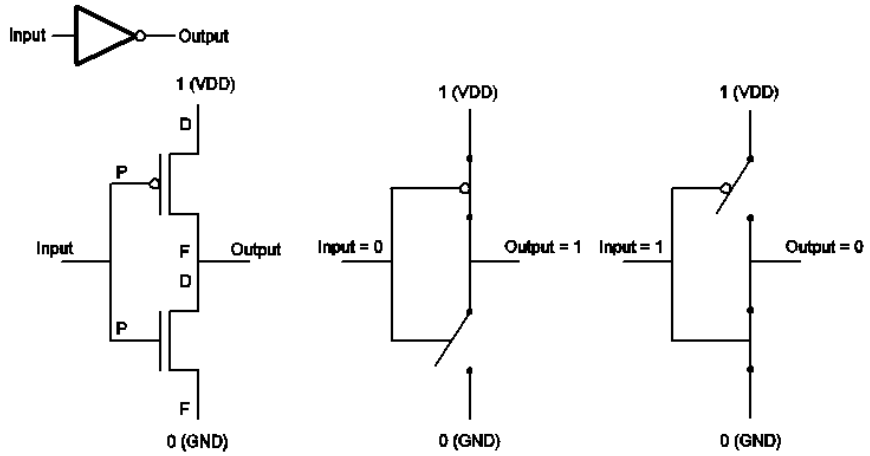
We will now see the use of transistor for designing logic gates. Further down in the course we will use the same transistors to design other blocks (such as flip-flops or memories)

Ideally, a transistor behaves like a switch. For NMOS transistors, if the input is a 1 the switch is on, otherwise it is off. On the other hand, for the PMOS, if the input is 0 the transistor is on, otherwise the transistor is off. Here is a graphical representation of these facts:



When a circuit contains both NMOS and PMOS transistors we say it is implemented in CMOS (Complementary MOS)

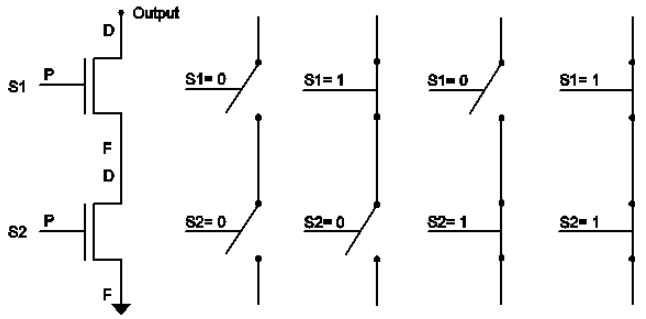
Understanding the basics of transistors, we can now design a simple NOR gate. Next figure shows the implementation in transistors of the NOR gate and how it works for different inputs (1 and 0). On the left there is the implementation, on the right the behavior. The symbol VDD is the source voltage (or the logic 1), GND is the ground (or the logical 0).



We have just seen how to implement a simple logic gate using transistors. To implement the rest of logical gates (and whatever circuit we might think off), we will analyze first the behavior of the transistors when connected in a “series” fashion or in a “parallel” way.

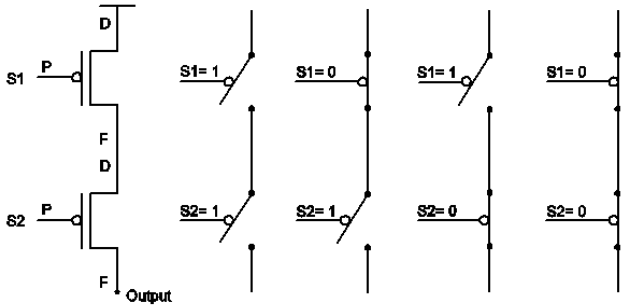
If we connect two NMOS transistors in series, we get the behaviour shown in next figure (the triangle in the bottom is a graphical representation of GND)

- Combinació serie nMOS:
  - Output = 0 si S1=1 i S2=1



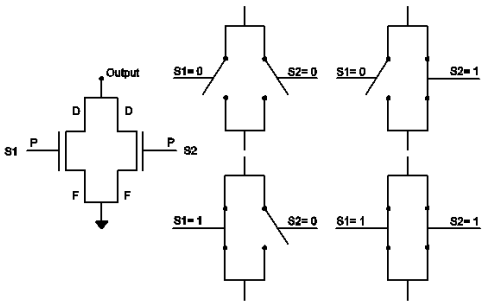
Next figure shows the behavior of the PMOSes when connected in series. (The horizontal line on top of the first transistor is a graphical representation of VDD).

- Combinació serie pMOS:
  - Output = 1 si S1=0 i S2=0

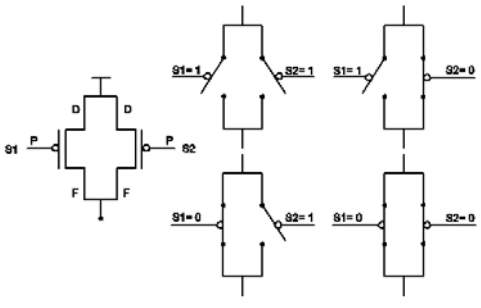


In the next figure, we can see the behavior of the NMOSes and PMOSes when connected in parallel.

Combinació paral·lela nMOS:
– Output = 0 si S1=1 o S2=1



Combinació paral·lela pMOS:
– Output = 1 si S1=0 o S2=0

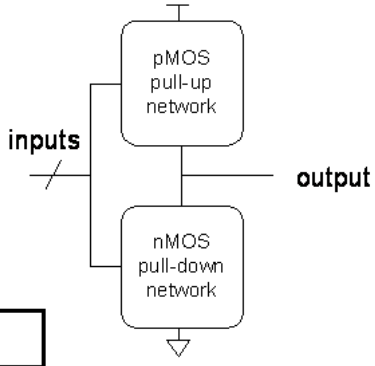


Summing up, NMOS transistors in series let the current flow when both inputs are 1; otherwise the output is undefined (Z). If we connect the NMOSes in parallel, then the current flows when any (or both) of the inputs are 1; otherwise the output is undefined (Z).

For the PMOSes, when connected in series the current flows when both inputs are 0; otherwise the output is undefined. Alternatively, when connected in parallel, if any (or both) of the inputs is 0 the current flows. Otherwise the output is undefined.

When using CMOS technology (and specifically static CMOS), we will design the circuits with two clearly defined parts. One (called *pull-up*) will be built of PMOS transistors and it has the duty of setting the output to 1 whenever the implemented function defines it. The other part (called *pull-down*) will be built of NMOS transistors and it will set the output to 0 whenever the implemented function defines it. All circuits will either set the output to 1 or 0 for any combination of the input values. Both *pull-up* and *pull-down* cannot be active at the same time (it makes no sense to set the output to 1 and 0 for the same inputs!!). Similarly, both the *pull-up* and the *pull-down* cannot be off at the same time (logic functions have always a defined output –either 0 or 1). Nevertheless, we will see further down the course that when not implementing logic functions we might be interested –sometimes- in setting the output to undetermined in certain cases.

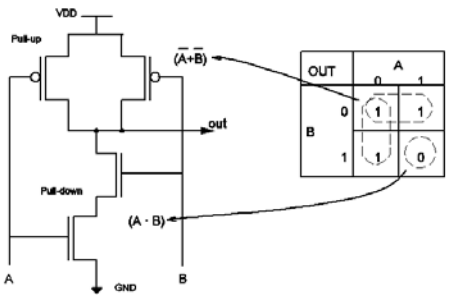
- Complementary CMOS logic gates
  - nMOS *pull-down network*
  - pMOS *pull-up network*
  - a.k.a. static CMOS



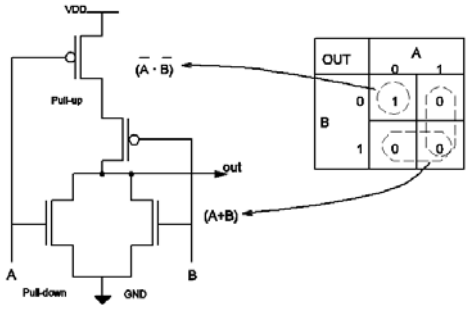
	Pull-up OFF	Pull-up ON
Pull-down OFF	Z (float)	1
Pull-down ON	0	X (crowbar)

Next figures show the implementations of the NAND and NOR gates in CMOS. For each one of them, there is the truth table and clear indications of what outputs are set by the pull-up and what outputs for the pull-down.

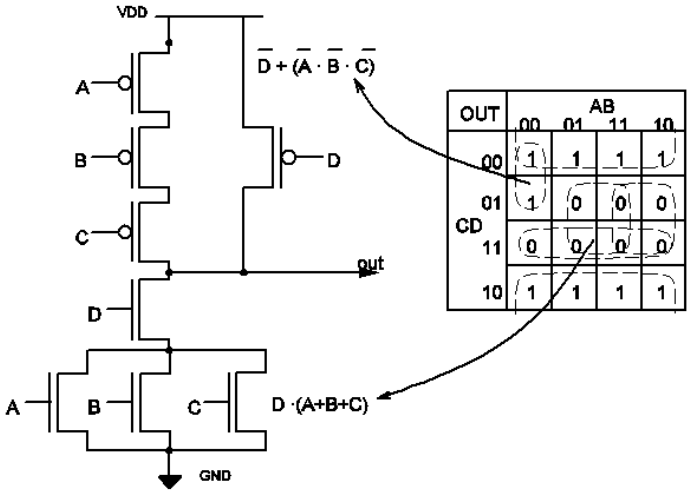
Porta NAND



Porta NOR



In the same way we implement the logic gates, we can implement any logic function. Next figure shows the implementation of the logic function $f(A,B,C,D) = \overline{D \cdot (A+B+C)}$.



We can actually define a design methodology for turning logical functions into CMOS circuits:

- The logic function must be complemented. (i.e., it must look like f(x,y,z)= NOT(*expression*); in a logic expression, $f(x,y,z) = \overline{\text{exp}\,ression}$
- AND operator ("·"):
  - Pull-down: NMOS transistors NMOS in series
  - Pull-up: PMOS transistors in parallel
- OR operator ("+"):
  - Pull-down: NOMS transistors in parallel
  - Pull-up: PMOS transistors in series

Problem sets to hand in:

Implement the following functions in CMOS logic.

- $Z = \overline{A \cdot B \cdot C \cdot D}$
- $Z = \overline{A + B + C + D}$
- $Z = \overline{((A \cdot B \cdot C) + D)}$
- $Z = \overline{(((A \cdot B) + C) \cdot D)}$
- $Z = \overline{(A \cdot B) + (C \cdot (A + B))}$

Implement the following functions in CMOS logic. Assume you can use a NOT gate whenever necessary.

- $Z = A$ (buffer)
- $Z = A \cdot \overline{B} + \overline{A} \cdot B$ (XOR)
- $Z = A \cdot B + \overline{A} \cdot \overline{B}$ (XNOR)
- $Z = A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot B \cdot C$ (funció suma en un sumador binari)