Algorithmes et complexité temporelle

1. Recherche du minimum dans une liste

```
algo1 (L, n)
    x ← L(1)
    Pour i allant de 2 à n
        si L(i) < x, alors x ← L(i)
    Sortir x
```

Comp. temp. $= \Theta(n)$

2. Somme des n premiers nombres entiers

```
algo2(n)
|  S ← 0
|  Pour i allant de 1 à n :
|      S ← S + i
|  Sortir S
```

Comp. temp. ~~$\Theta(n)$~~

$\Theta(n \log_2 n)$

$\log_2 n$ op.

$\sim n$: $a * b$ $\mathcal{O}((\log_2 n)^2)$ op.

Pour représenter n, il faut $\sim \log_2(n)$ bits

3. Somme des 2^n premiers nbs entiers

```
algo3 (n)
    S ← 0
    Pour i allant 1 à 2^n
        S ← S+i
    Sortir S
```

Comp. temp. $\Theta(2^n \cdot n)$

- $X+Y$ avec $X, Y \sim 2^n$ → n opérations
- $X*Y$ avec $X, Y \sim 2^n$ → n^2 opérations

Rappels

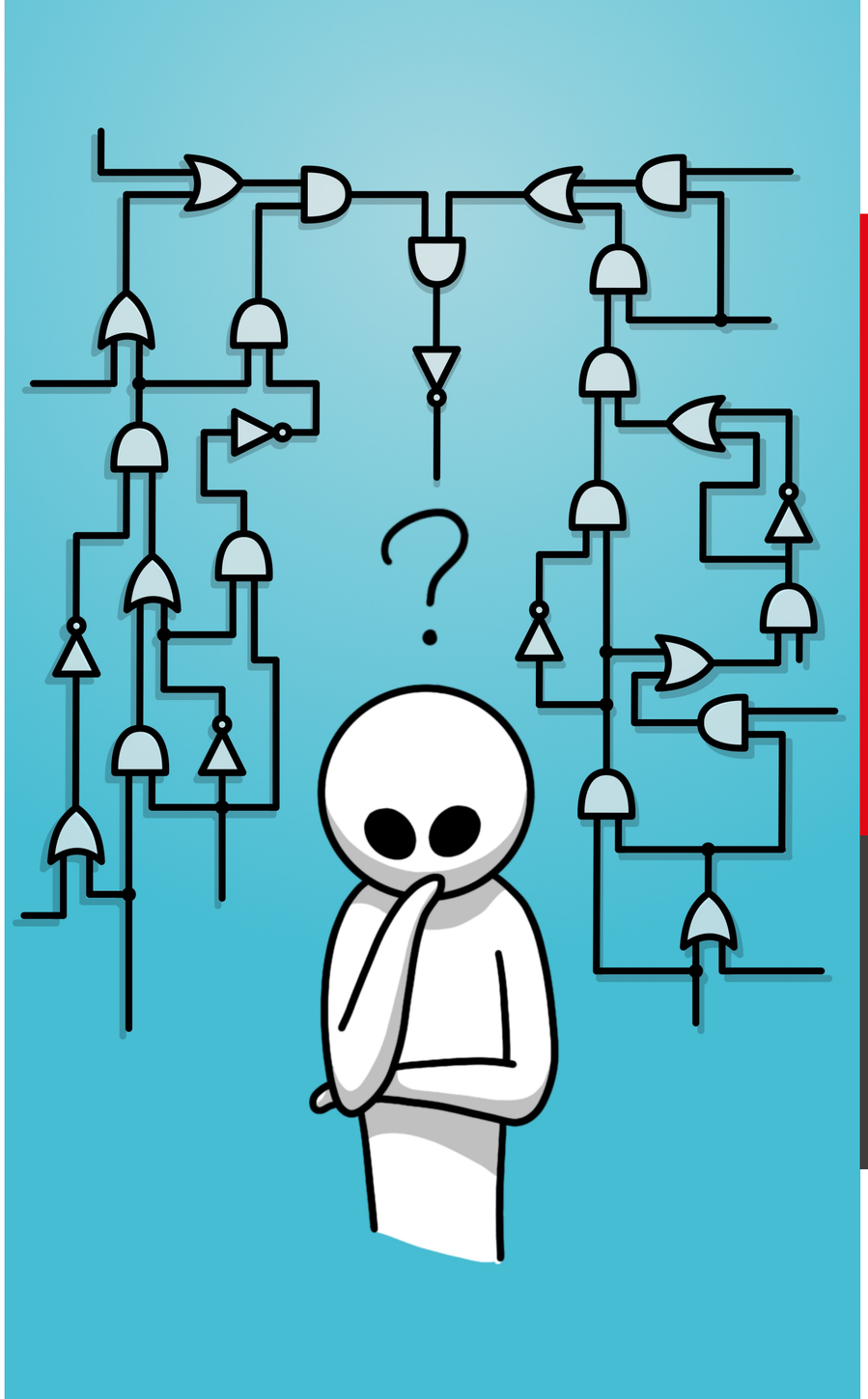
- Représentation binaire des nombres :
  - nombres entiers positifs et relatifs
  - nombres réels (représentation inexacte; erreurs absolue et relative)

- Opérations sur les nombres entiers :

  = “bonne vieille” addition / multiplication en colonnes, mais en binaire

  (avec les règles 1 + 1 = 10, 1 + 1 + 1 = 11)

- Aujourd’hui: comment faire ça avec des circuits logiques



Information, Calcul et Communication

Circuits logiques

Olivier Lévêque

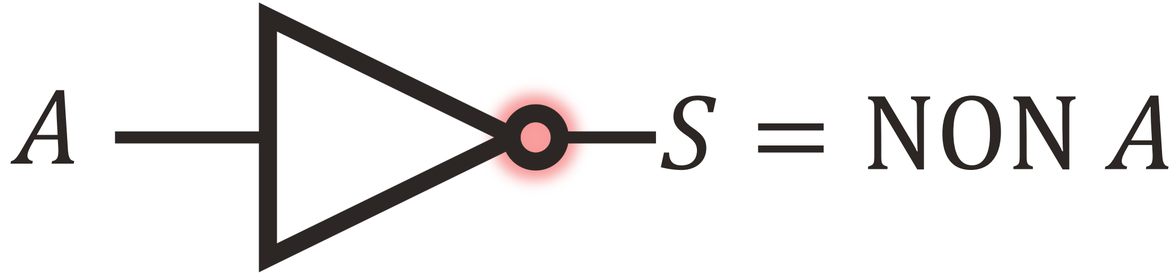
École polytechnique fédérale de Lausanne

Introduction

- Un **circuit logique** est un ensemble de **portes logiques** reliées entre elles.
- Ces portes logiques permettent de réaliser des **opérations élémentaires** sur des bits.
- Chaque porte logique est caractérisée par **une table de vérité** établissant une correspondance entre les **entrées** et les **sorties** de cette porte.
- Chaque porte logique est également représentée par un **symbole**.
- Nous verrons que l'on peut combiner plusieurs portes logiques ensemble pour faire tout type d'opération, comme un **additionneur**, par exemple.

La porte NON *(NOT)*

- Elle possède une seule entrée



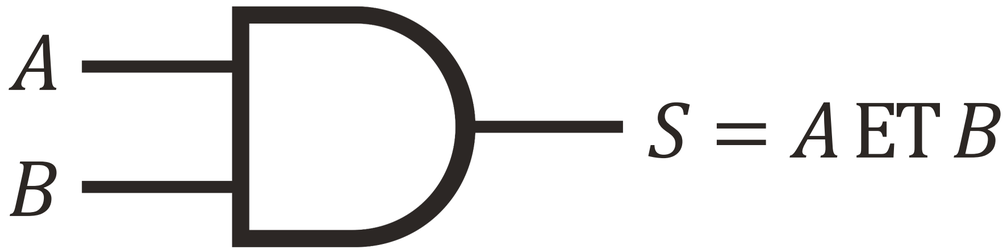
- La porte NON donne en sortie, **l'inverse** de la valeur du bit d'entrée

- Notez que le cercle à la sortie d'une porte logique signifiera toujours l'inverse

NON	
A	$S = \text{NON}\, A$
0	1
1	0

La porte ET *(AND)*

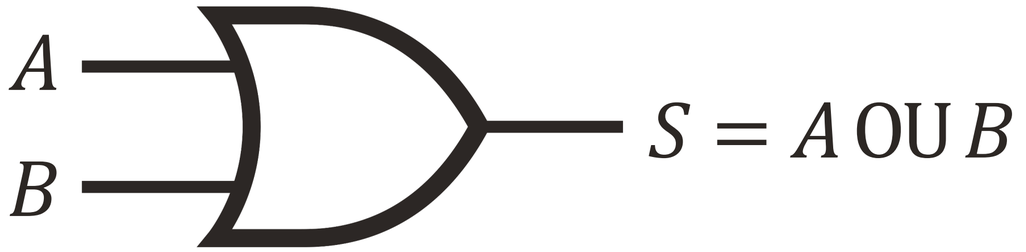
- Elle comporte deux ou plusieurs entrées.
- La porte ET génère un 1 en sortie si et seulement si les deux bits en entrée sont égaux à 1. Dans le cas contraire, la sortie vaut 0.
- Notez que la valeur de la sortie S correspond au produit des valeurs d'entrées $A \cdot B$.



ET		
A	B	$S = A \text{ ET } B$
0	0	0
0	1	0
1	0	0
1	1	1

La porte OU *(OR)*

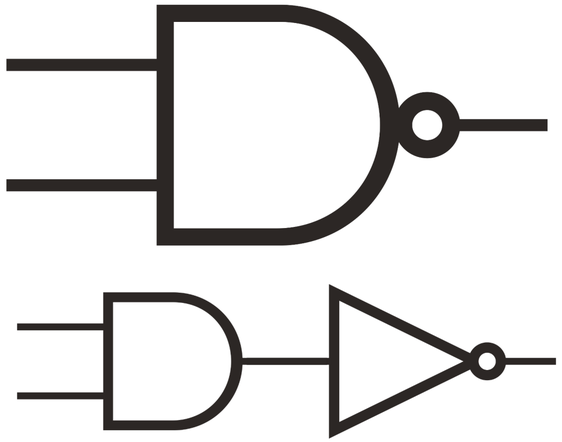
- Elle comporte deux ou plusieurs entrées.
- La porte OU génère un 1 en sortie si au moins un des bits en entrée vaut 1. La sortie vaut donc 0 en sortie si et seulement si les deux bits en entrée valent 0.
- Notez que la valeur de sortie S vaut 1 quand $A + B \geq 1$ (mais n'est donc **pas** égale à $A + B$).



OU		
A	B	$S = A$ OU B
0	0	0
0	1	1
1	0	1
1	1	1

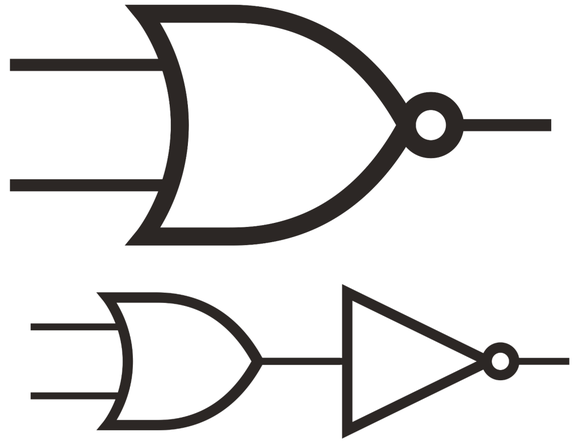
Les portes NON ET *(NAND)* et NON OU *(NOR)*

- Porte NON ET



NON ET		
A	B	$S = \text{NON}(A \text{ ET } B)$
0	0	1
0	1	1
1	0	1
1	1	0

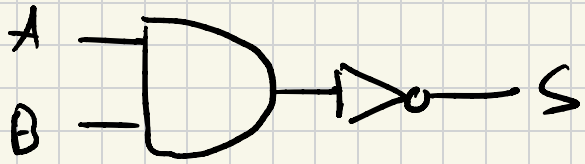
- Porte NON OU

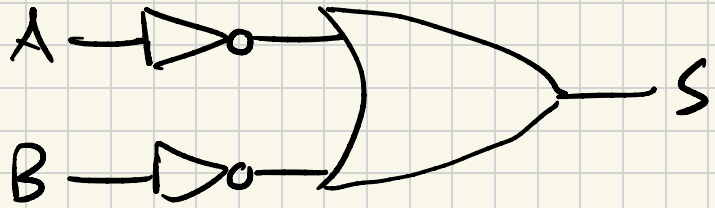


NON OU		
A	B	$S = \text{NON}(A \text{ OU } B)$
0	0	1
0	1	0
1	0	0
1	1	0

Loi de De Morgan:

NON (A ET B) = (NON A) OU (NON B)





A	B	S	
0	0	1	✓
0	1	1	✓
1	0	1	✓
1	1	0	✓

En pratique

- Avec les trois portes de base (NON, ET, OU), on peut créer tous les circuits possibles et donc effectuer toutes les opérations possibles.

- Il est possible de représenter une porte logique comme étant la composition d'autres portes logiques.

- En électronique, la porte NON ET est la plus simple à réaliser du point de vue technologique. Pour cette raison, elle sert souvent de **brique de base** aux circuits intégrés. On peut reconstituer toutes les fonctions logiques uniquement à l'aide de portes NON ET.

Additionner deux bits *(sans retenue)*

- On aimerait créer un circuit avec entrées A et B et sortie S dont la table de vérité soit :

Additionneur		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

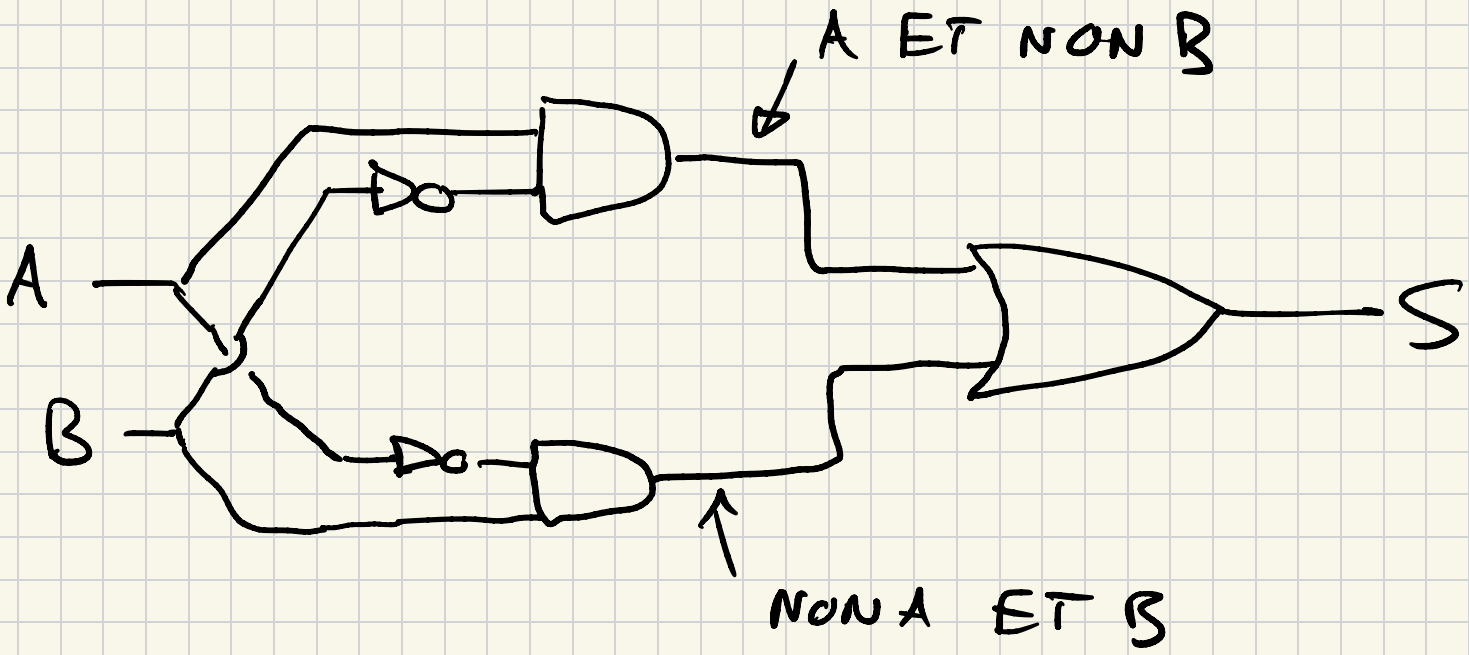
- Pour créer ce circuit, remarquez que : $S = 1$ si et seulement si :

$$(A = 1 \text{ ET } B = 0) \text{ OU } (A = 0 \text{ ET } B = 1)$$

- Autrement dit:

$$S = (A \text{ ET NON } B) \text{ OU } (\text{NON } A \text{ ET } B)$$

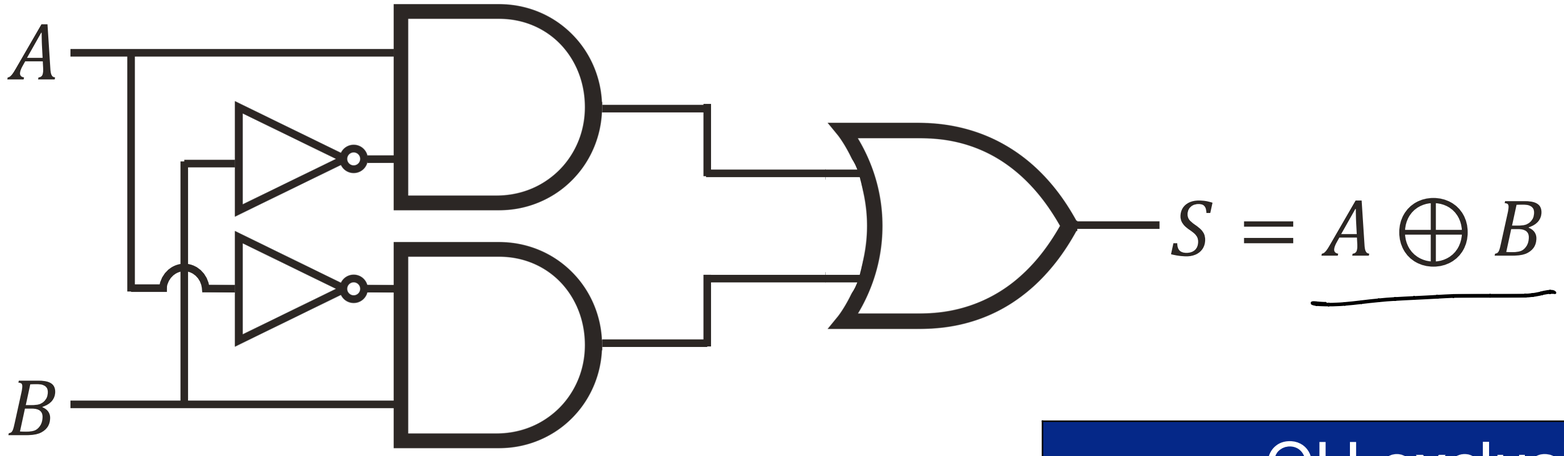
$A \oplus B$ (addition modulo 2)



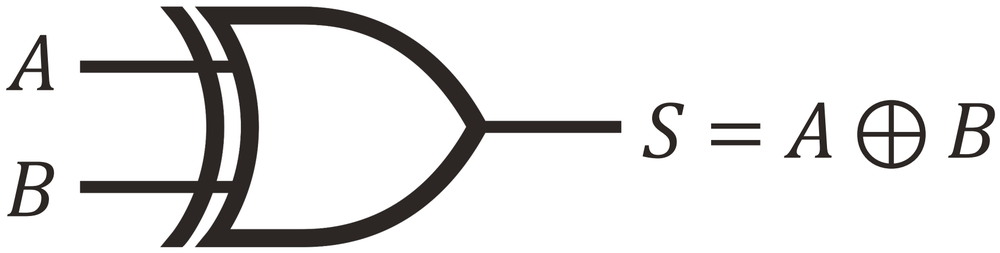
Additionner deux bits: la porte OU Exclusif *(XOR)*

= addition modulo 2

- Circuit correspondant :



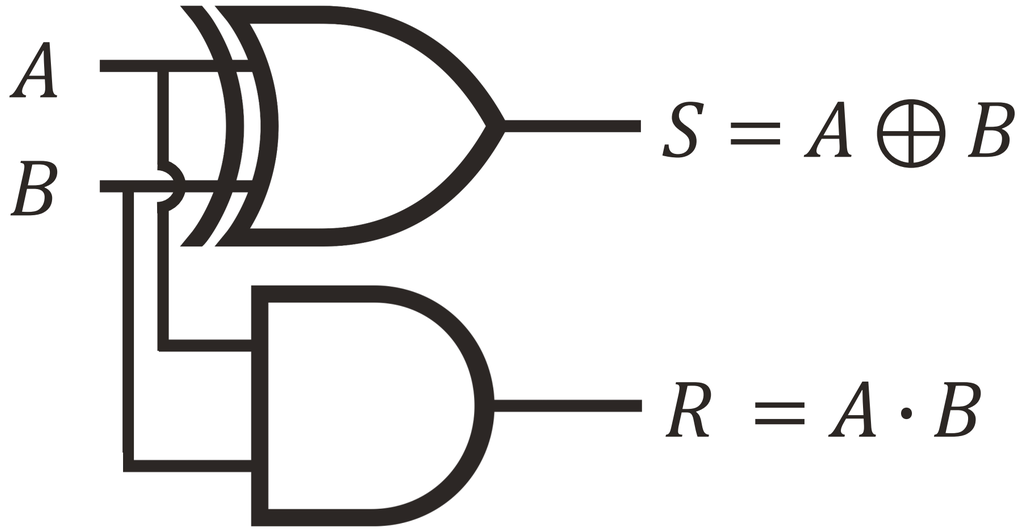
- Symbole résumant ce nouveau circuit :



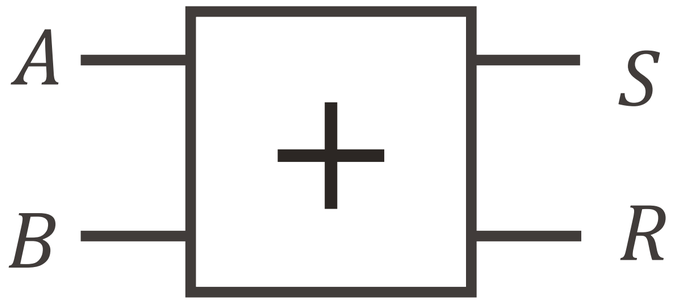
OU exclusif		
A	B	$S = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Additionner deux bits *(avec retenue)*

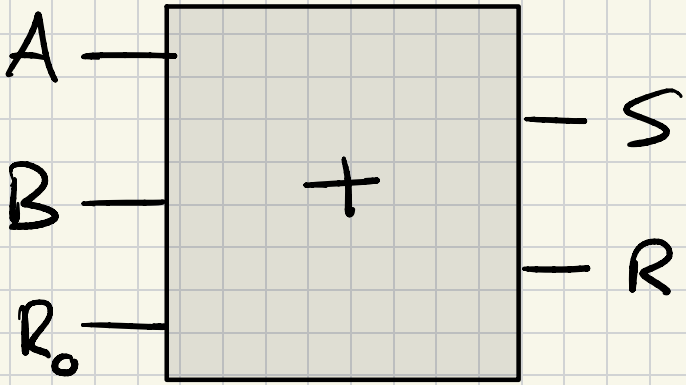
- On aimerait maintenant créer un circuit avec entrées A et B et sortie $S = A \oplus B$, ainsi qu'une retenue $\mathrm{R} = 1$ si et seulement si $A = 1$ et $B = 1$.



Symbole :



- Exercice : créer un circuit avec une retenue de plus en entrée, soit un additionneur avec **trois entrées** A, B et R_0 et **deux sorties** S et R_1.



A	B	R_0	R	S
0	0	0	0	0
0	0	1	0	1
0	1	1	1	0
1	1	1	1	1
0	1	0	-	-
1	0	0	-	-
1	0	1	-	-
1	1	0	-	-

Notre but : additionner des nombres !

- Rappel

Avec des nombres entiers :

$$\begin{array}{r} {\scriptstyle 1\ 1} \\ 57 \\ +\ 43 \\ \hline = 100 \end{array}$$

Avec des bits :

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\quad 1\ 1} \\ 111001 \\ +\ 101011 \\ \hline = 1100100 \end{array}$$

- La règle d'addition est la même !
  Il faut juste se rappeler que $1 + 1 = 10$ et $1 + 1 + 1 = 11$ en binaire.

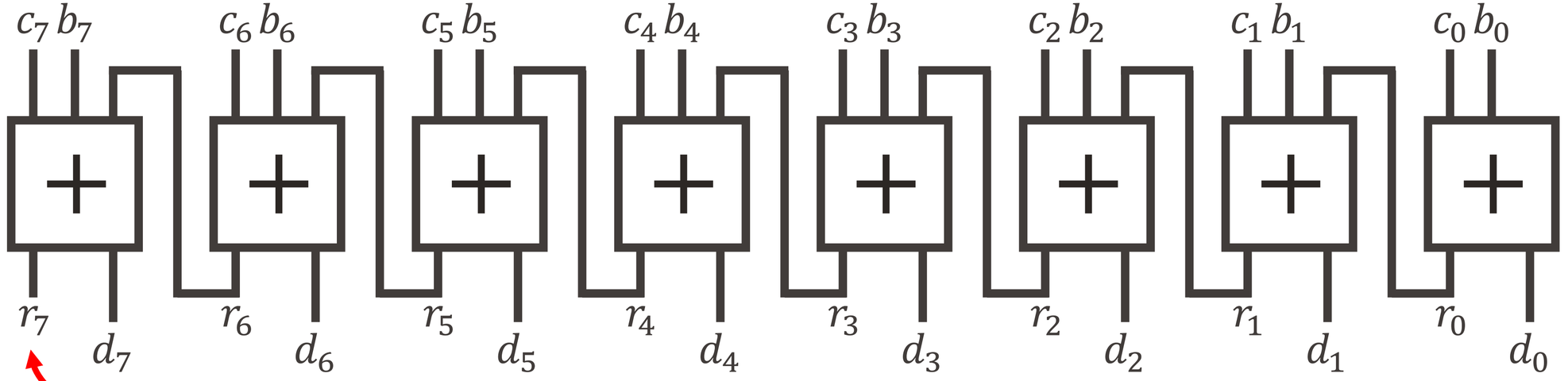
Additionneur sur 8 bits

Effectuer :

$$\begin{array}{r} b_7 b_6 b_5 \ldots b_1 b_0 \\ +\ c_7 c_6 c_5 \ldots c_1 c_0 \\ \hline = \ d_7 d_6 d_5 \ldots d_1 d_0 \end{array}$$

Exemple :

$$\begin{array}{r} 1\,1\,1\quad 1\,1 \\ 00111001 \\ +\ 00101011 \\ \hline = 01100100 \end{array}$$



Si $r_7 = 1 \Rightarrow$ overflow !

Information, Calcul et Communication

Transistors

École polytechnique fédérale de Lausanne

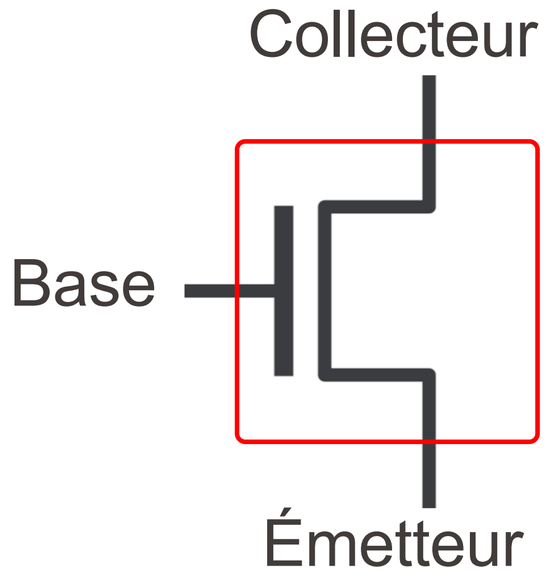
Olivier Lévêque

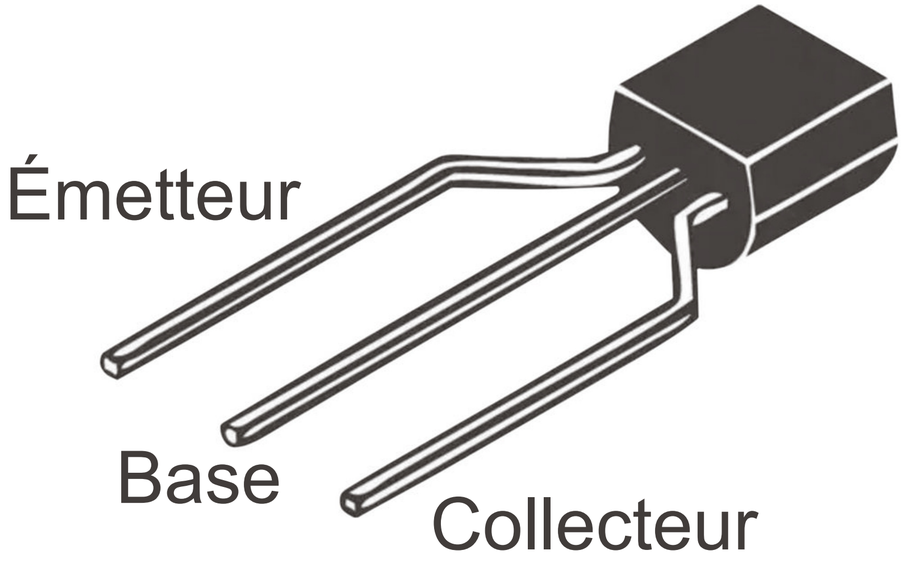
Le transistor

- Inventé en 1947 par trois américains : Bardeen, Shockley & Brattain

- Ce composant, qui est à la base de toute l'électronique moderne, a remplacé avantageusement les relais électromécaniques et les tubes à vide utilisés dans les premiers ordinateurs à la même époque → ***miniaturisation***

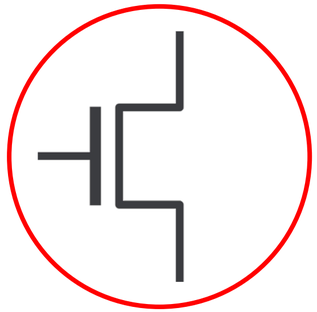
- Schéma :





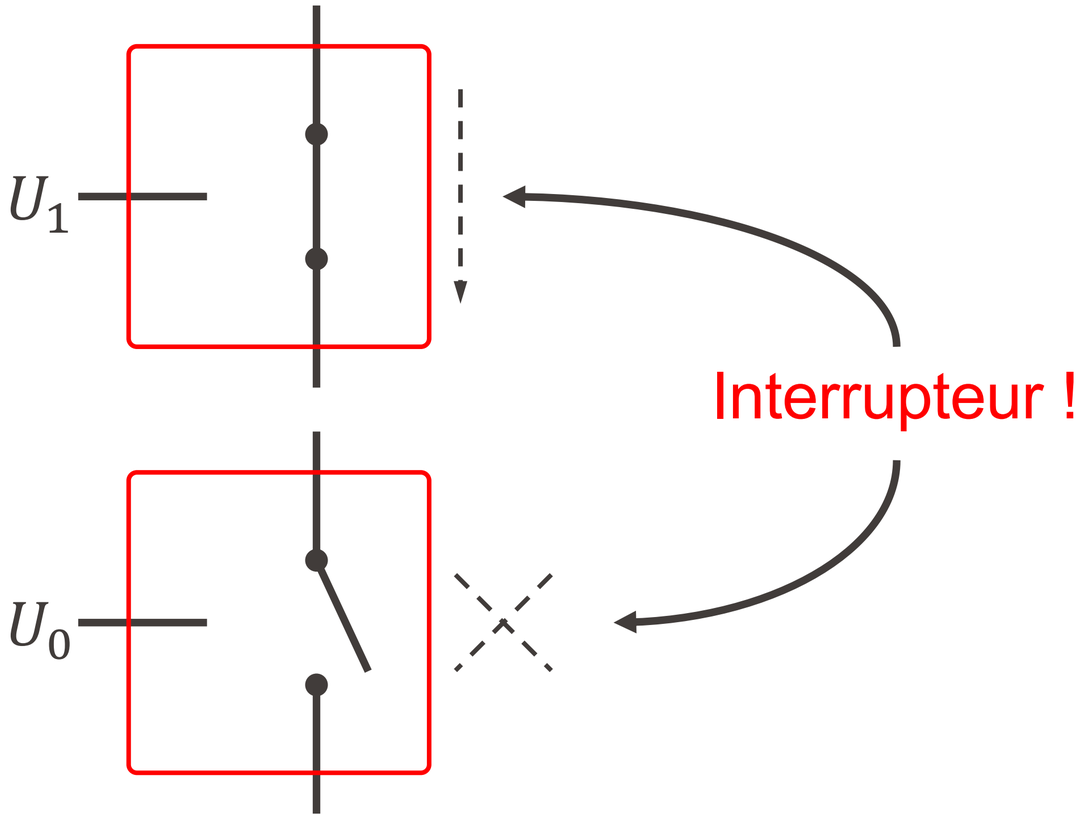
Principe de fonctionnement *(n-mos)*

- Symbole :



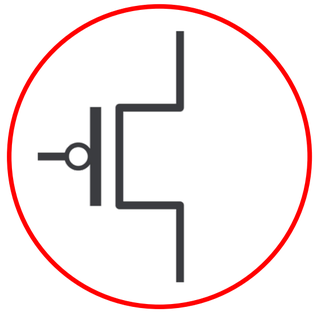
- Si la tension à la base est ***haute*** ($U_1 = 5V$) alors le courant passe entre l'émetteur et le collecteur :

- Si la tension à la base est ***basse*** ($U_0 = 0V$) alors le courant ne passe pas entre l'émetteur et le collecteur :

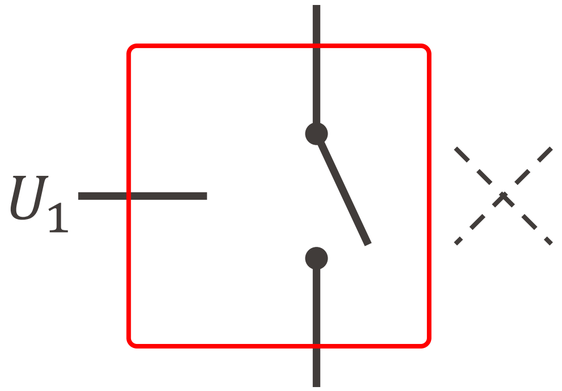


Principe de fonctionnement *(p-mos)*

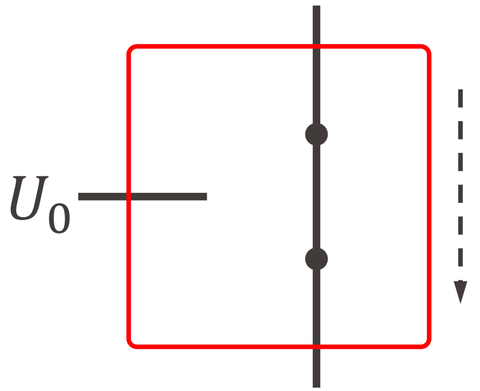
- Symbole :



- Si la tension à la base est ***haute*** ($U_1 = 5V$) alors le courant ne passe pas entre l'émetteur et le collecteur :

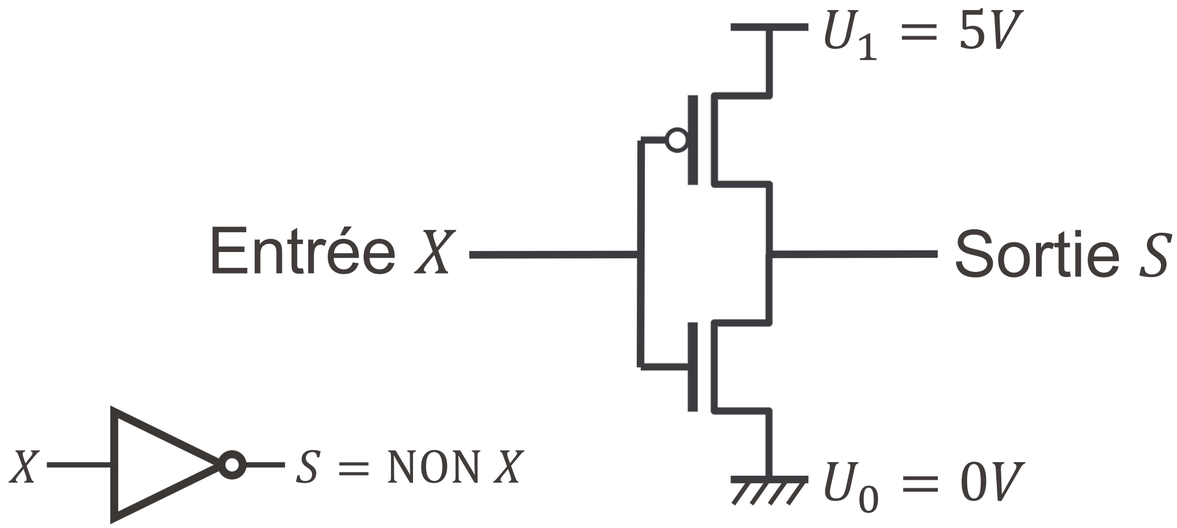


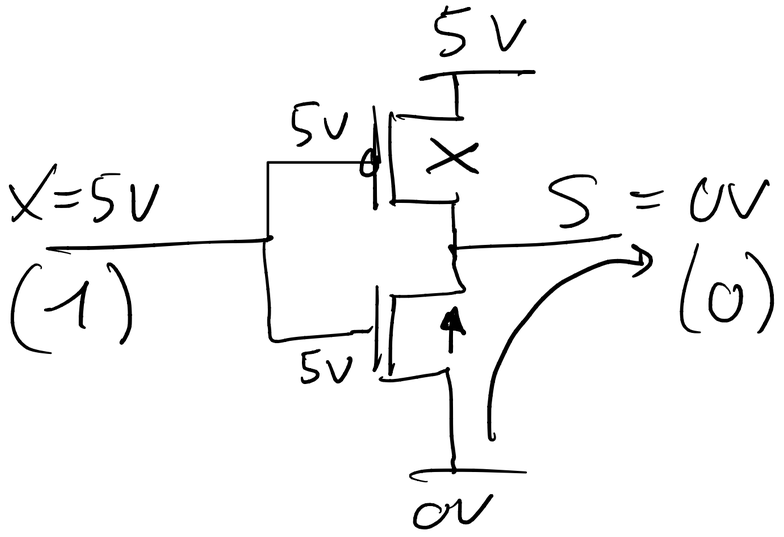
- Si la tension à la base est ***basse*** ($U_0 = 0V$) alors le courant passe entre l'émetteur et le collecteur :

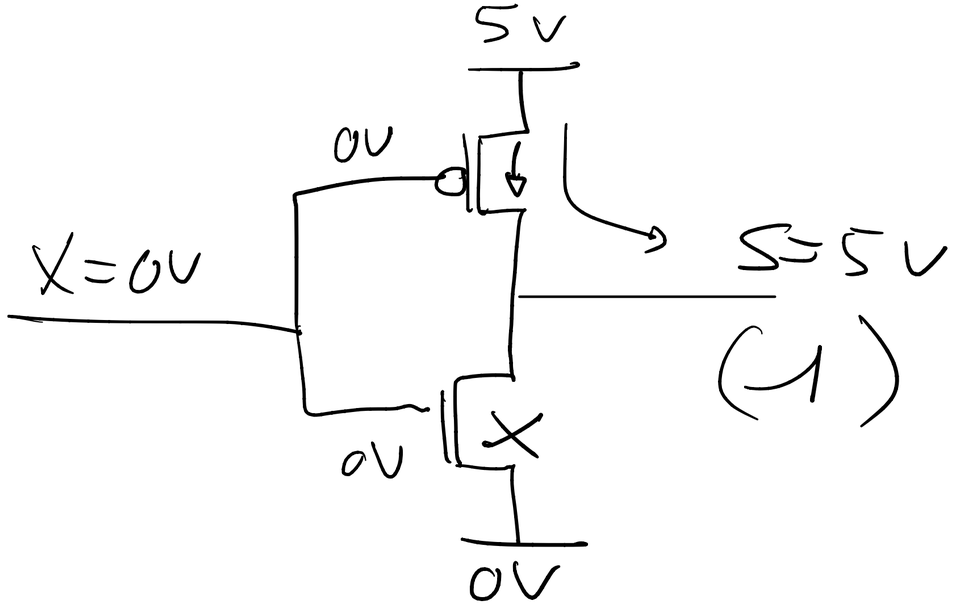


Création d'un inverseur

- Si on identifie U_0 comme 0 et U_1 comme 1, on peut créer un inverseur ***(porte NOT)*** à l'aide d'un transistor n-mos et d'un transistor p-mos

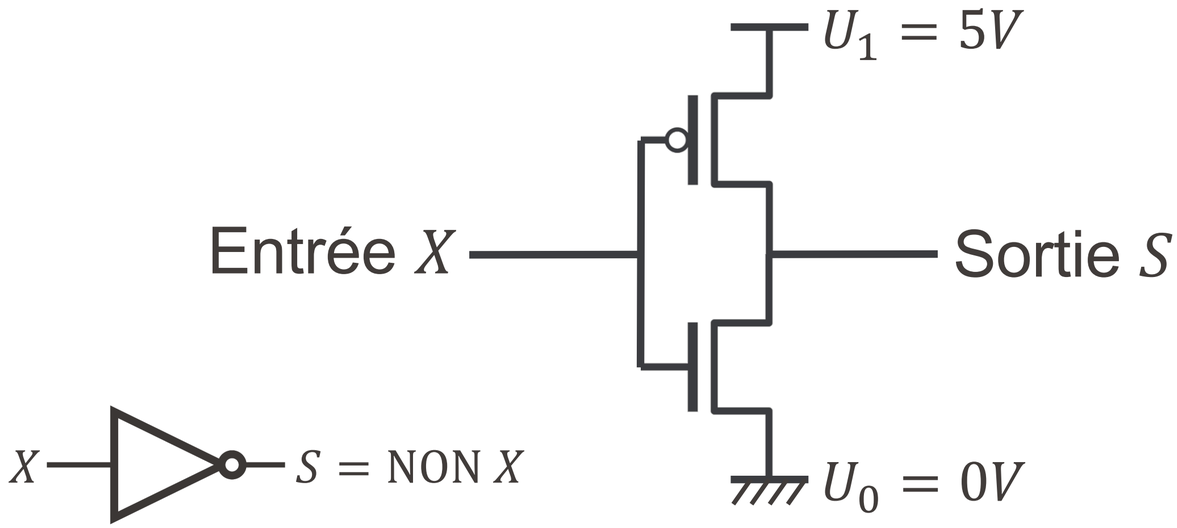


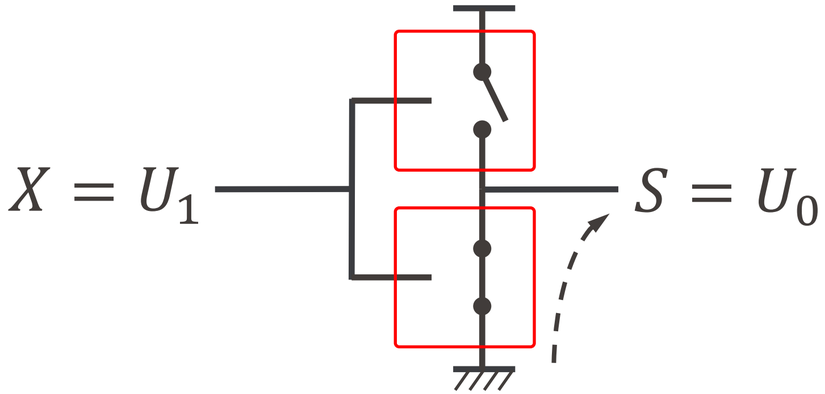


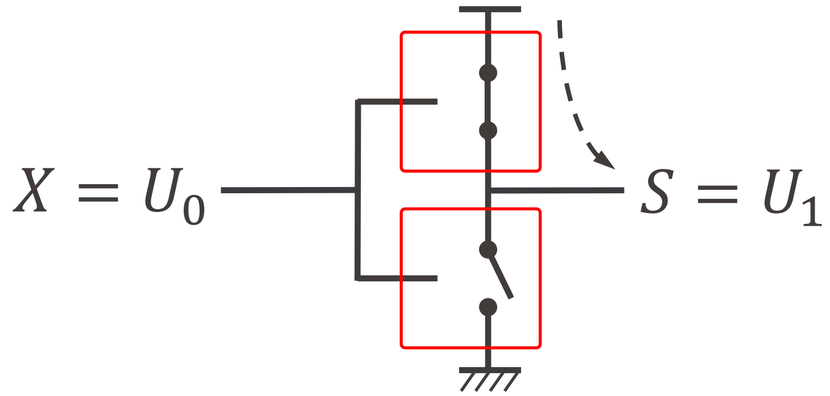


Création d'un inverseur

- Si on identifie U_0 comme 0 et U_1 comme 1, on peut créer un inverseur ***(porte NOT)*** à l'aide d'un transistor n-mos et d'un transistor p-mos

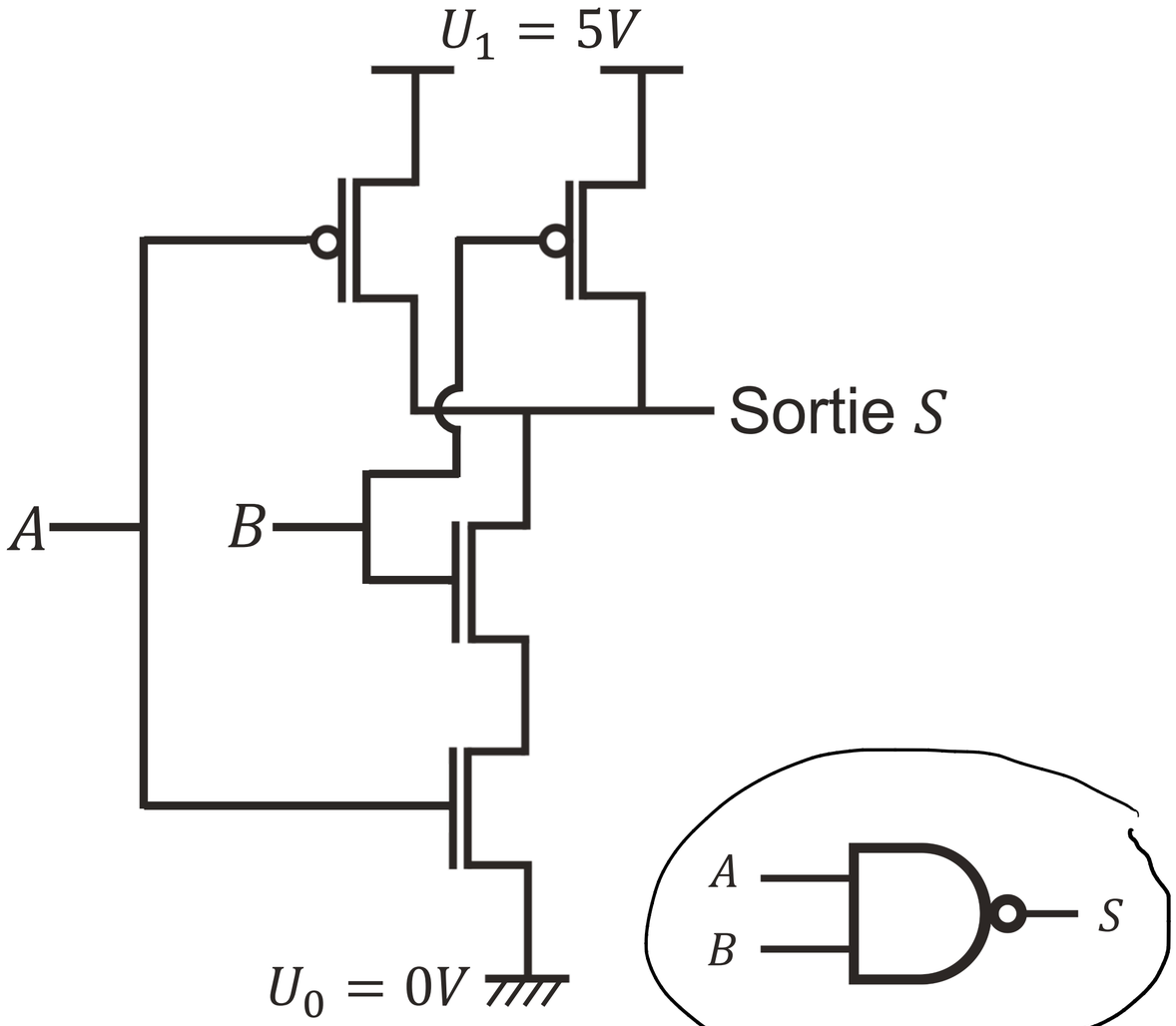






- En exercice : on peut généraliser aux portes AND et OR, il faut ***6 transistors*** pour créer ces portes (contre ***4*** pour les portes NAND et NOR).

Création de la porte NAND



Porte ET

4 transistors

2 transistors

6 transistors