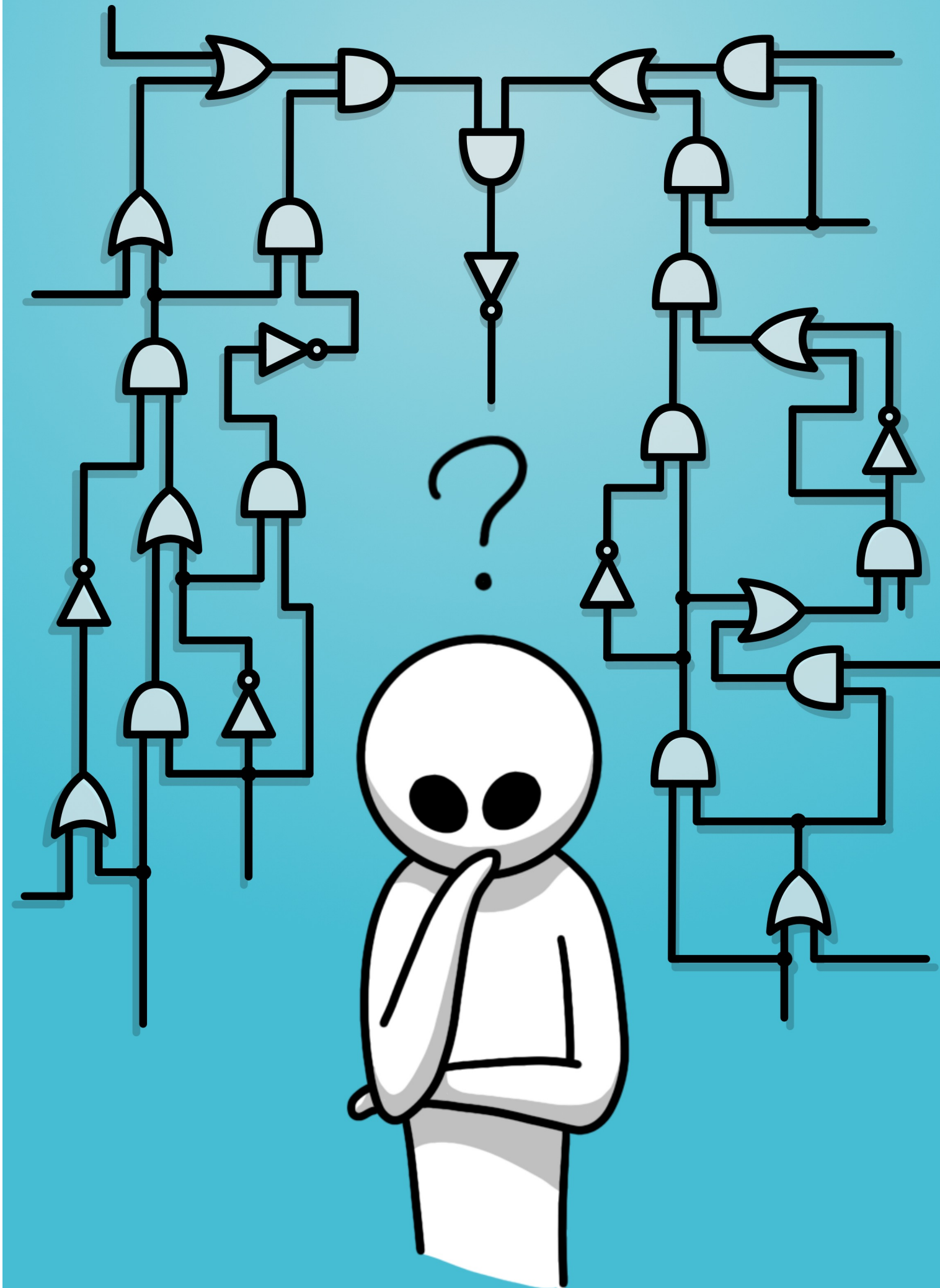


- Représentation binaire des nombres :
 - nombres entiers positifs et relatifs
 - nombres réels (représentation inexacte; erreurs absolue et relative)
- Opérations sur les nombres entiers :
 - = “bonne vieille” addition / multiplication en colonnes, mais en binaire (avec les règles $1 + 1 = 10$, $1 + 1 + 1 = 11$)
- Aujourd’hui: comment faire ça avec des circuits logiques



Information, Calcul et Communication

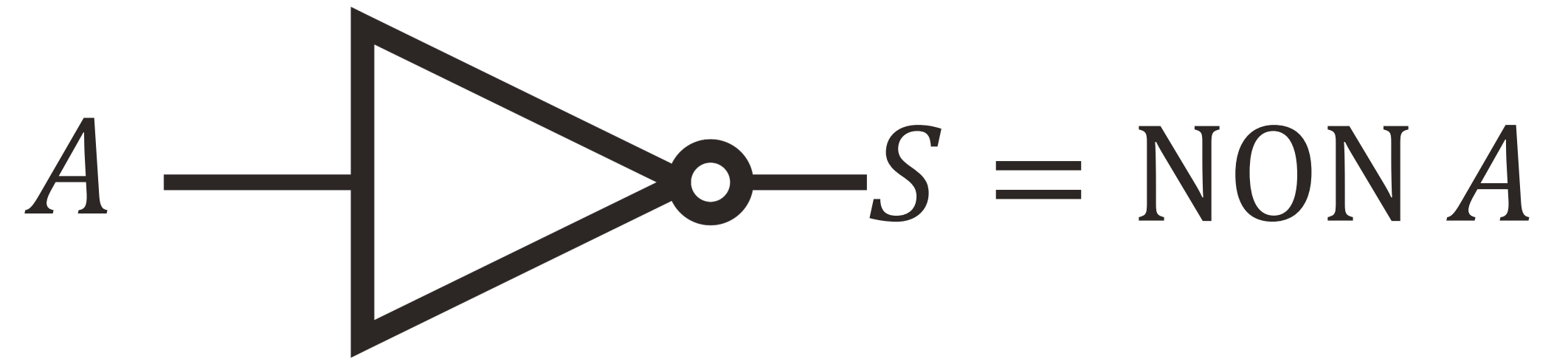
Circuits logiques

Olivier Lévêque

- 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

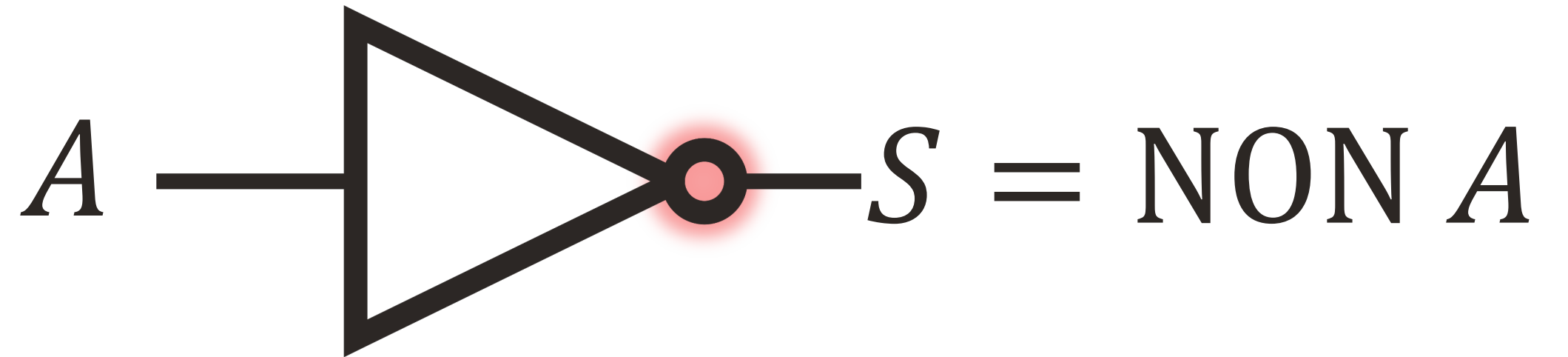


- La porte NON donne en sortie, l'**inverse** de la valeur du bit d'entrée

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

La porte NON (*NOT*)

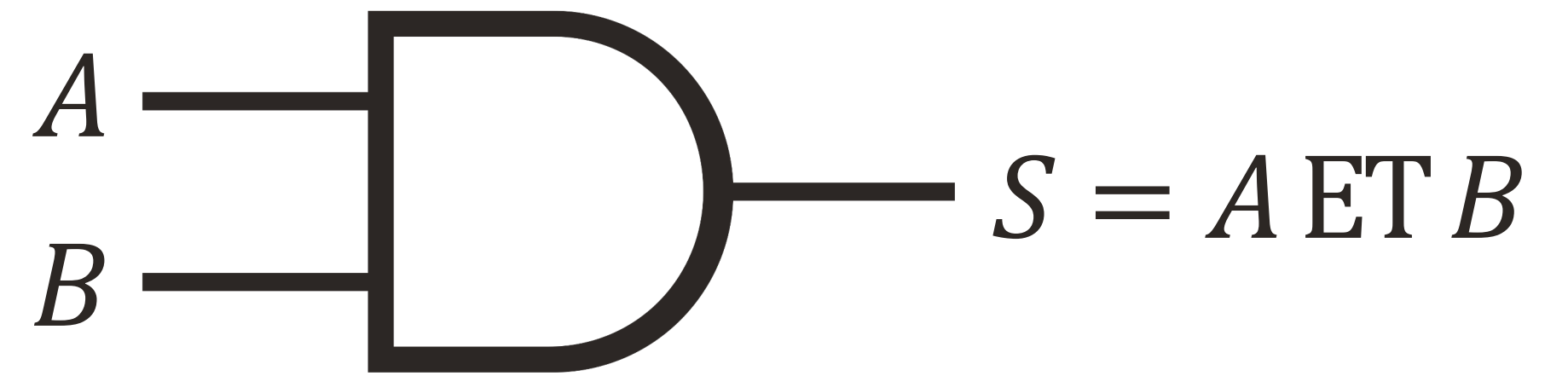
- Elle possède une seule entrée
- La porte NON donne en sortie, l'**inverse** de la valeur du bit d'entrée
- Notez que le cercle à la sortie d'une porte logique signifiera toujours l'inverse



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

La porte ET (*AND*)

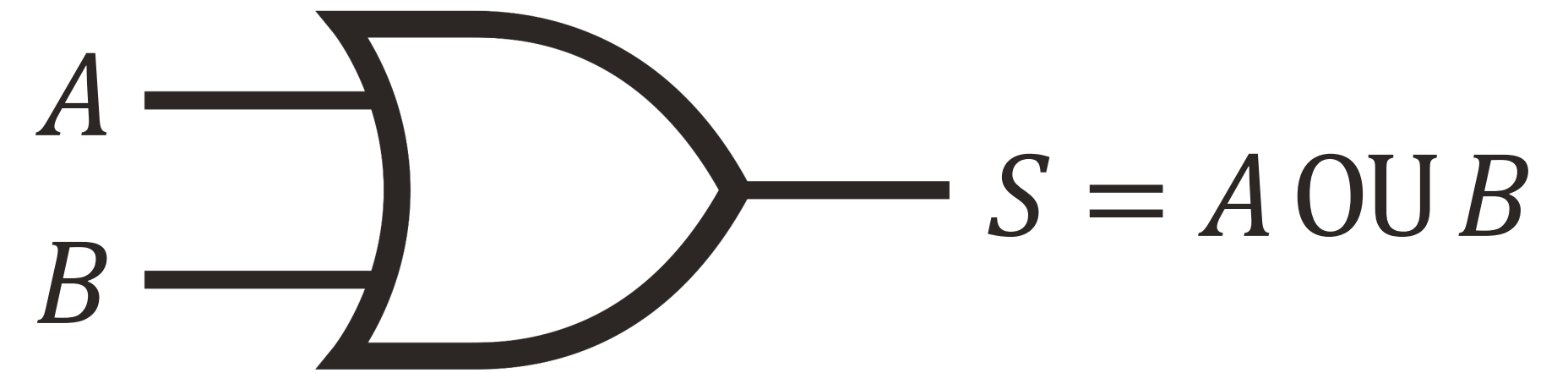
- Elle comporte deux ou plusieurs entrées.
- La porte ET génère un 1 en sortie si et seulement si les deux bits en entrée sont égaux à 1. Dans le cas contraire, la sortie vaut 0.
- Notez que la valeur de la sortie S correspond au produit des valeurs d'entrées $A \cdot B$.



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

La porte OU (*OR*)

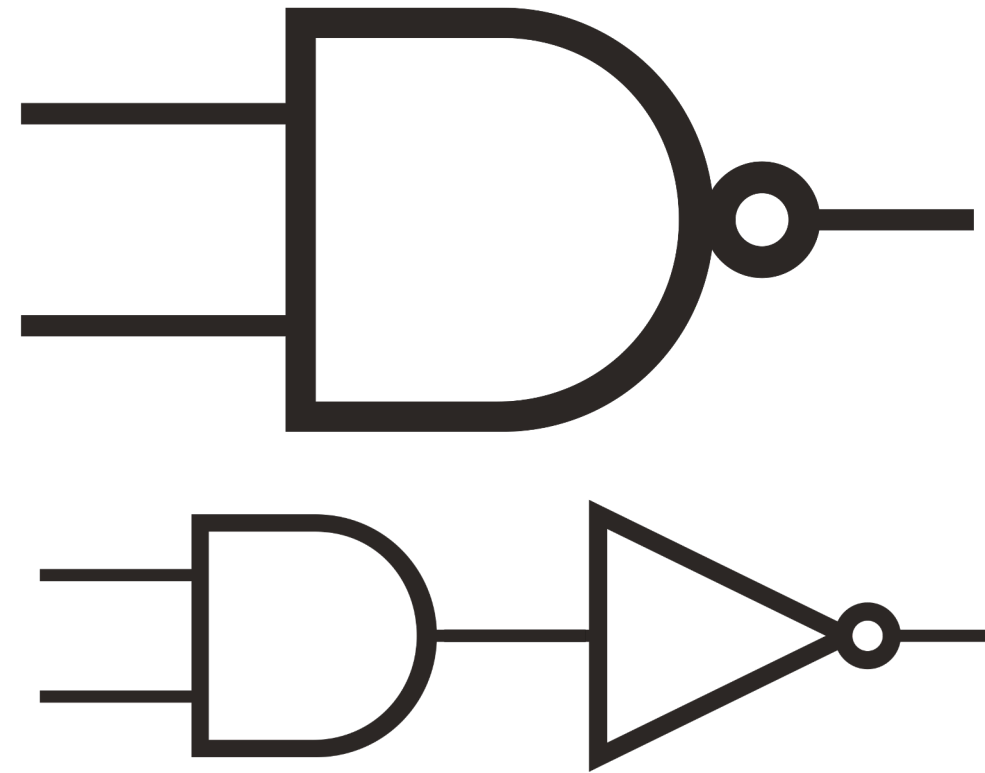
- Elle comporte deux ou plusieurs entrées.
- La porte OU génère un 1 en sortie si au moins un des bits en entrée vaut 1. La sortie vaut donc 0 en sortie si et seulement si les deux bits en entrée valent 0.
- Notez que la valeur de sortie S vaut 1 quand $A + B \geq 1$ (mais n'est donc **pas** égale à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



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

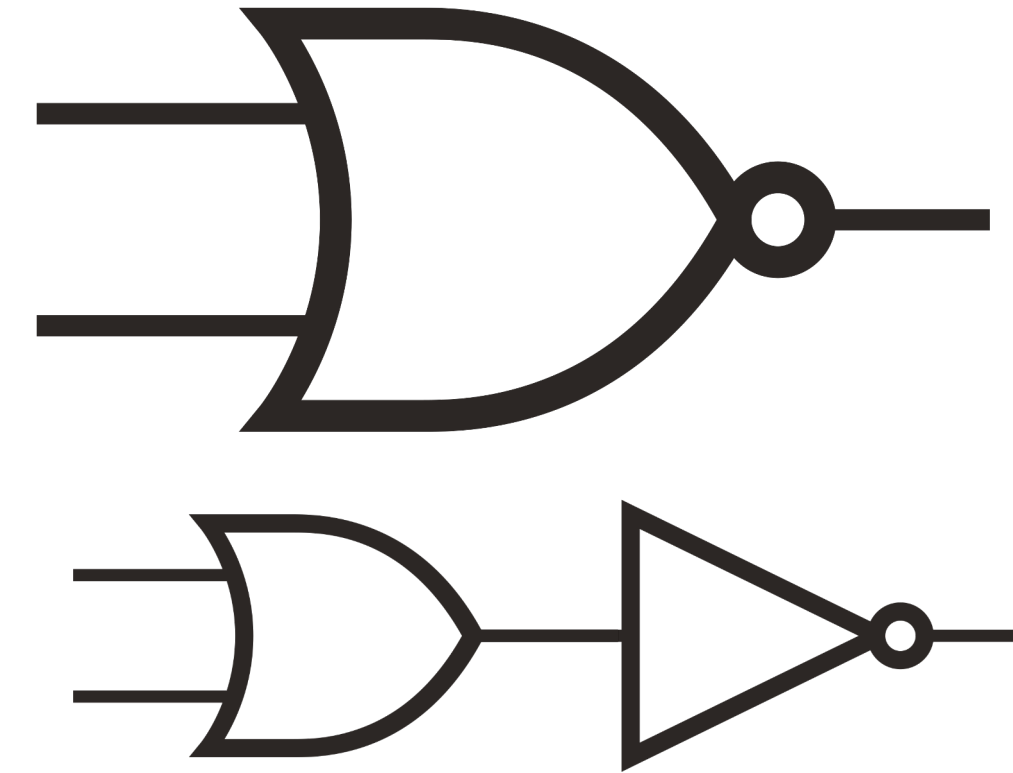
Les portes NON ET (*NAND*) et NON OU (*NOR*)

■ 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

- 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 :

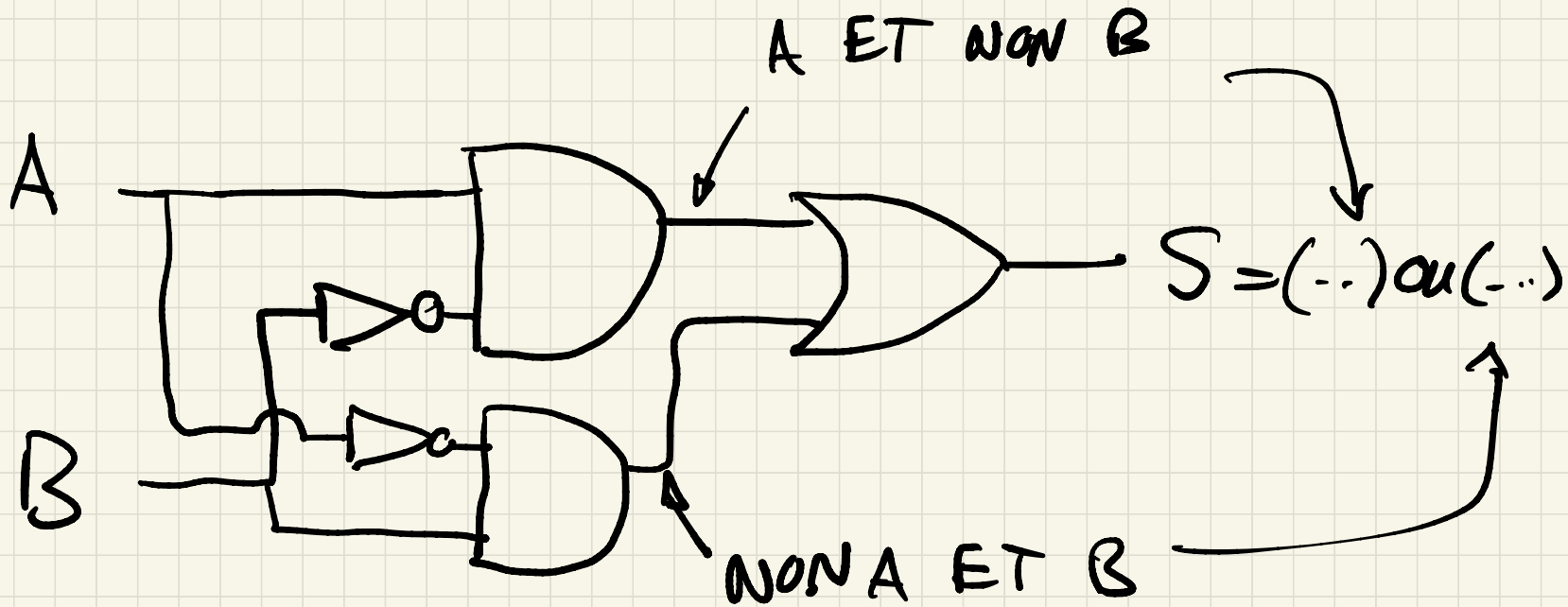
Additionneur		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

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

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

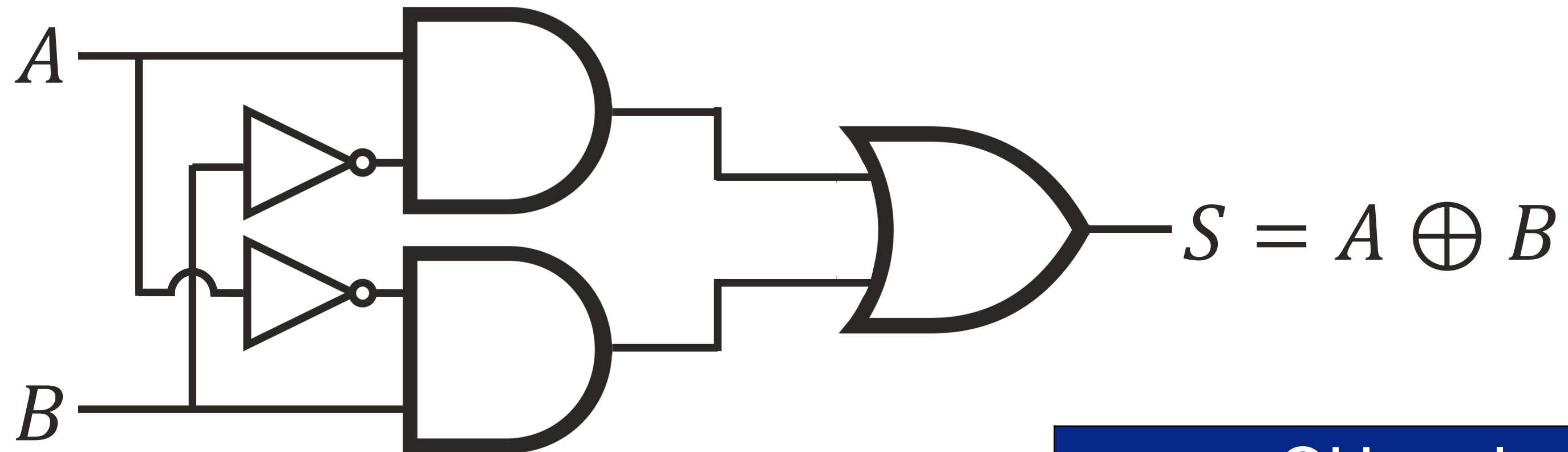
- Autrement dit:

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

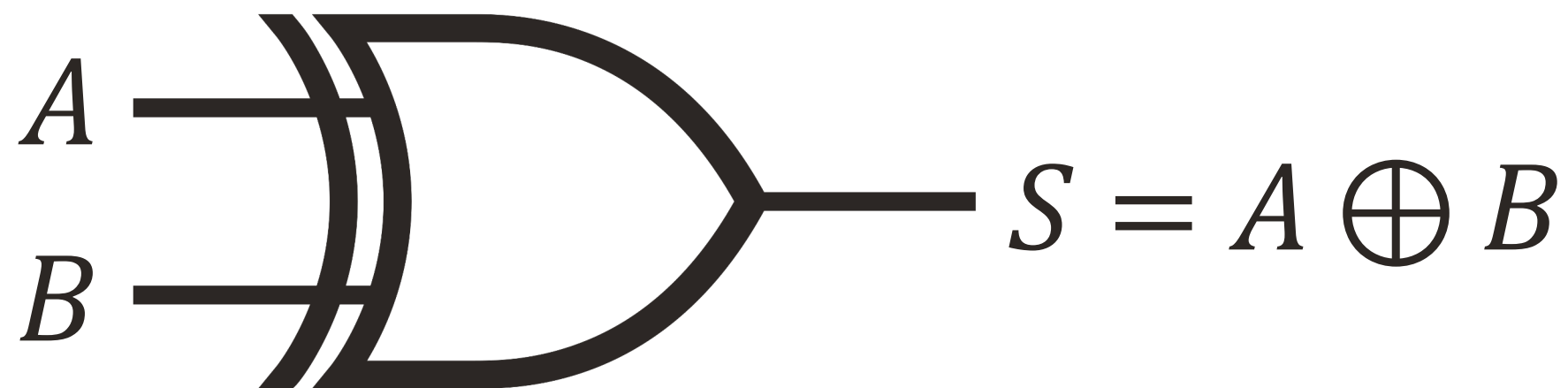


Additionner deux bits: la porte OU Exclusif (*XOR*)

- Circuit correspondant :



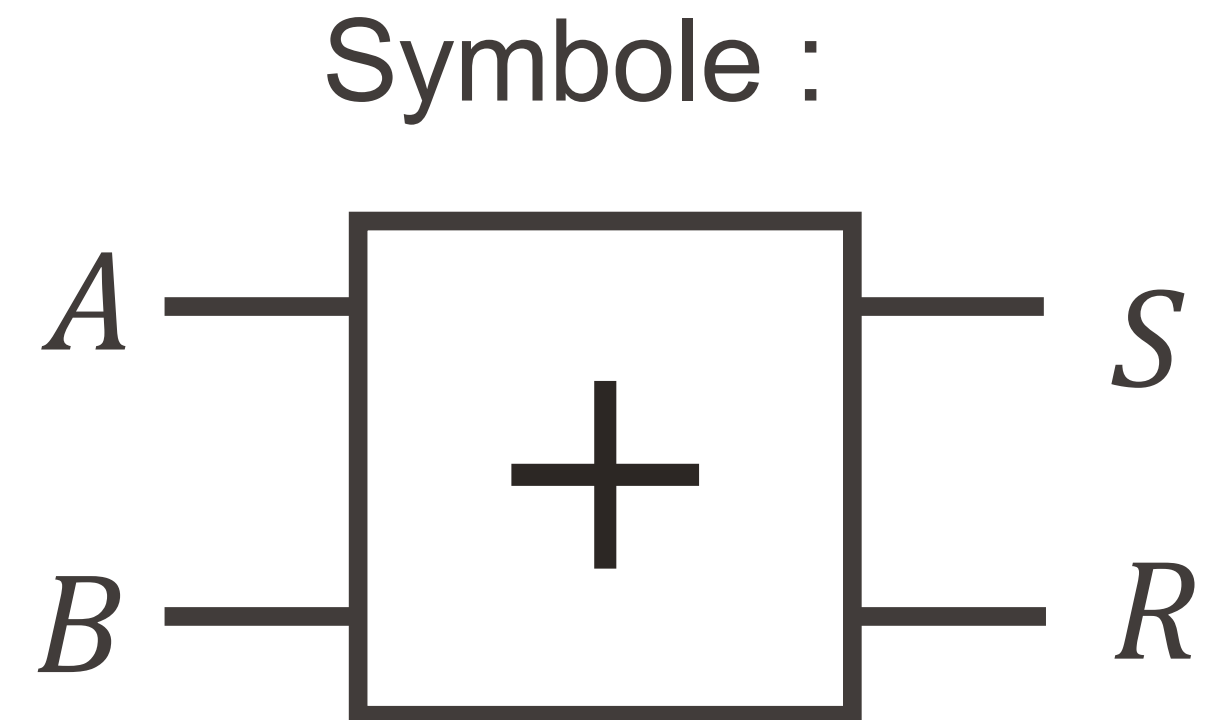
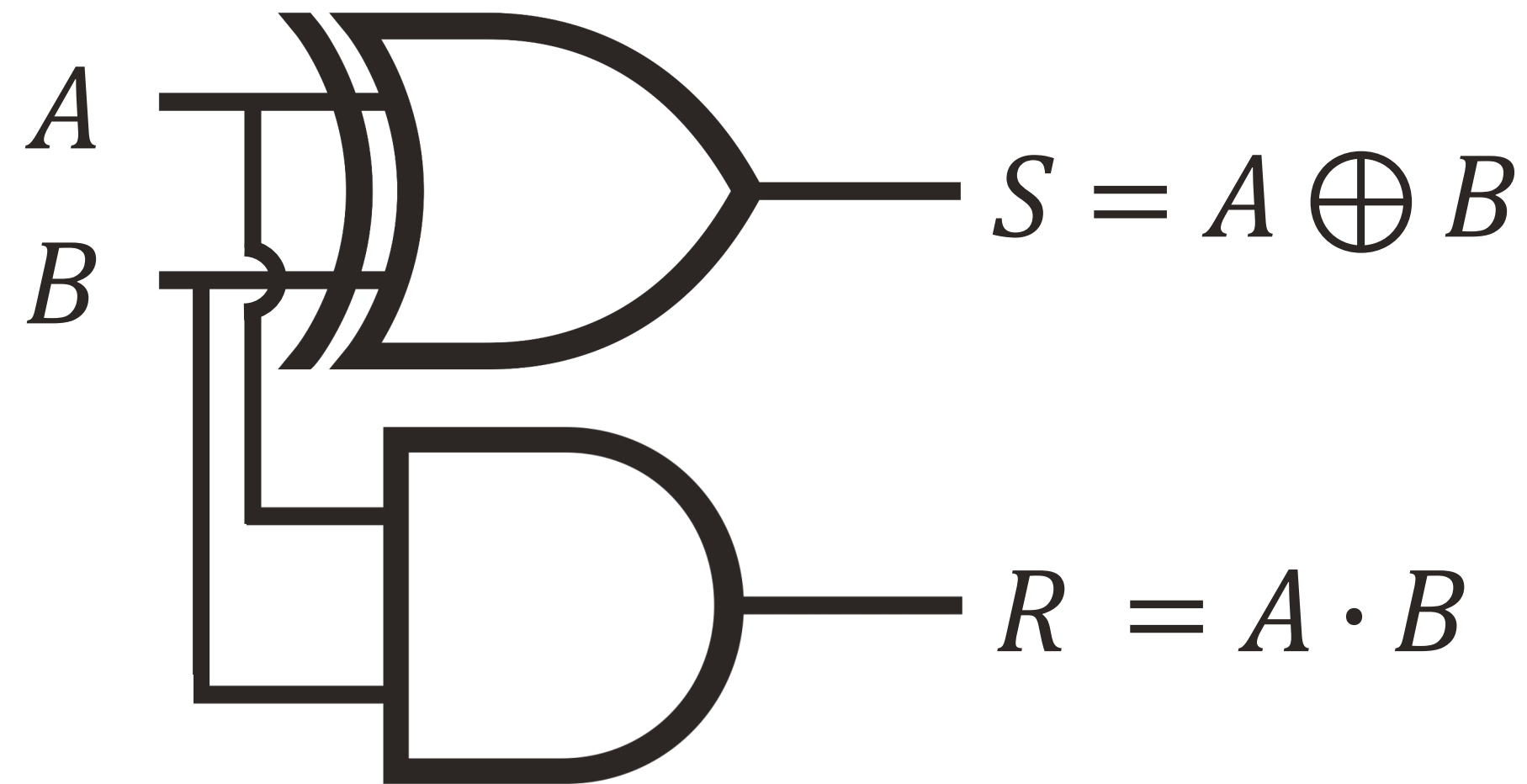
- Symbole résumant ce nouveau circuit :



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

EPFL Additionner deux bits (*avec retenue*)

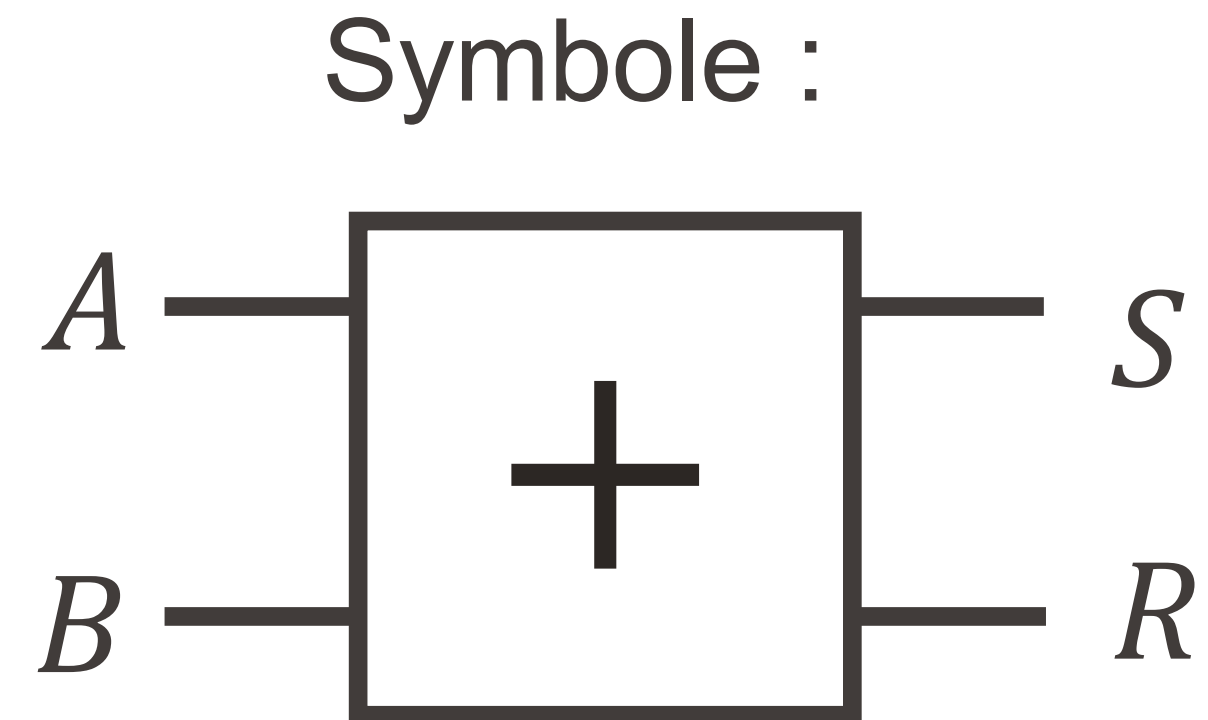
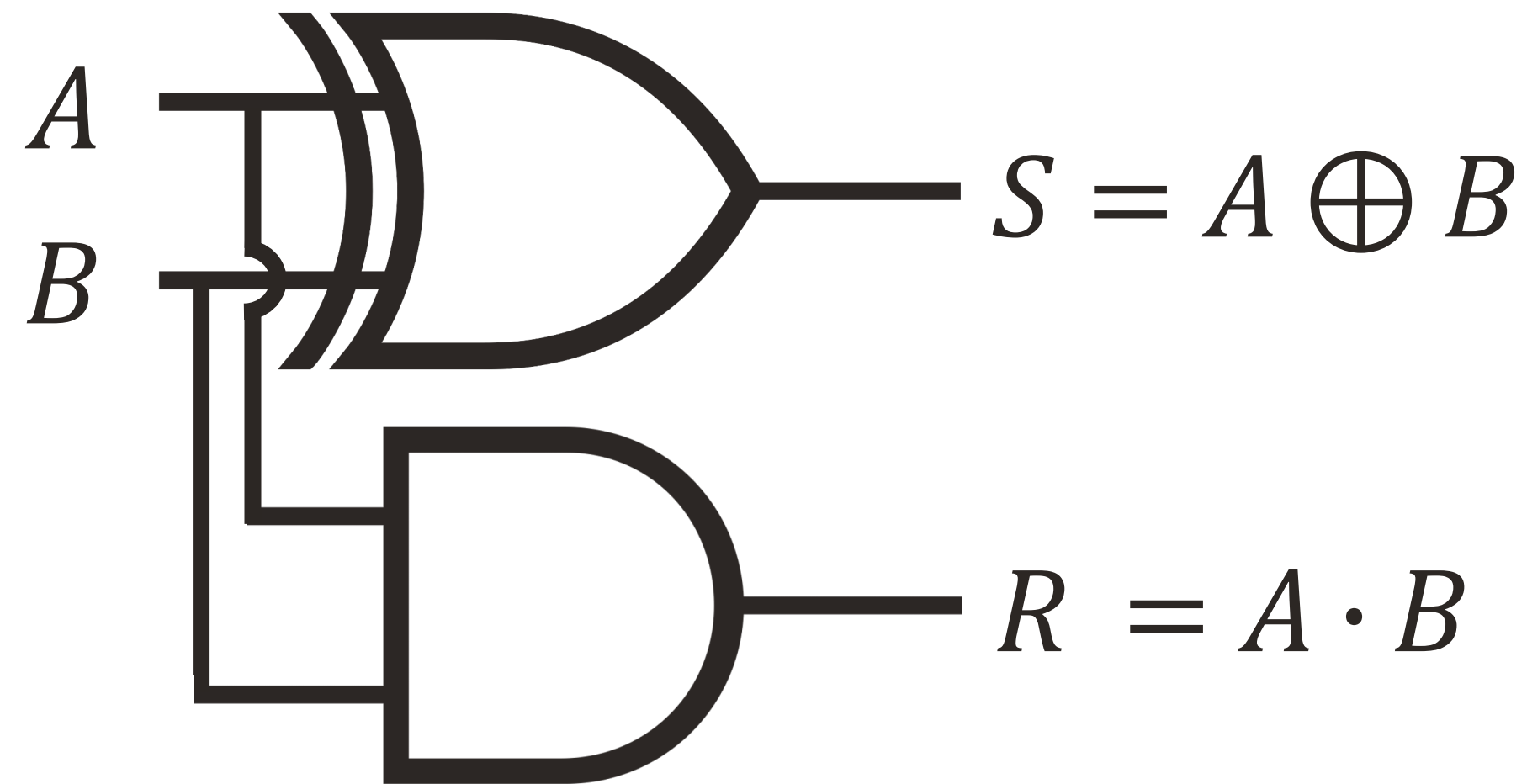
- On aimerait maintenant créer un circuit avec entrées A et B et sortie $S = A \oplus B$, ainsi qu'une retenue $R = 1$ si et seulement si $A = 1$ et $B = 1$.



A	B	R	S
0	0	0	0
0	1	0	1
1	0	0	1
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 $R = 1$ si et seulement si $A = 1$ et $B = 1$.



- 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 .

Ex: $0 + 1 + 0 = 01$ $1 + 0 + 1 = 10$ $1 + 1 + 1 = 11$

Notre but : additionner des nombres !

- Rappel

Avec des nombres entiers :

$$\begin{array}{r}
 11 \\
 57 \\
 + 43 \\
 \hline
 = 100
 \end{array}$$

Avec des bits :

$$\begin{array}{r}
 11111 \\
 111001 \\
 + 101011 \\
 \hline
 = 1100100
 \end{array}$$

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

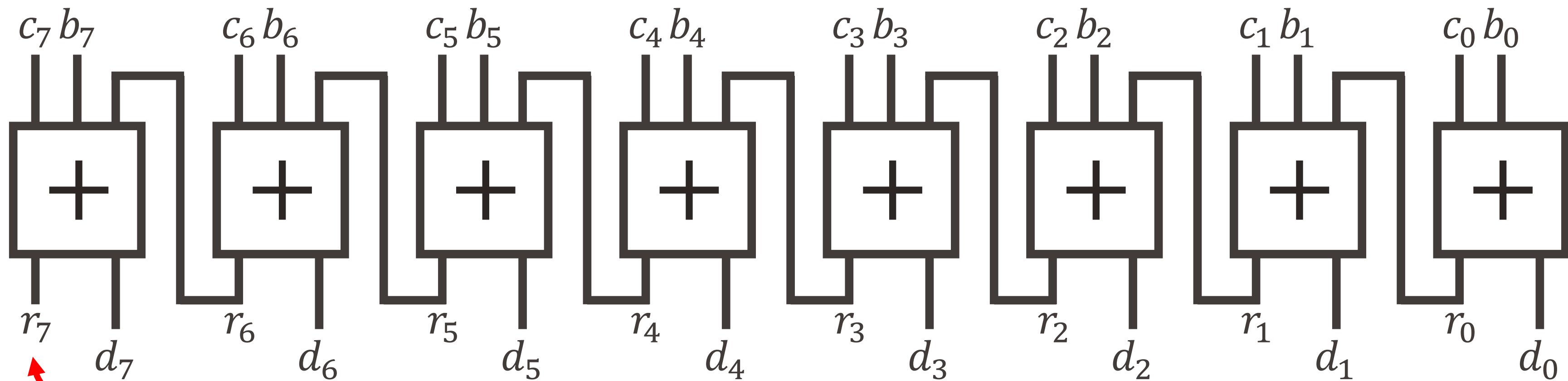
Additionneur sur 8 bits

Effectuer :

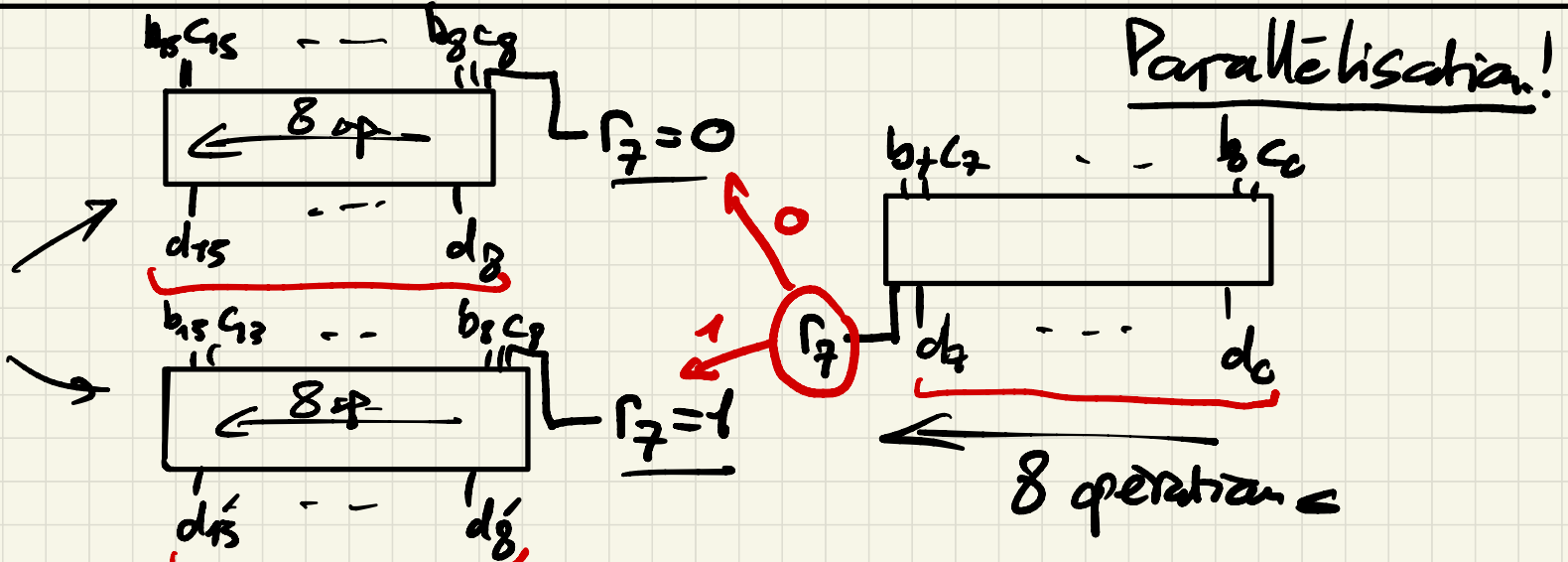
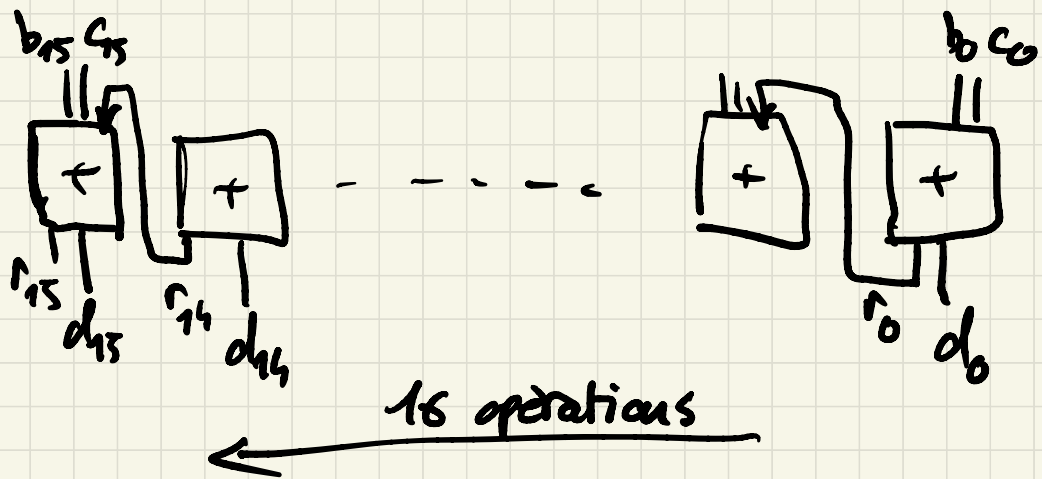
$$\begin{array}{r}
 b_7 b_6 b_5 \dots b_1 b_0 \\
 + c_7 c_6 c_5 \dots c_1 c_0 \\
 \hline
 = d_7 d_6 d_5 \dots d_1 d_0
 \end{array}$$

Exemple :

$$\begin{array}{r}
 111 \ 11 \\
 00111001 \\
 + 00101011 \\
 \hline
 = 01100100
 \end{array}$$



Si $r_7 = 1 \Rightarrow$ **overflow !**





**AUJOURD'HUI, ON PEUT METTRE DES
MILLIONS DE TRANSISTORS SUR UN CHIP !**

Information, Calcul et Communication

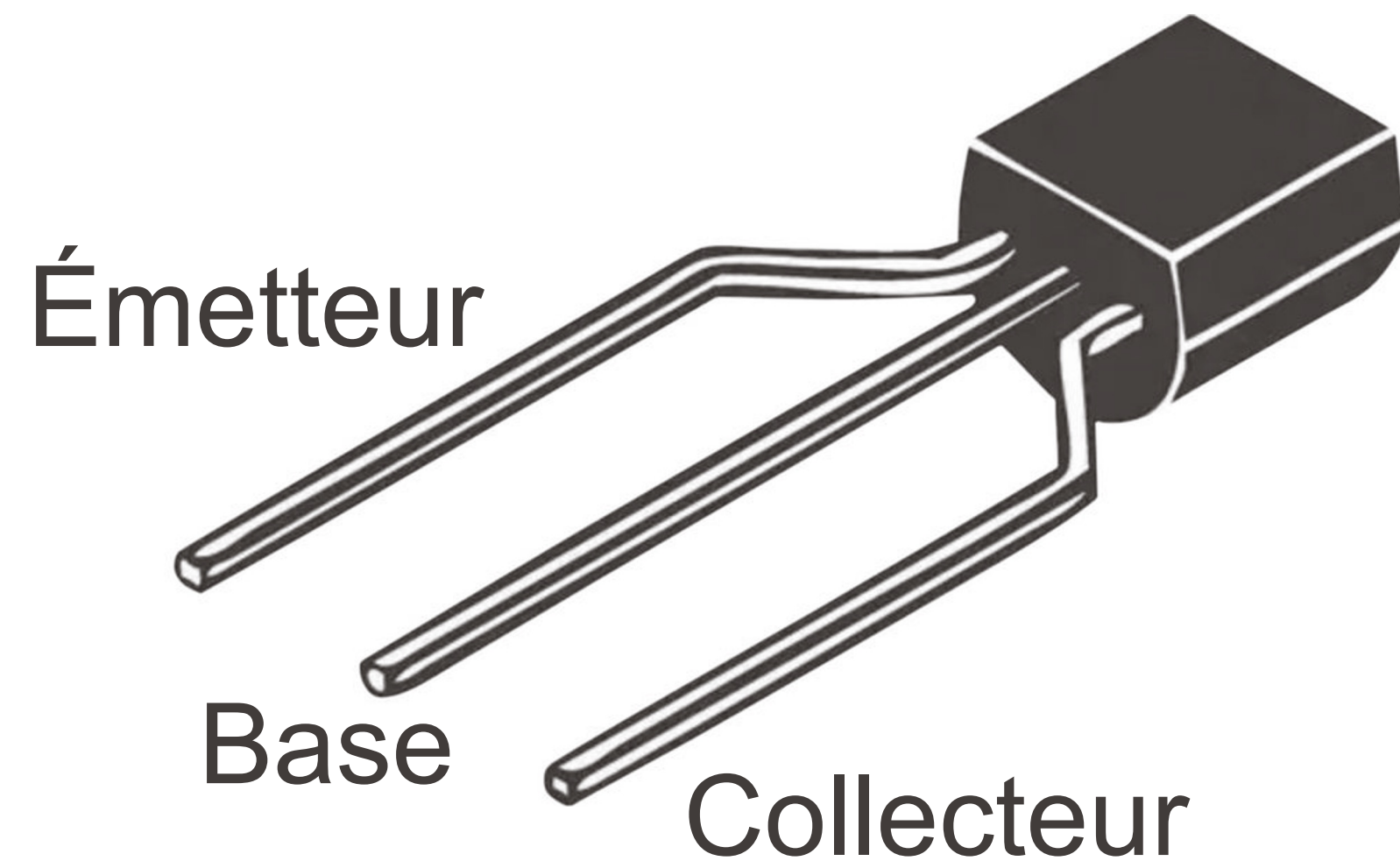
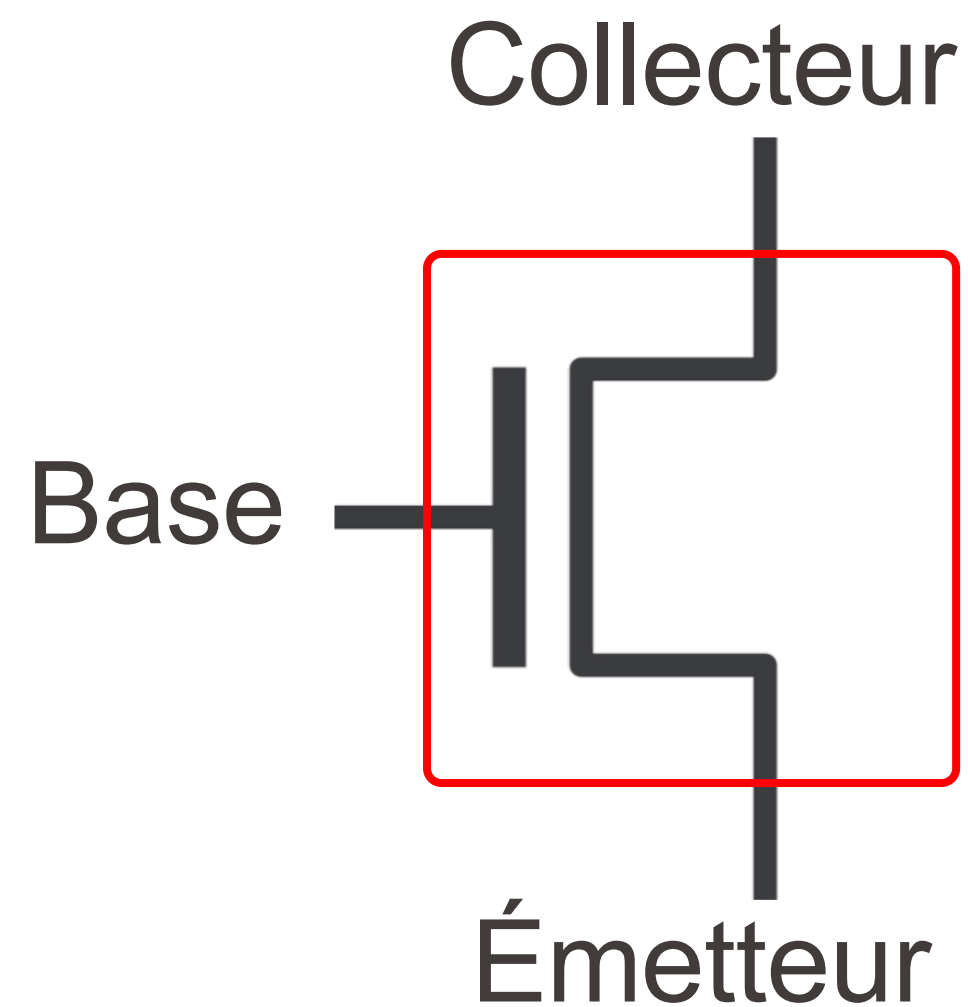
Transistors

Olivier Lévêque

Le transistor

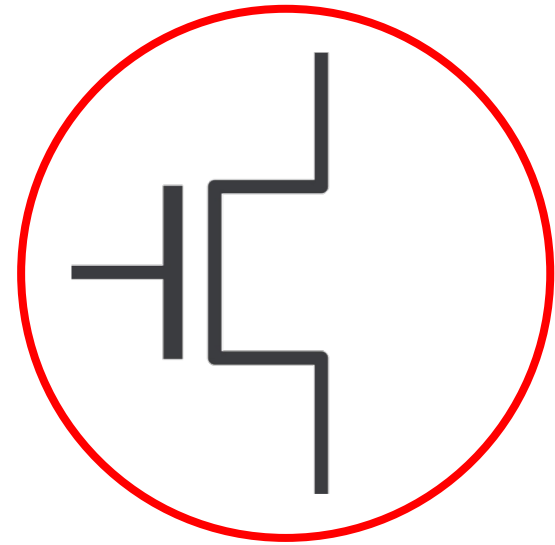
- Inventé en 1947 par trois américains : Bardeen, Shockley & Brattain
- Ce composant, qui est à la base de toute l'électronique moderne, a remplacé avantageusement les relais électromécaniques et les tubes à vide utilisés dans les premiers ordinateurs à la même époque → **miniaturisation**

- Schéma :

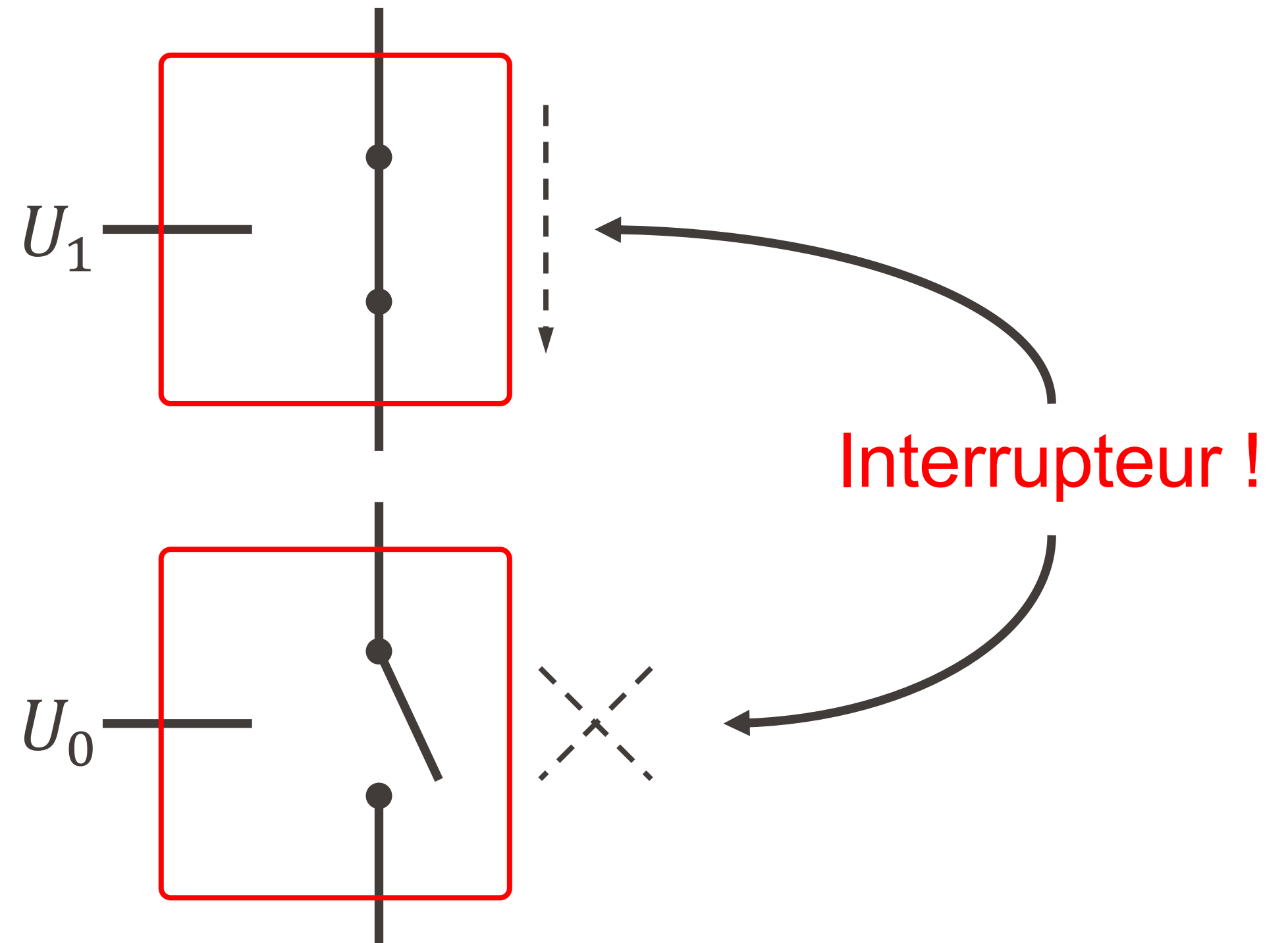


Principe de fonctionnement (*n-mos*)

- Symbole :

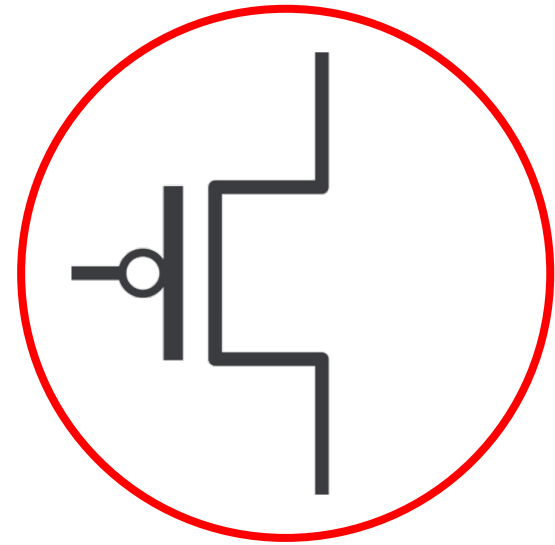


- Si la tension à la base est **haute** ($U_1 = 5V$) alors le courant passe entre l'émetteur et le collecteur :
- Si la tension à la base est **basse** ($U_0 = 0V$) alors le courant ne passe pas entre l'émetteur et le collecteur :

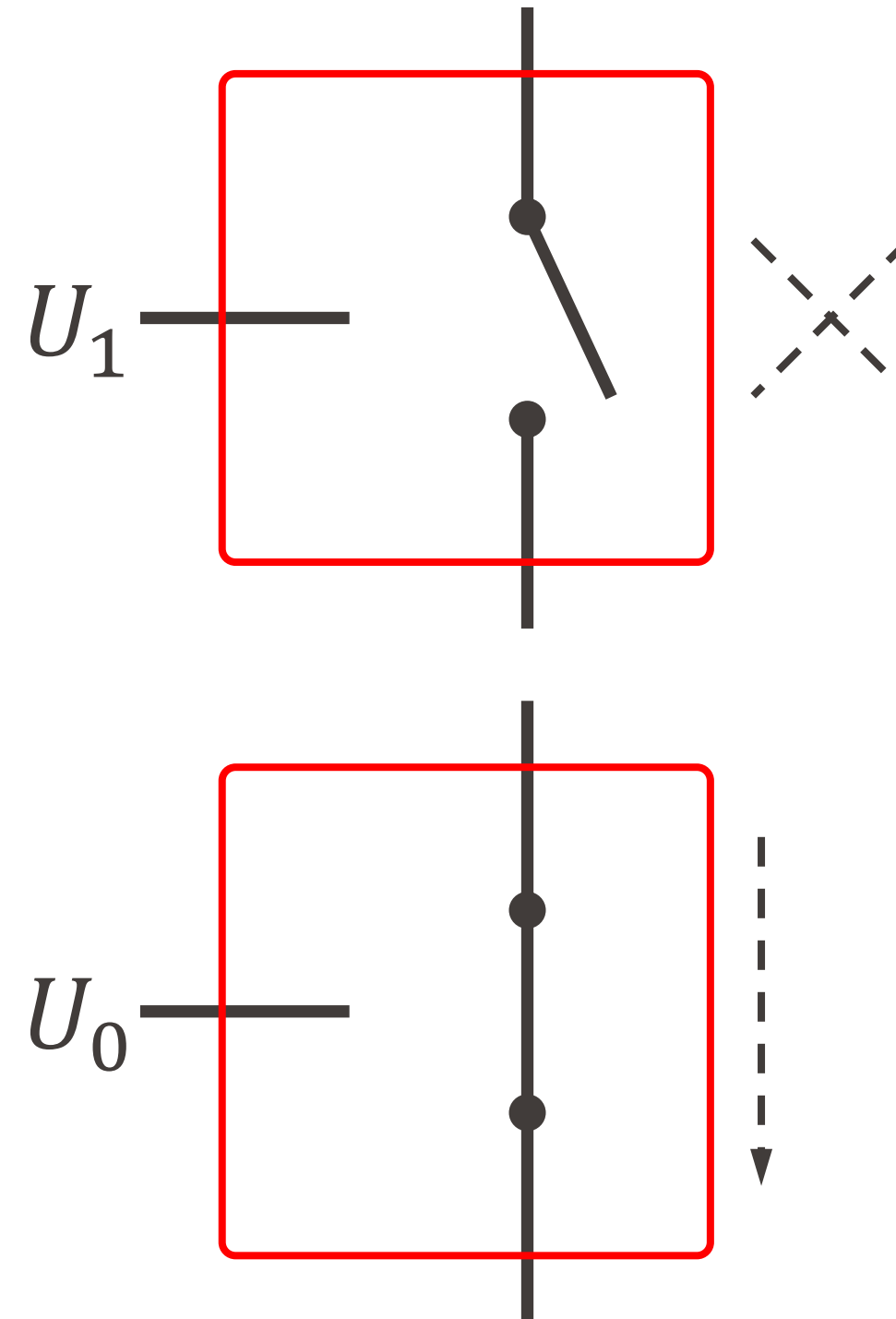


Principe de fonctionnement (*p-mos*)

- Symbole :

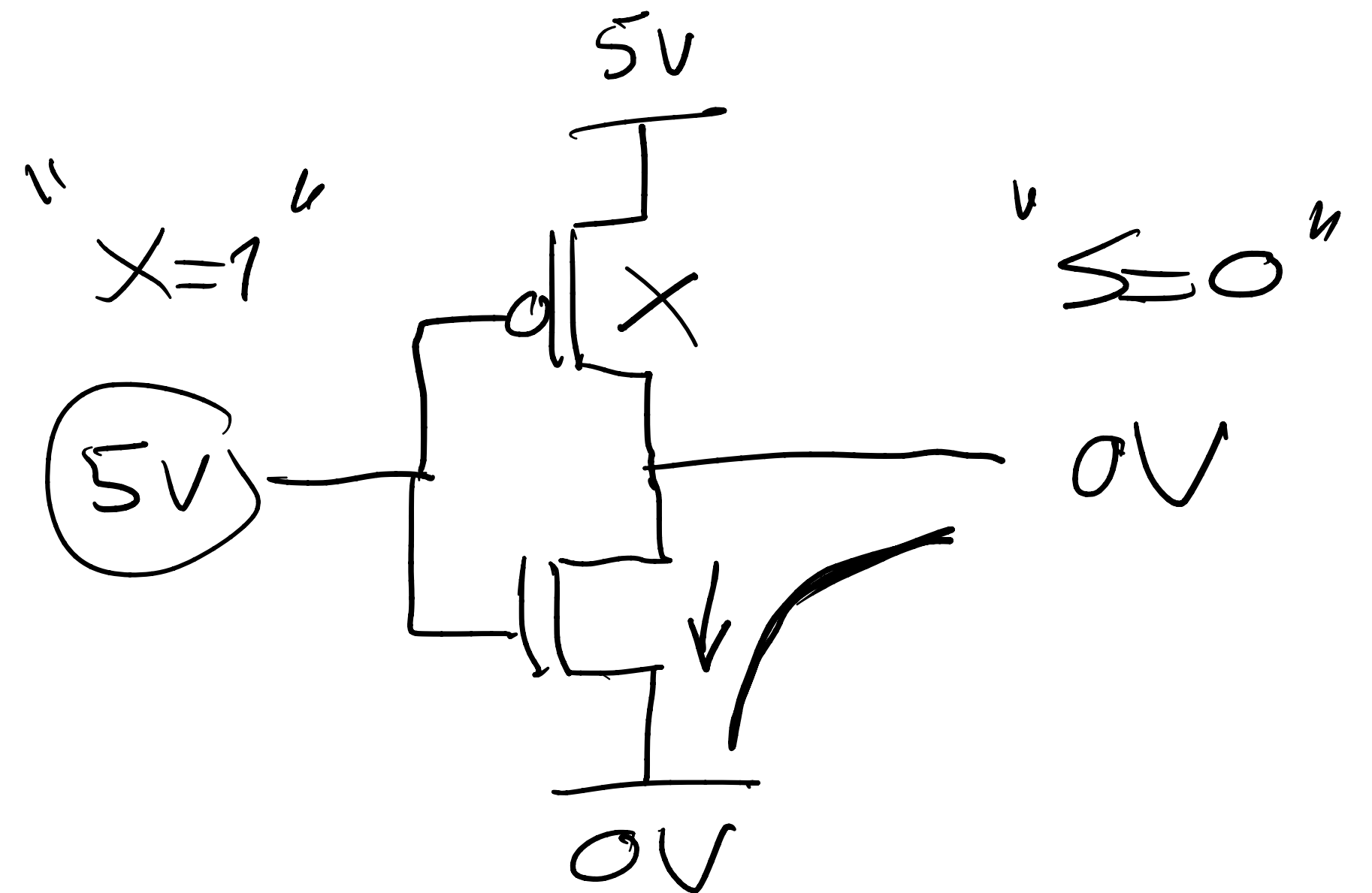
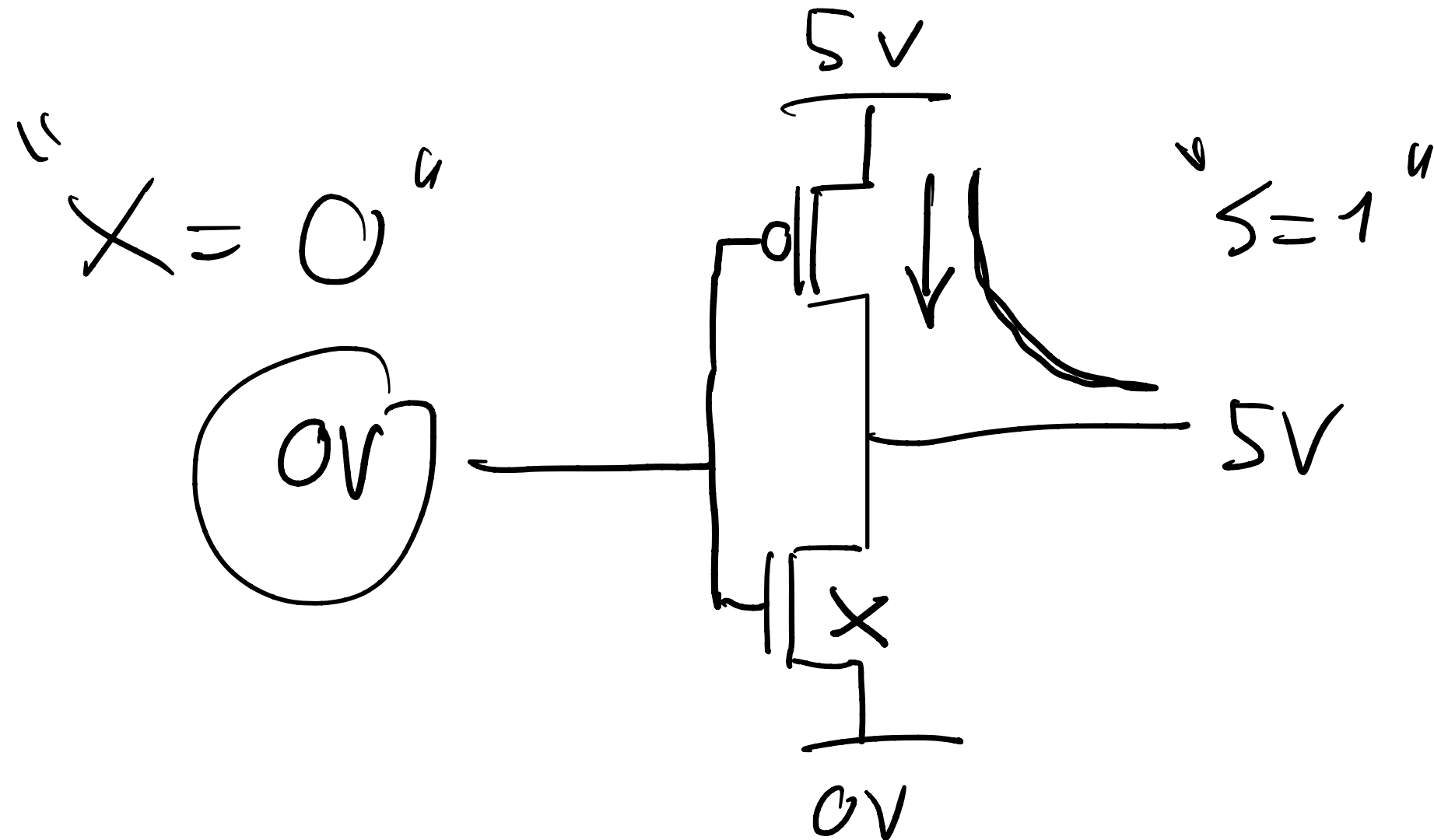
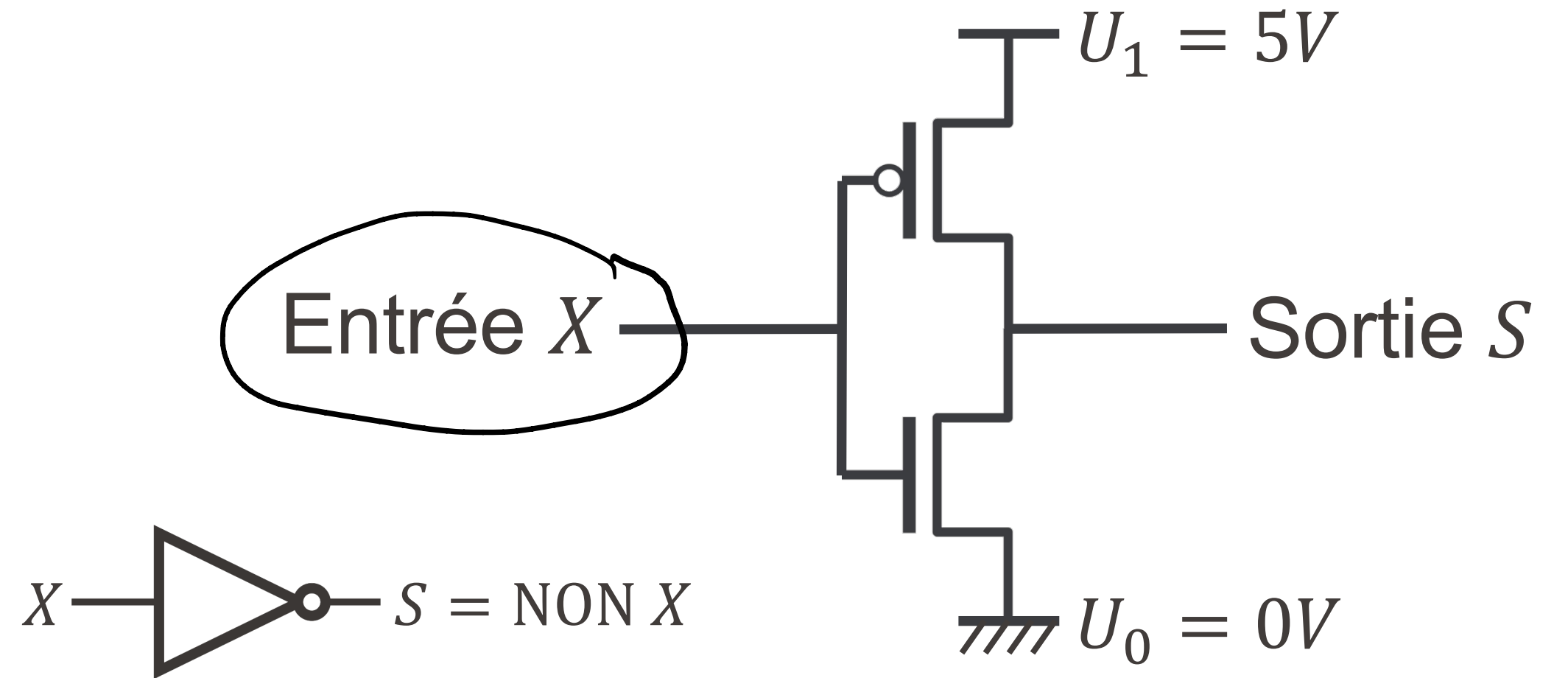


- Si la tension à la base est **haute** ($U_1 = 5V$) alors le courant ne passe pas entre l'émetteur et le collecteur :
- Si la tension à la base est **basse** ($U_0 = 0V$) alors le courant passe entre l'émetteur et le collecteur :



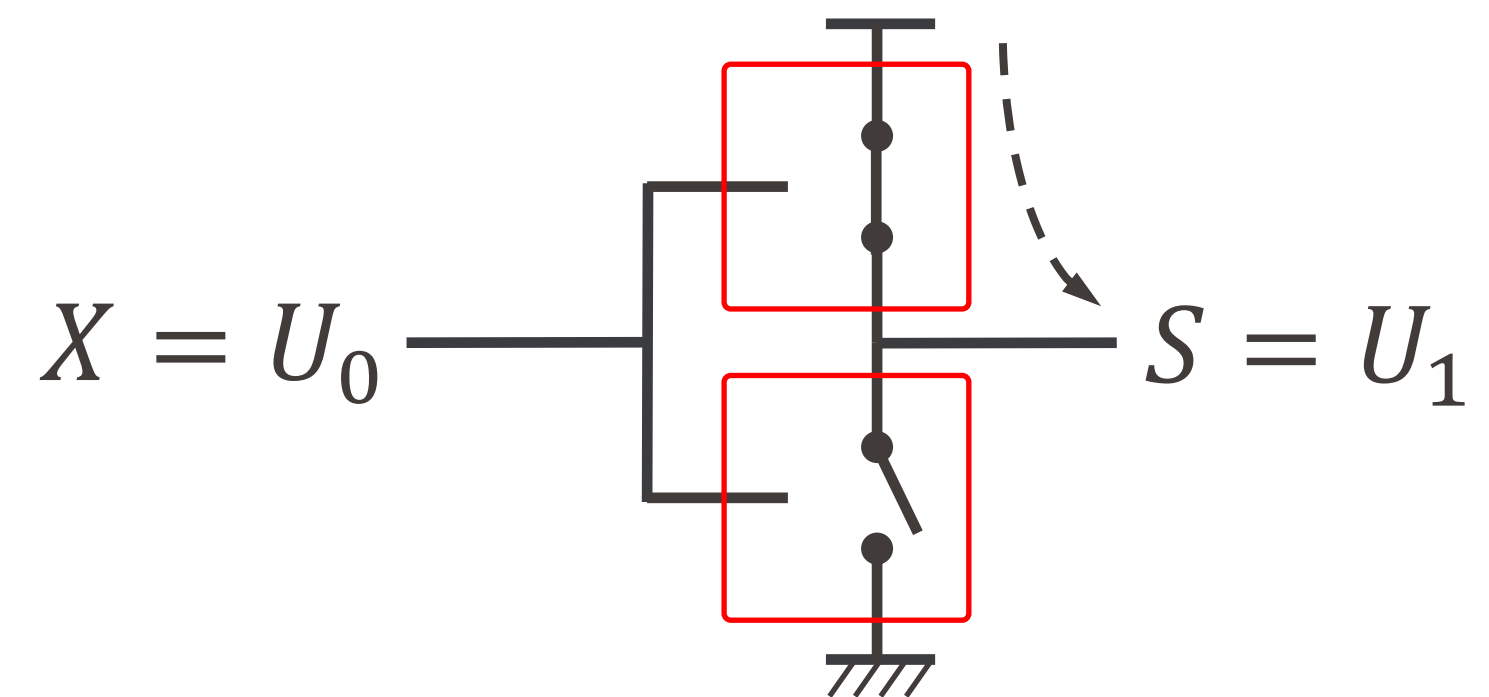
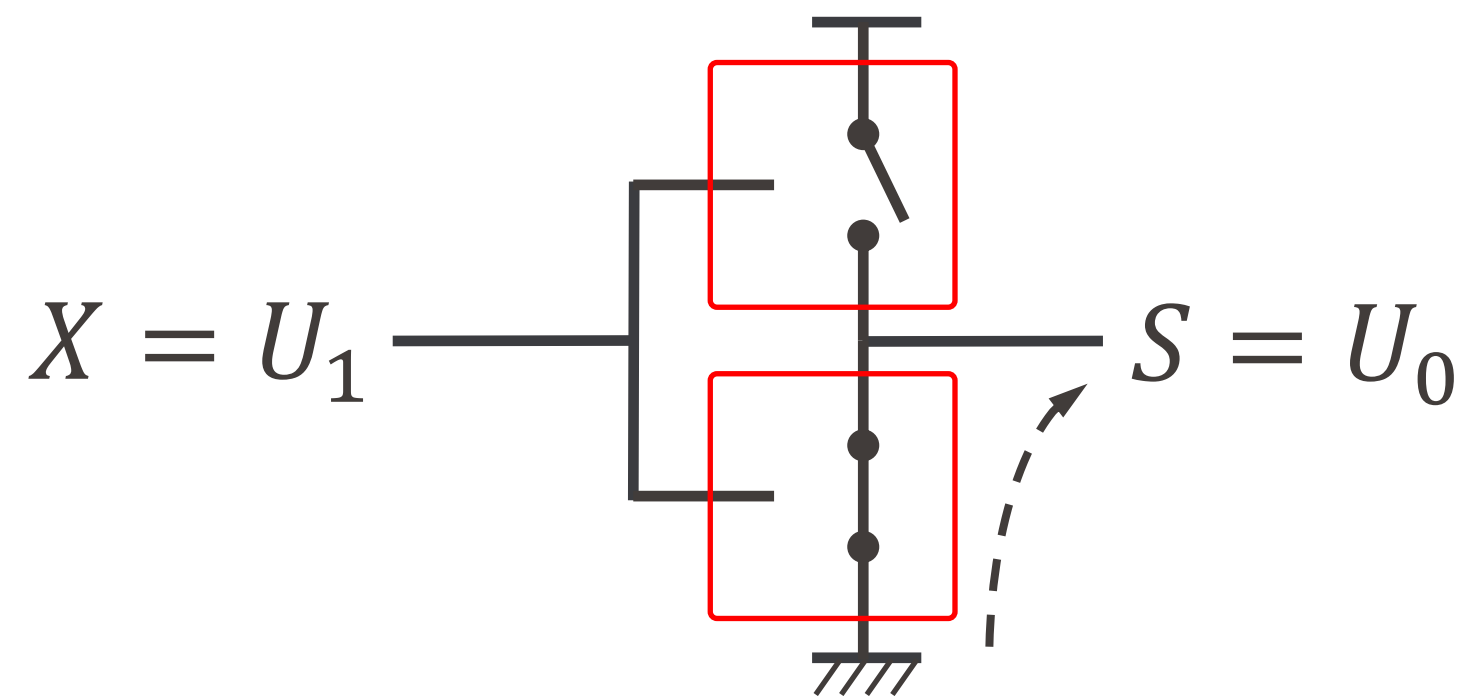
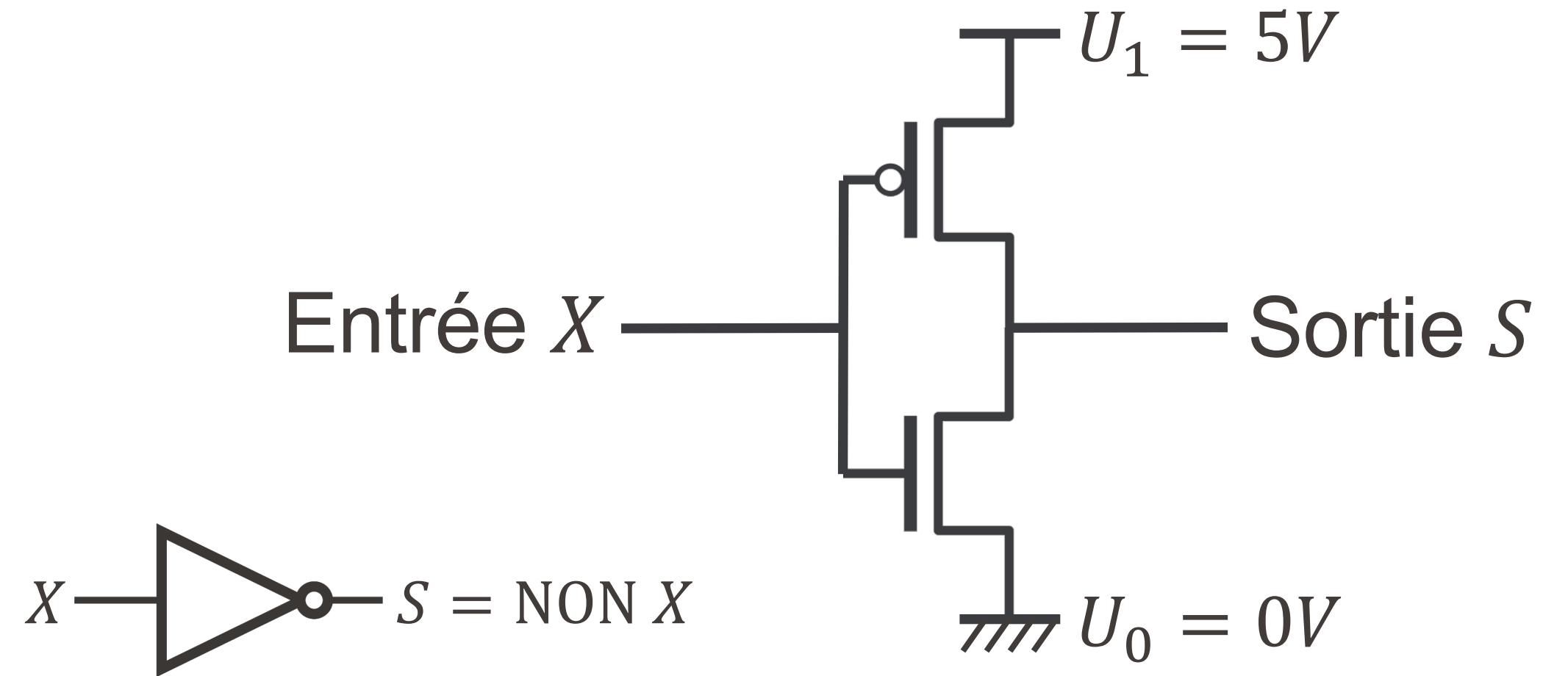
Création d'un inverseur

- 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



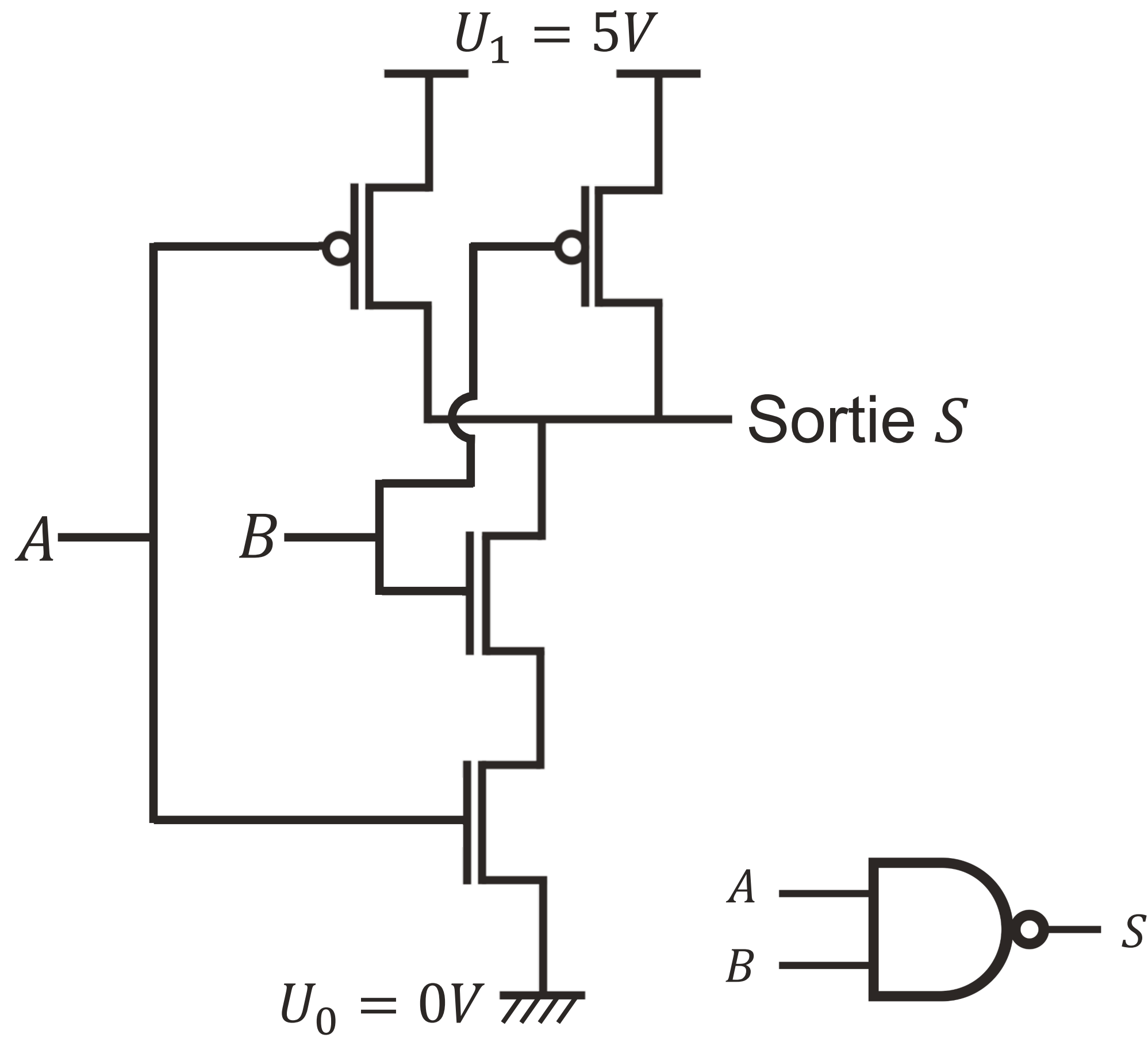
Création d'un inverseur

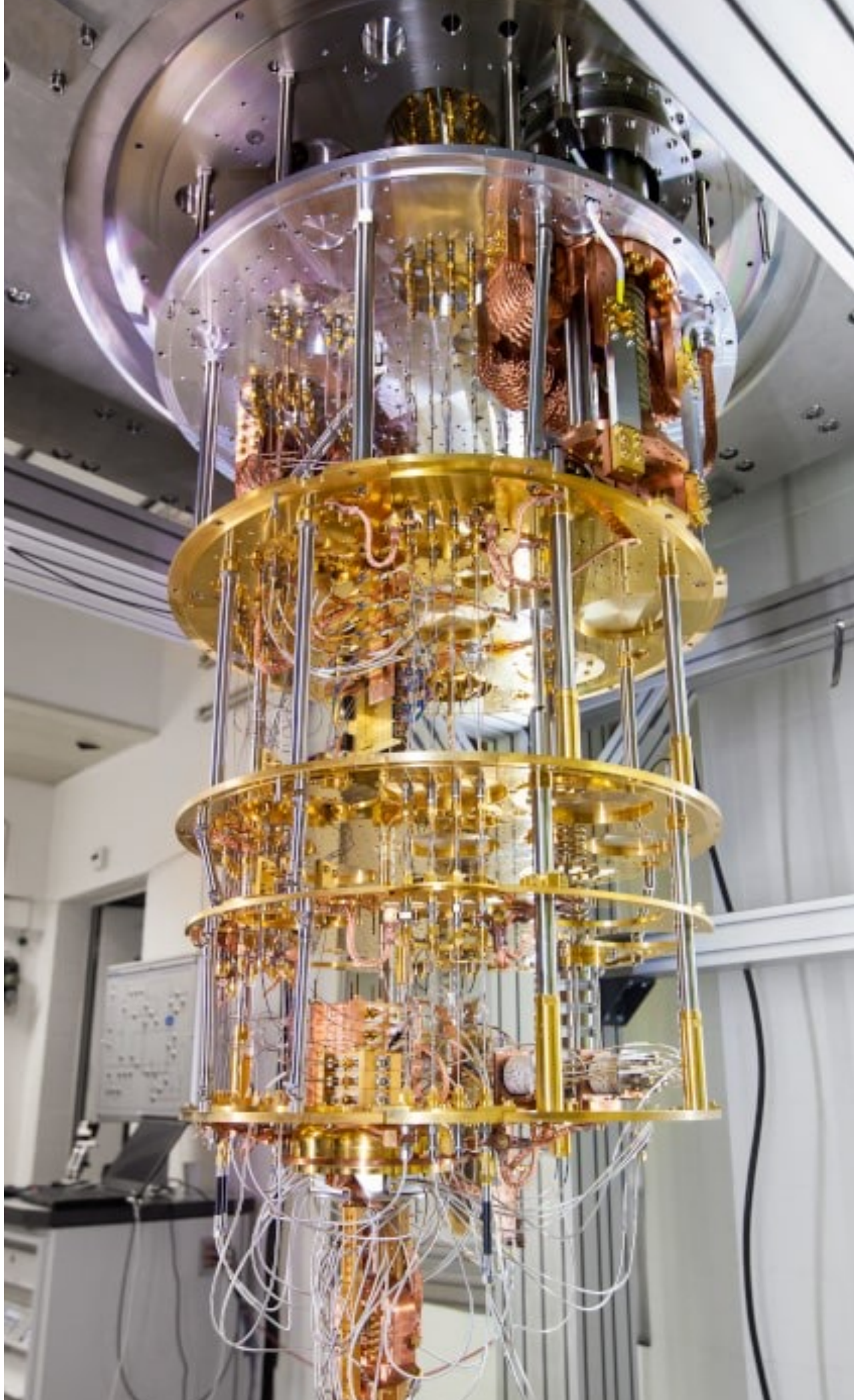
- Si on identifie U_0 comme 0 et U_1 comme 1, on peut créer un inverseur (*porte NOT*) à l'aide d'un transistor n-mos et d'un transistor p-mos



- 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





Information, Calcul et Communication

Circuits quantiques

Olivier Lévêque

- Durant les **années 80** germe l'idée qu'un ordinateur utilisant les propriétés quantiques de la matière au niveau microscopique pourrait obtenir des résultats de manière bien plus efficace qu'un ordinateur classique.
- **1992 : algorithme de Deutsch-Josza**, résolvant en une seule étape un problème qu'un ordinateur classique ne résolverait qu'en temps exponentiel.
- **1994 : algorithme de Shor**, permettant de factoriser de grands nombres en temps polynomial, alors que les meilleurs algorithmes classiques ont besoin d'un temps exponentiel.
- Depuis les **années 2000** : progression impressionnante dans la construction de différents ordinateurs quantiques pouvant traiter des données de plus en plus grande taille...

EPFL Algorithme de Deutsch

- Voici le problème à résoudre :

Etant donné une fonction $f: \{0,1\} \rightarrow \{0,1\}$,
on aimerait savoir si $f(1) = f(0)$ ou si $f(1) \neq f(0)$.

$$f_1(0) = f_1(1) = 0$$

$$f_2(0) = f_2(1) = 1$$



ou

$$f_3(0) = 0, f_3(1) = 1$$

$$f_4(0) = 1, f_4(1) = 0$$



EPFL Algorithme de Deutsch

- Voici le problème à résoudre :

Etant donné une fonction $f: \{0,1\} \rightarrow \{0,1\}$,
on aimerait savoir si $f(1) = f(0)$ ou si $f(1) \neq f(0)$.

- Classiquement, pour obtenir la réponse à cette question, il faut évaluer la fonction f à deux reprises, à savoir évaluer $f(0)$ et $f(1)$, et comparer les deux valeurs obtenues.
- Nous allons voir qu'avec un circuit quantique, une seule évaluation de la fonction f suffit !

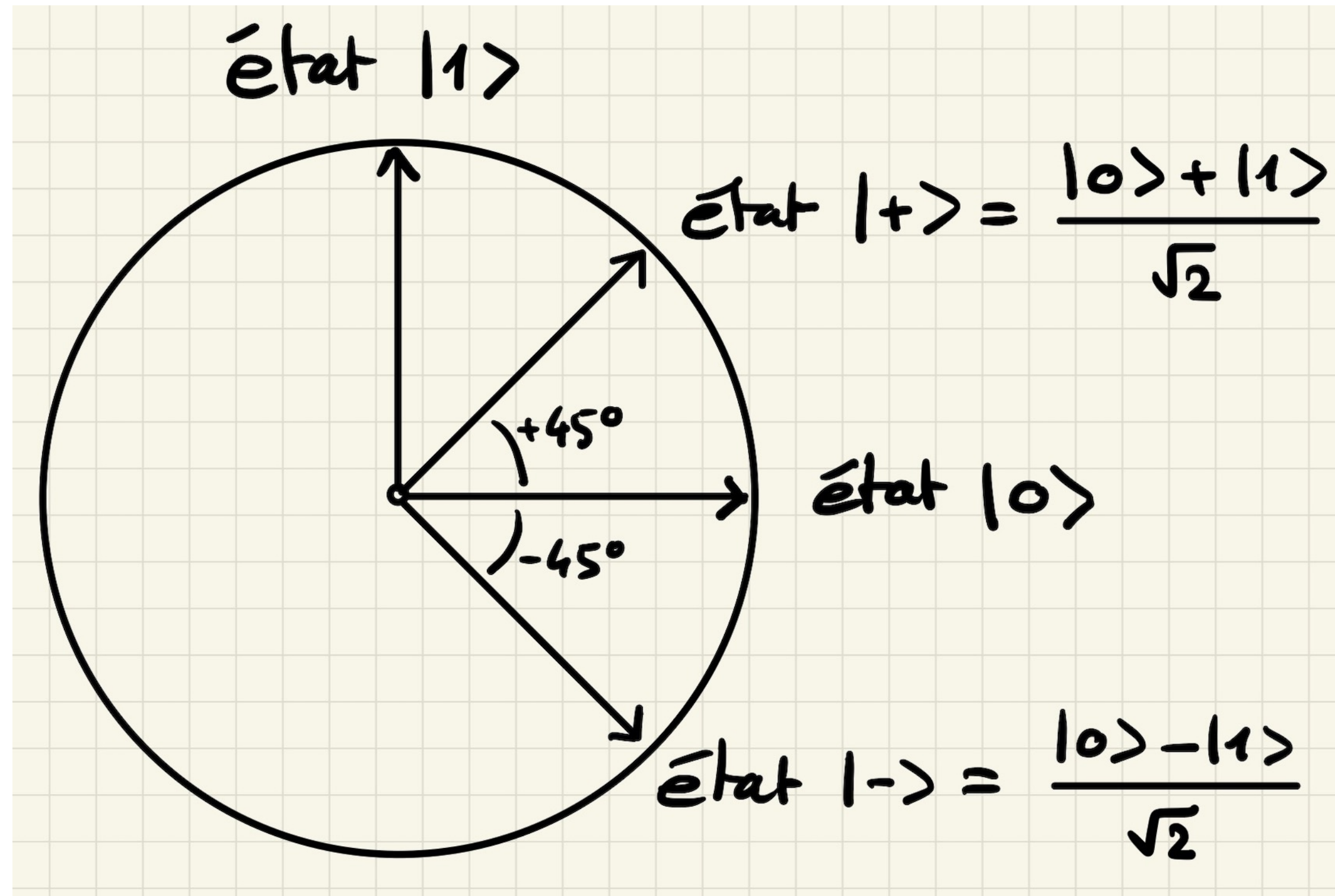
- En informatique quantique, les bits quantiques ou “qubits”, remplacent les bits classiques.
- L'état d'un qubit est noté $|\varphi\rangle$. Cet état peut valoir $|0\rangle$ ou $|1\rangle$, comme pour un bit classique, mais peut aussi être dans une **superposition d'états**, comme

$$|+\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \quad \text{ou} \quad |-\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

- C'est ce principe de superposition qui permet d'effectuer des calculs simultanés et donne ainsi un avantage aux ordinateurs quantiques.

Représentation vectorielle

Mathématiquement, l'état d'un qubit peut être représenté comme un vecteur unité en deux dimensions :



- La porte quantique H est une porte qui permet de créer des états superposés :

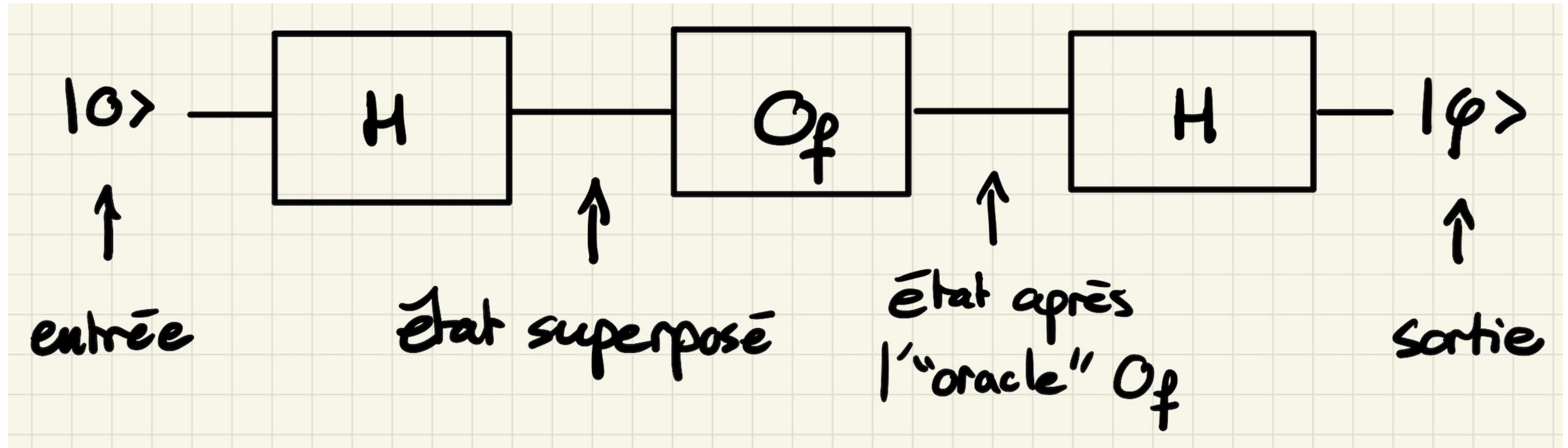
$$\begin{cases} |0\rangle \rightarrow H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \\ |1\rangle \rightarrow H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \end{cases}$$

- La porte quantique O_f (pour “oracle”) est une porte qui effectue la transformation :

$$O_f |+\rangle = \frac{1}{\sqrt{2}} ((-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle)$$

$\underbrace{\hspace{10em}}_{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}$

Circuit quantique de Deutsch



Note : Le fonctionnement de la porte "oracle" O_f est simplifié ici !

Circuit quantique de Deutsch

La sortie de ce circuit quantique vaut :

$$\begin{aligned}
 |\varphi\rangle &= H O_f H |0\rangle = H O_f \left(\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \right) \\
 &= H \left(\frac{1}{\sqrt{2}} \left((-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle \right) \right) \\
 &= \frac{1}{\sqrt{2}} (-1)^{f(0)} \underbrace{H |0\rangle}_{= |+\rangle} + \frac{1}{\sqrt{2}} (-1)^{f(1)} \underbrace{H |1\rangle}_{= |-\rangle} \\
 &= \frac{1}{\sqrt{2}} (-1)^{f(0)} (|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}} (-1)^{f(1)} (|0\rangle - |1\rangle) \\
 &= \frac{1}{2} (-1)^{f(0)} (|0\rangle + |1\rangle) + \frac{1}{2} (-1)^{f(1)} (|0\rangle - |1\rangle)
 \end{aligned}$$

Sortie du circuit :

$$|\varphi\rangle = \frac{(-1)^{f(0)} + (-1)^{f(1)}}{2} |0\rangle + \frac{(-1)^{f(0)} - (-1)^{f(1)}}{2} |1\rangle$$

- Si $f(1) = f(0)$, alors $|\varphi\rangle = (-1)^{f(0)} |0\rangle$.
- Si $f(1) \neq f(0)$, alors $|\varphi\rangle = (-1)^{f(0)} |1\rangle$.
- En mesurant la sortie du circuit, on obtient la réponse à la question posée ($f(1) = f(0)$ ou $f(1) \neq f(0)$?) en faisant appel **une seule fois** à l'oracle O_f (donc en effectuant une seule évaluation de la fonction f).

- Le circuit quantique de Deutsch permet de déterminer si $f(1) = f(0)$ ou au contraire si $f(1) \neq f(0)$ en une seule évaluation de la fonction f .
- Cet algorithme se généralise pour une fonction $f: \{0,1\}^n \rightarrow \{0,1\}$, et permet de déterminer en une seule évaluation de f si celle-ci est constante ou “balancée” (à savoir que f prend la valeur 0 la moitié du temps et la valeur 1 l’autre moitié).
- ... tandis qu’un algorithme classique nécessite dans le pire des cas de l’ordre de 2^n évaluations de la fonction f pour répondre à cette question !
- C’est le **parallélisme quantique** (superposition d’états) qui permet ceci !