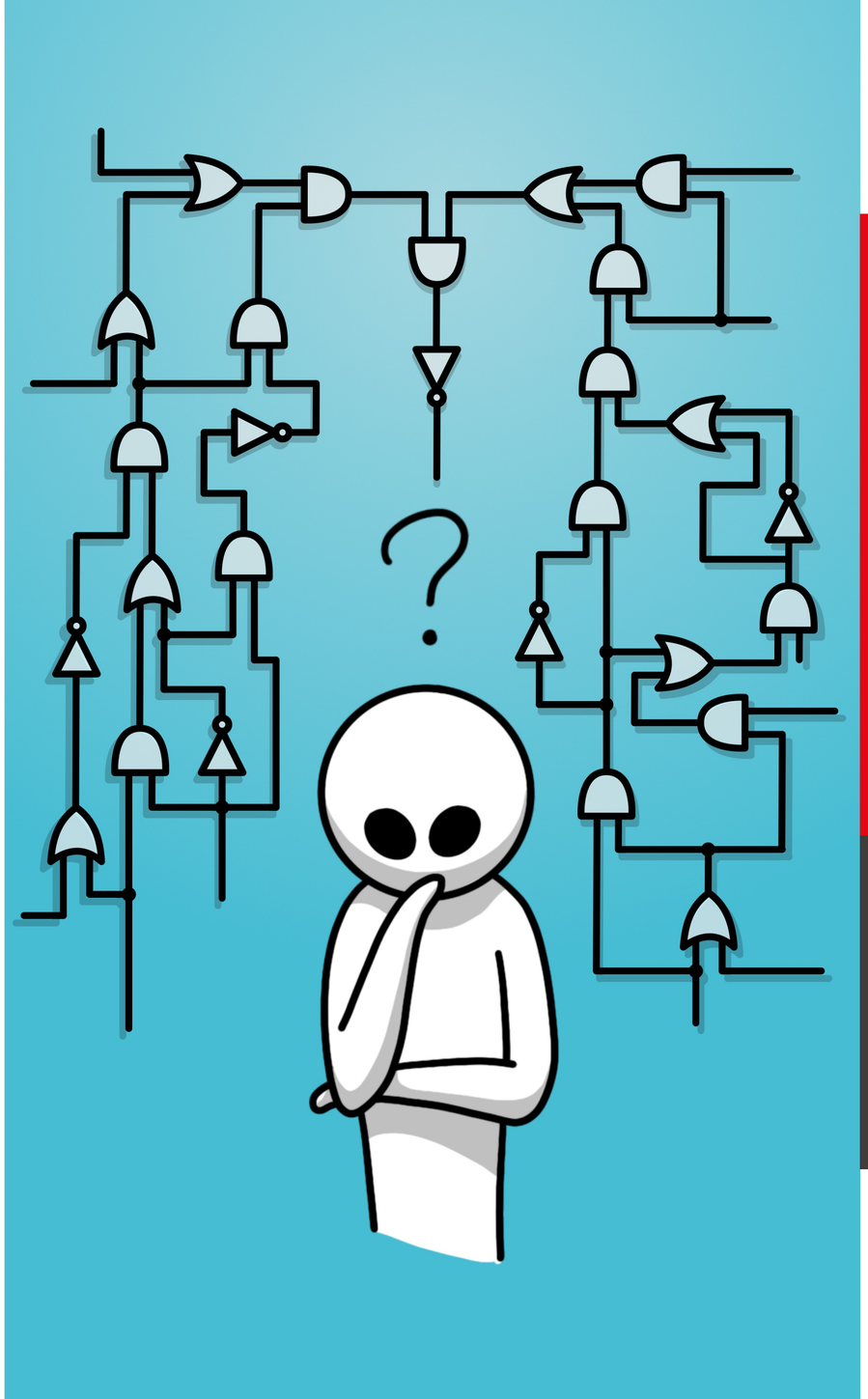


Information, Calcul et Communication

Circuits logiques

Olivier Lévêque

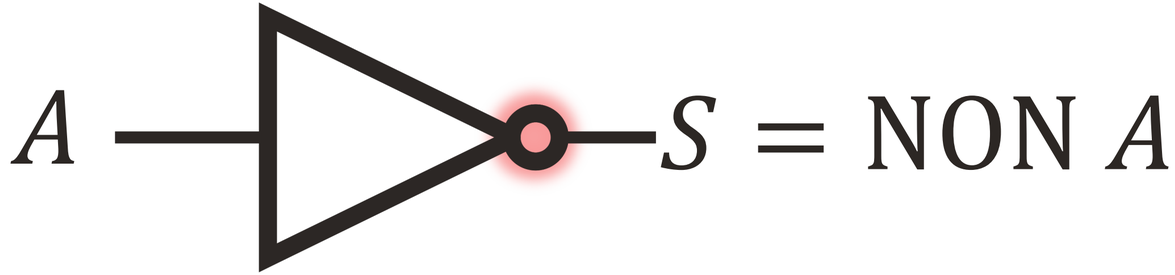
École polytechnique fédérale de Lausanne

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



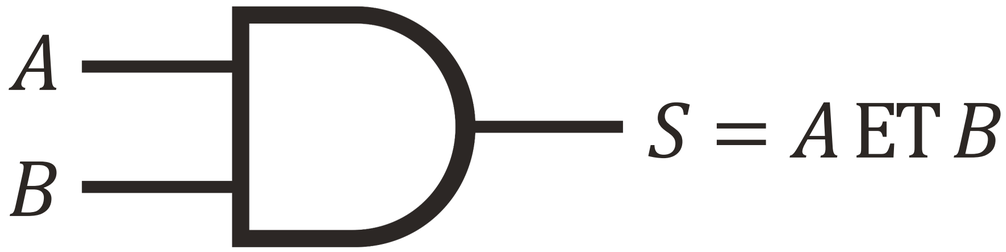
- La porte NON donne en sortie, **l'inverse** de la valeur du bit d'entrée

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

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

La porte ET *(AND)*

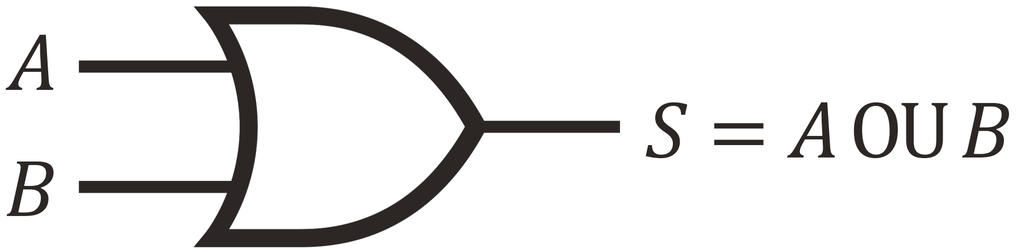
- Elle comporte deux ou plusieurs entrées.
- La porte ET génère un 1 en sortie si et seulement si les deux bits en entrée sont égaux à 1. Dans le cas contraire, la sortie vaut 0.
- Notez que la valeur de la sortie S correspond au produit des valeurs d'entrées $A \cdot B$.



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

La porte OU *(OR)*

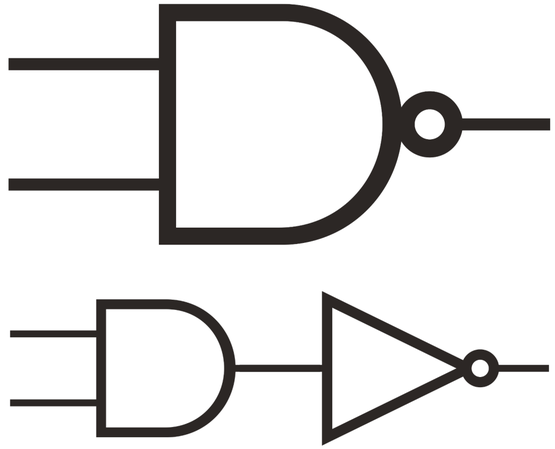
- Elle comporte deux ou plusieurs entrées.
- La porte OU génère un 1 en sortie si au moins un des bits en entrée vaut 1. La sortie vaut donc 0 en sortie si et seulement si les deux bits en entrée valent 0.
- Notez que la valeur de sortie S vaut 1 quand $A + B \geq 1$ (mais n'est donc **pas** égale à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



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

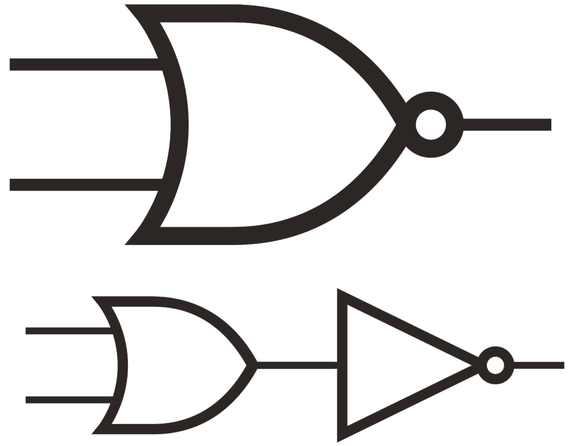
Les portes NON ET *(NAND)* et NON OU *(NOR)*

- 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

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 :

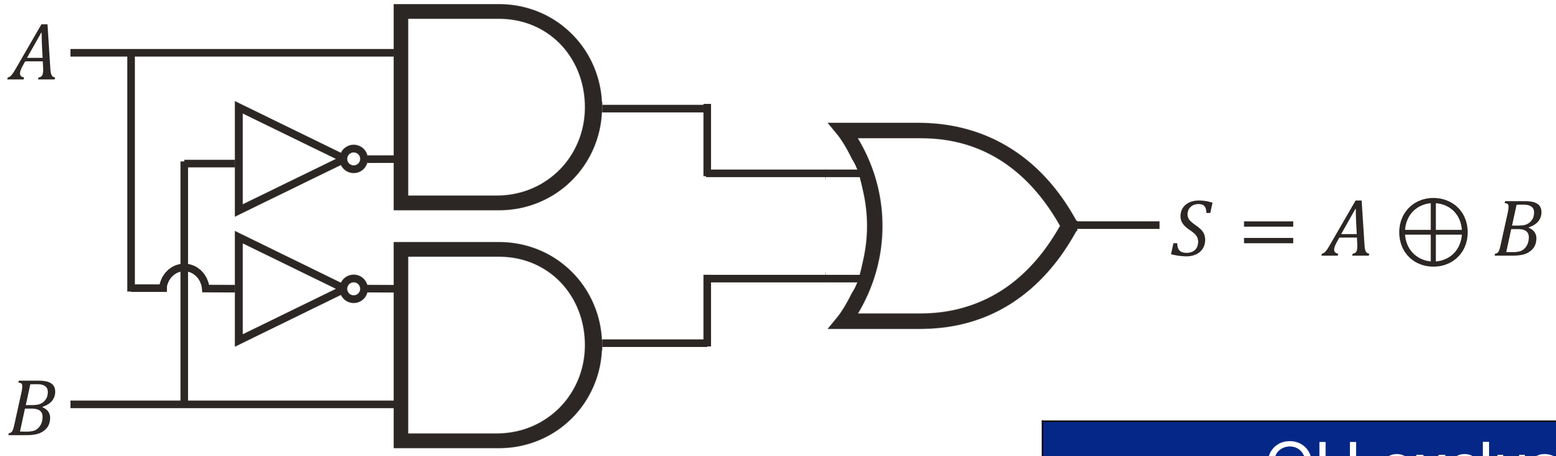
$$(A = 1 \text{ ET } B = 0) \text{ OU } (A = 0 \text{ ET } B = 1)$$

- Autrement dit:

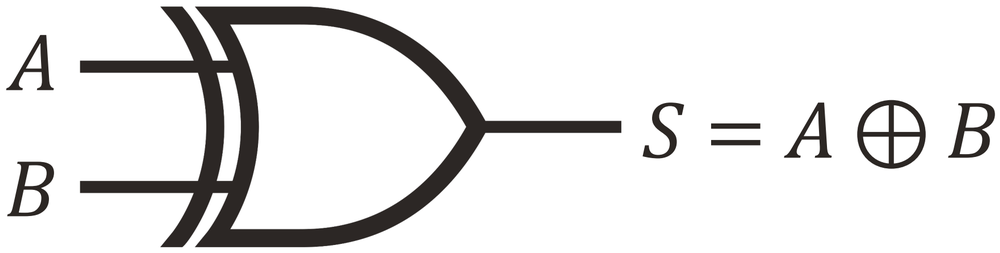
$$S = (A \text{ ET NON } B) \text{ OU } (\text{NON } A \text{ ET } B)$$

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

- Circuit correspondant :



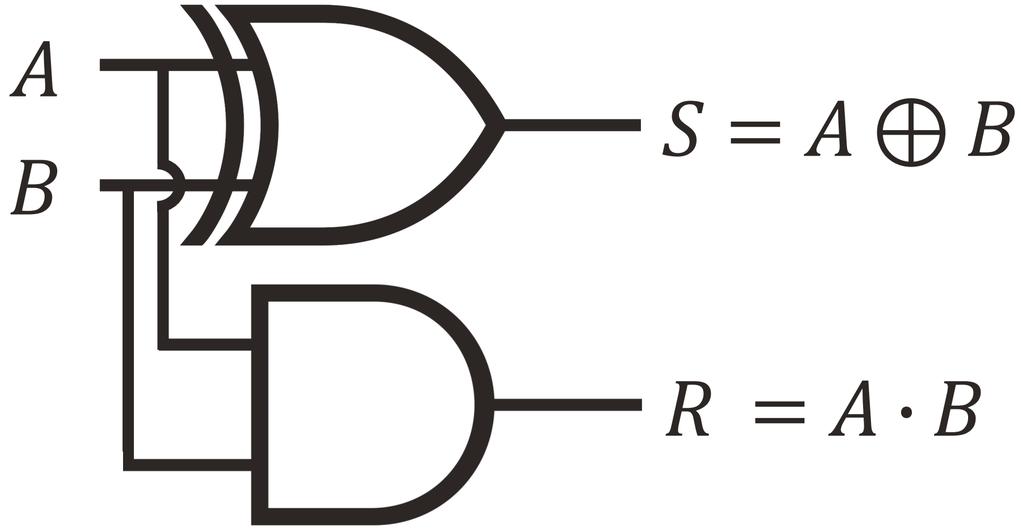
- Symbole résumant ce nouveau circuit :



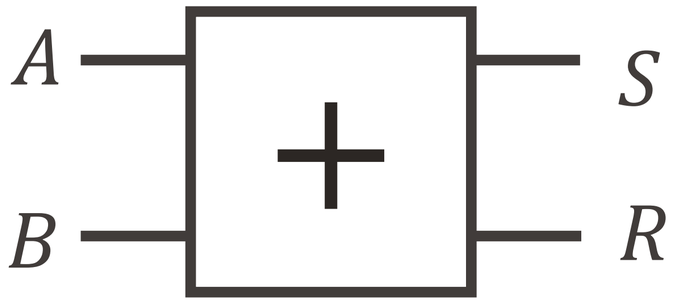
OU exclusif		
A	B	$S = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

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.

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 :

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

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

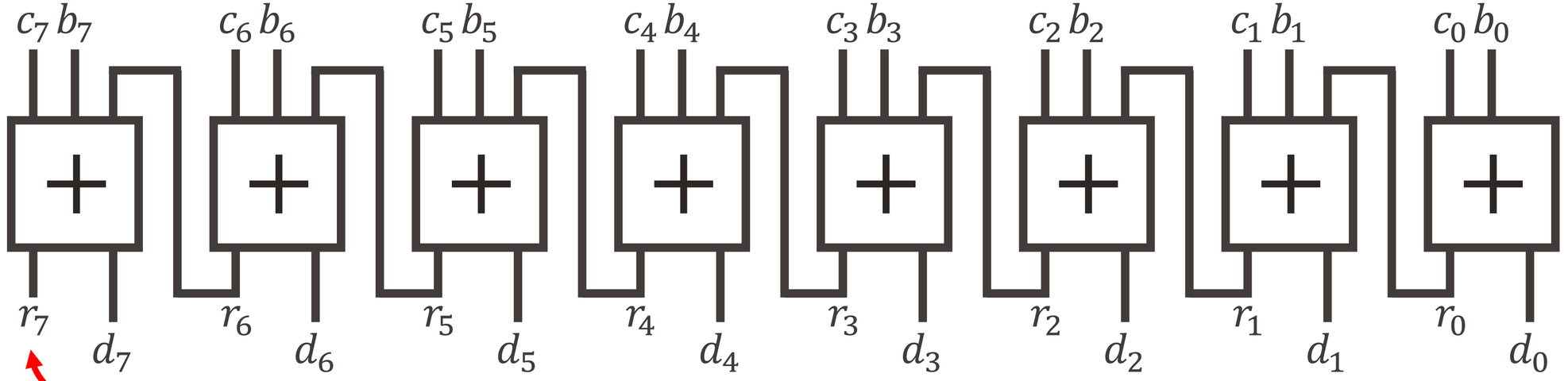
Additionneur sur 8 bits

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 :

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



Si $r_7 = 1 \Rightarrow$ overflow !

Information, Calcul et Communication

Transistors

École polytechnique fédérale de Lausanne

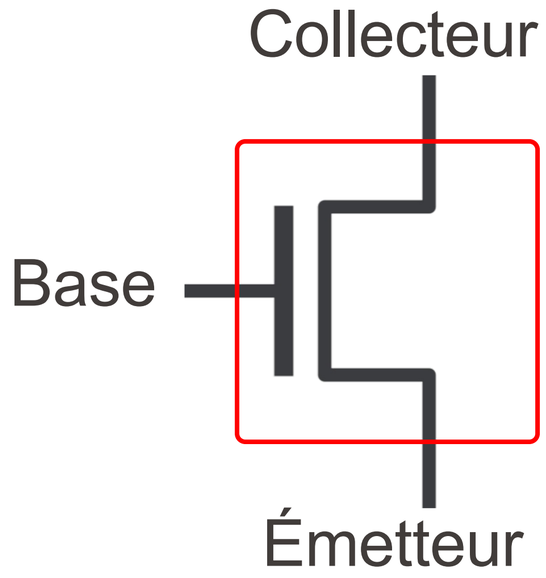
Olivier Lévêque

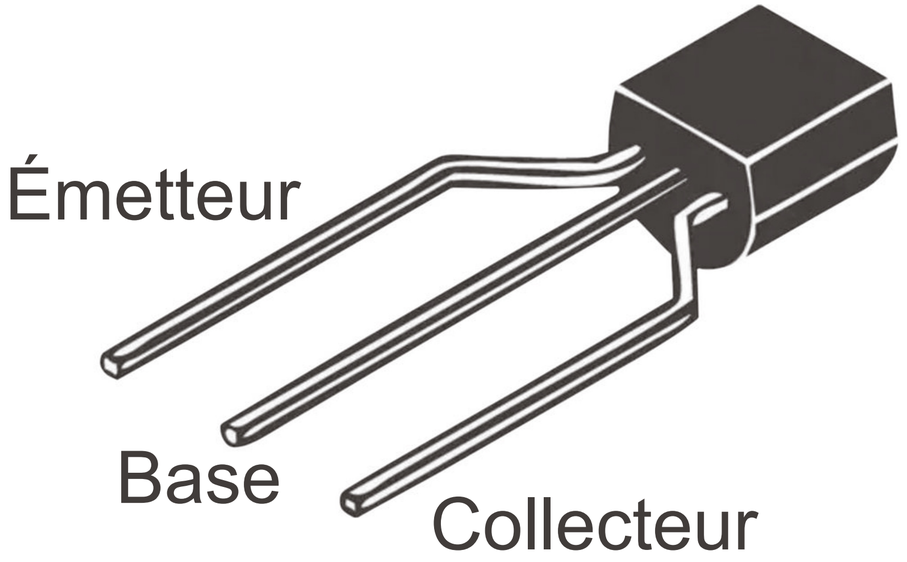
Le transistor

- 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***

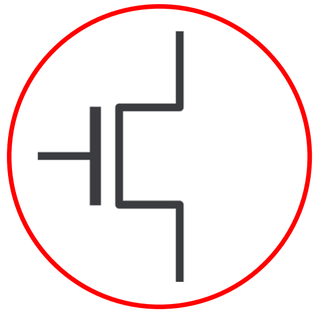
- Schéma :





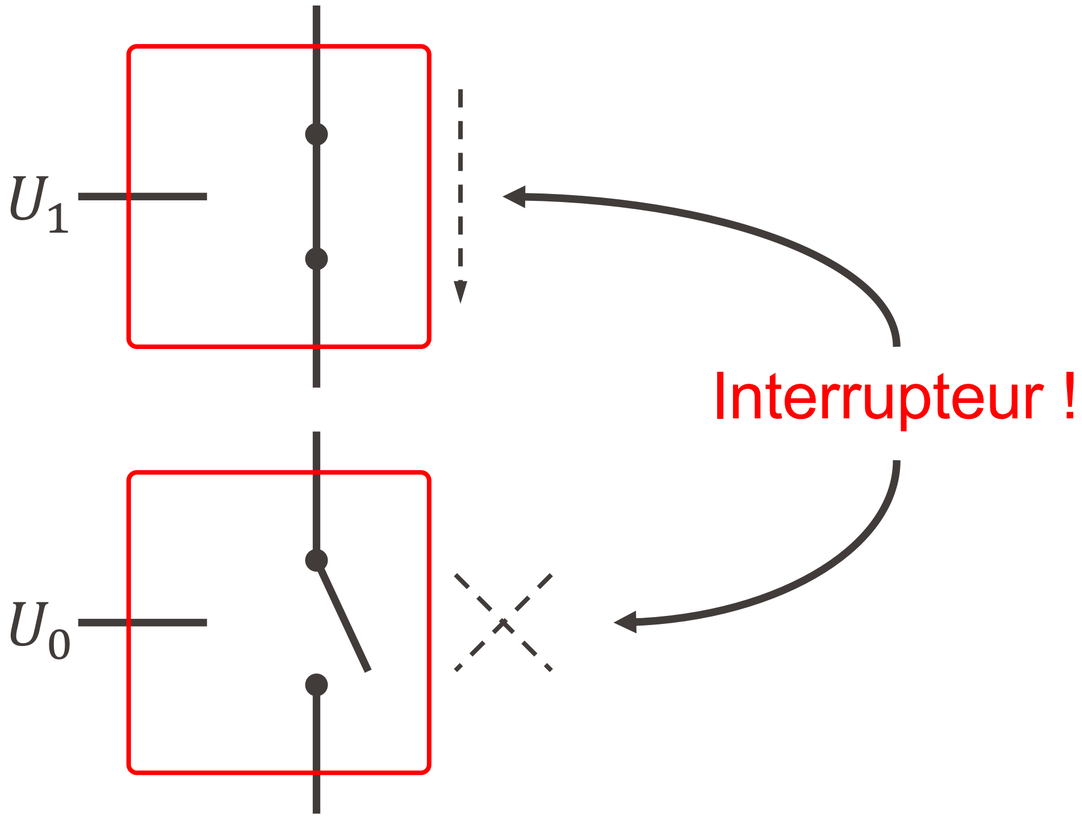
Principe de fonctionnement *(n-mos)*

- Symbole :



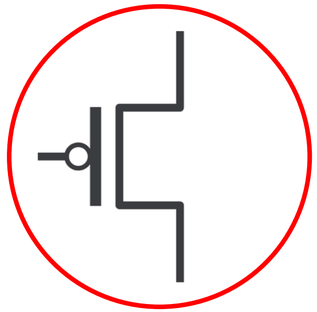
- Si la tension à la base est ***haute*** ($U_1 = 5V$) alors le courant passe entre l'émetteur et le collecteur :

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

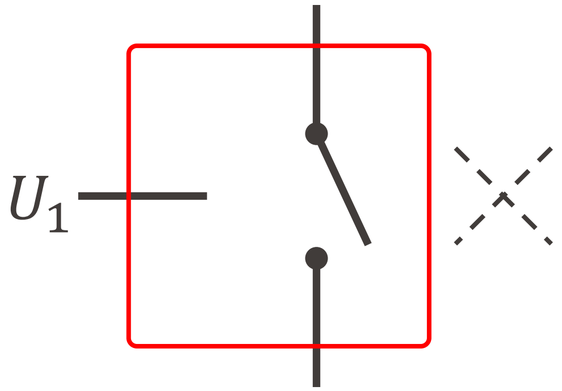


Principe de fonctionnement *(p-mos)*

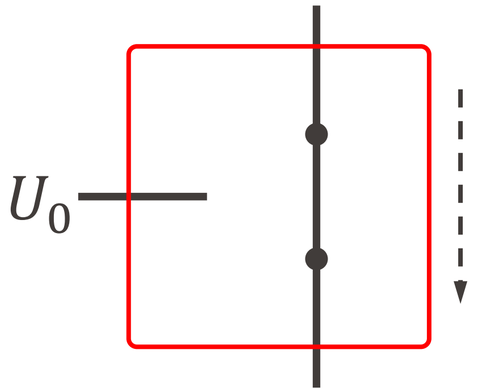
- Symbole :



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

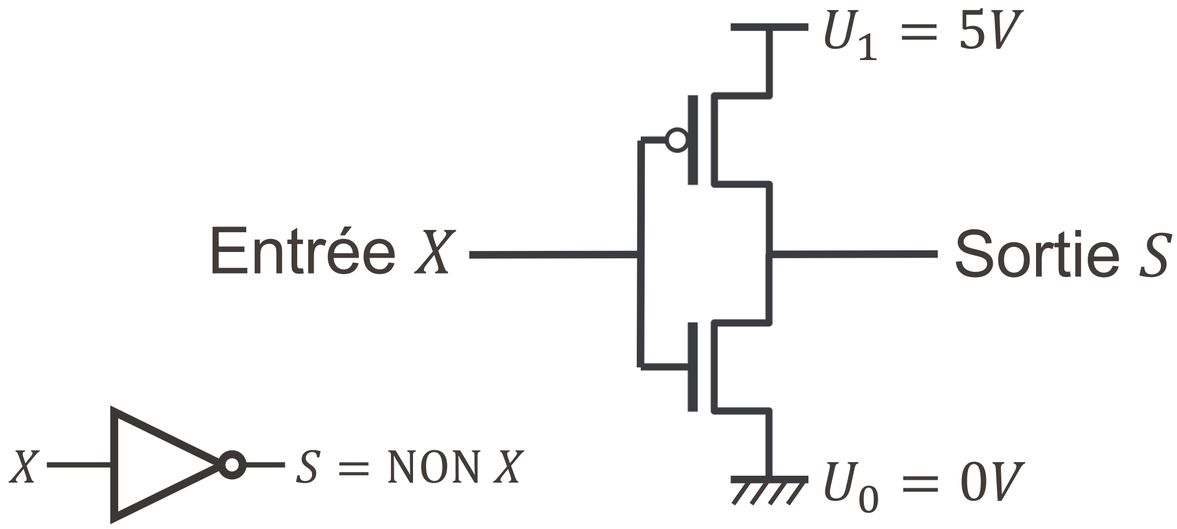


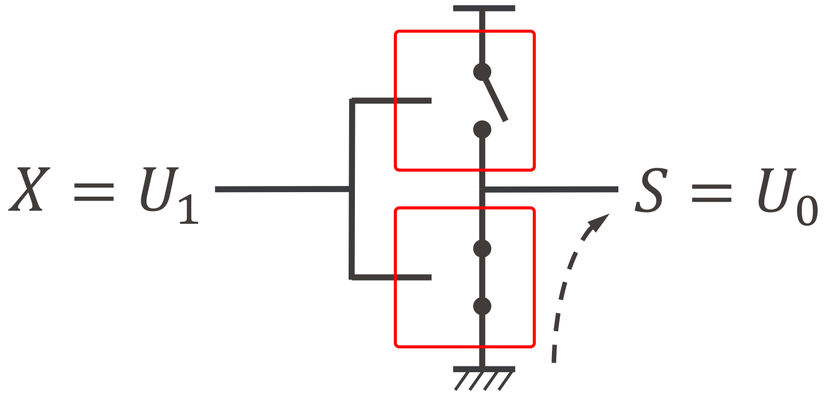
- Si la tension à la base est ***basse*** ($U_0 = 0V$) alors le courant passe entre l'émetteur et le collecteur :

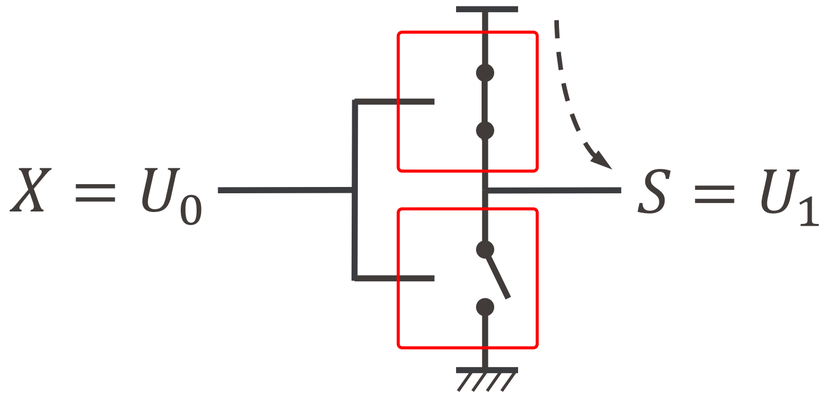


Création d'un inverseur

- 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

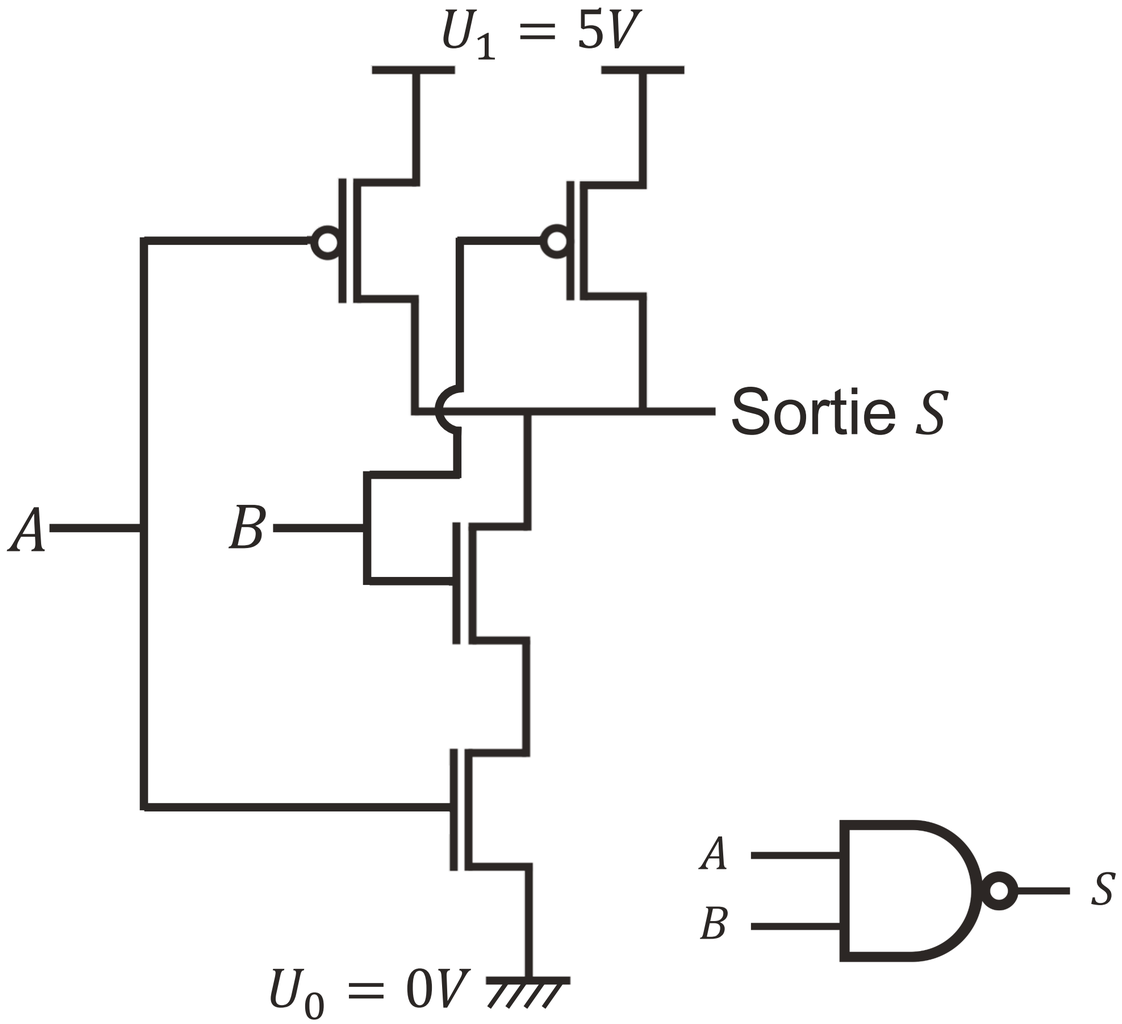






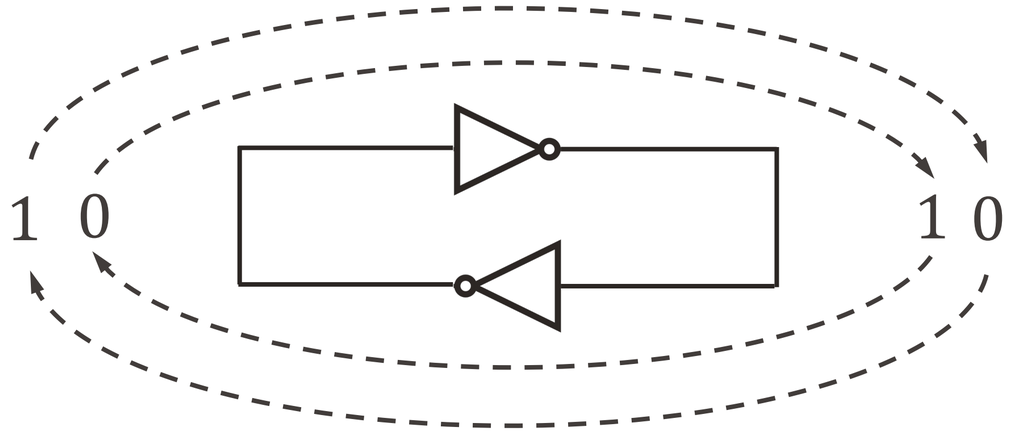
- En exercice : on peut généraliser aux portes AND et OR, il faut ***6 transistors*** pour créer ces portes (contre ***4*** pour les portes NAND et NOR).

Création de la porte NAND



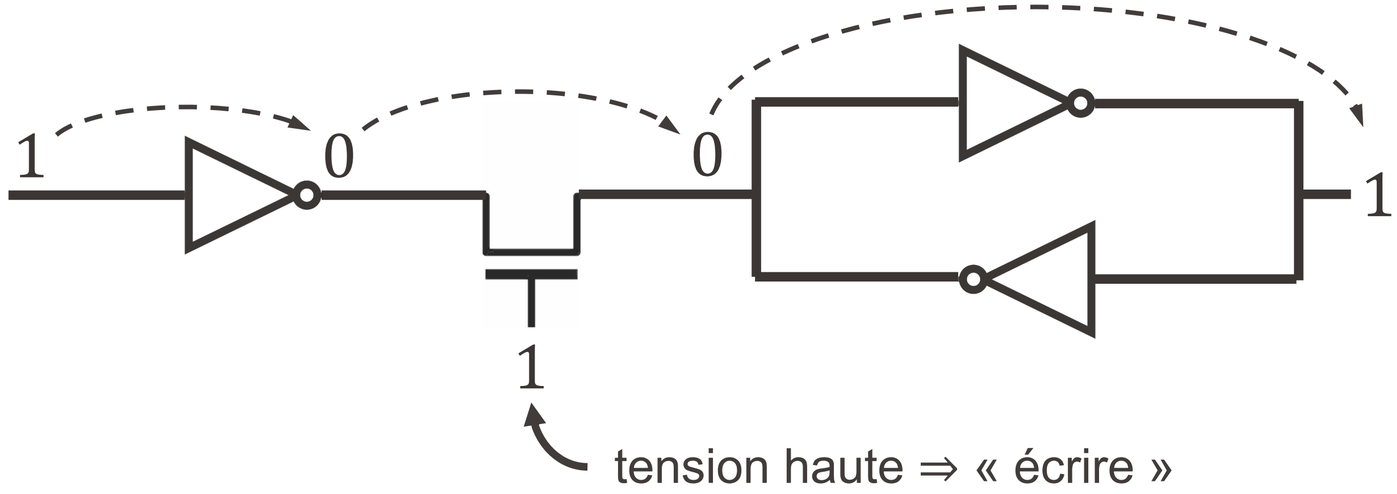
Mémoire vive

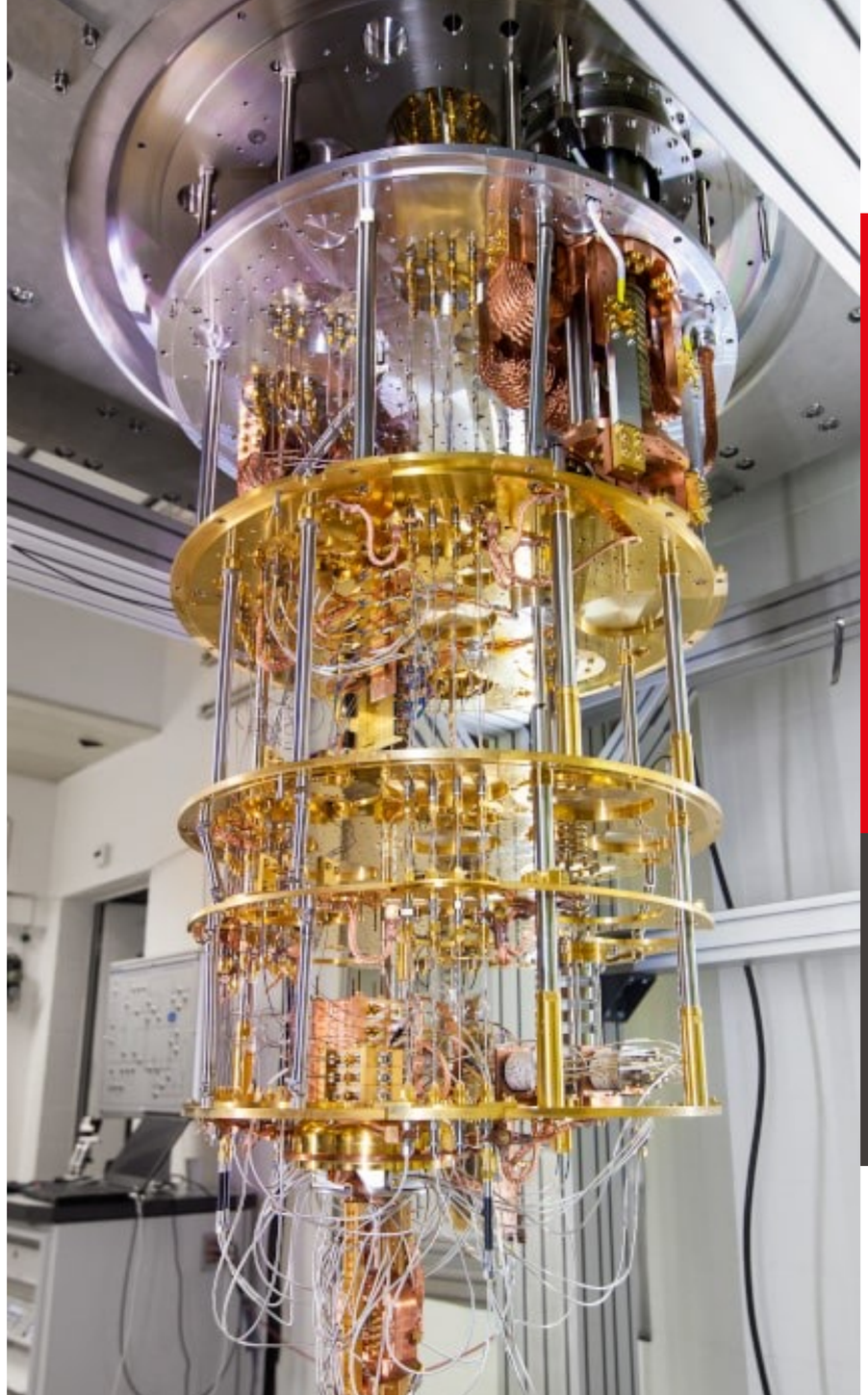
« Comment garder un bit 0 ou 1 en ***mémoire*** dans un circuit électrique ? »



→ Circuit stable !

- Écrire dans la mémoire :





Information, Calcul et Communication

Circuits quantiques

Olivier Lévêque

École polytechnique fédérale de Lausanne

Introduction

- 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…

Illustration de l'algorithme de Deutsch-Josza dans un cas simple

- Voici le problème à résoudre :

Etant donné une fonction $f: \{0,1\} \to \{0,1\}$,

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 function 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 !

Qubits

- En informatique quantique, les bits quantiques, ou “qubits”, remplacent les bits classiques.

- Un qubit peut valoir 0 ou 1, comme un bit classique, mais peut aussi être dans ce qu’on appelle un **état de superposition**, par exemple dans l’état “$0 + 1$” ou l’état “$0 - 1$” [attention: ne pas effectuer l’addition de 0 et 1 ici !].

- Pour résoudre le problème de Deutsch-Josza à l’aide d’un circuit quantique, nous allons avoir besoin de **deux qubits**, pouvant être dans les états $(0,0), (0,1), (1,0), (1,1)$ ou encore dans toute superposition de ceux-ci.

- Et bien sûr, nous allons aussi avoir besoin de **portes quantiques**…

Porte quantique H (H pour “Hadamard”)

- La porte quantique H est une porte avec un qubit en entrée et en sortie, qui permet de créer des états superposés :

$$\begin{cases} 0 \rightarrow H(0) = \text{“}0 + 1\text{”} \\ 1 \rightarrow H(1) = \text{“}0 - 1\text{”} \end{cases}$$

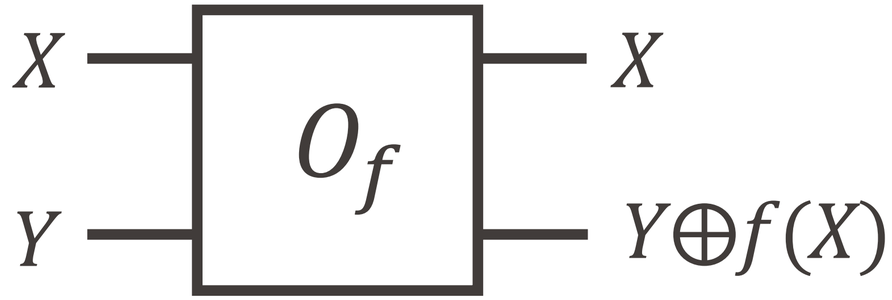
Porte H	
X	$H(X)$
0	“$0 + 1$”
1	“$0 - 1$”

- Grâce à cette porte, il est possible, en agissant individuellement sur chaque qubit, de créer l’état de superposition suivant de deux qubits, que nous allons utiliser pour l’algorithme de Deutsch-Josza :

$$(0,0) - (0,1) + (1,0) - (1,1)$$

Porte quantique O_f (O pour “Oracle”)

- La porte quantique O_f est une porte avec deux qubits en entrée et en sortie :



Porte O_f				
X	Y		X	$Y \oplus f(X)$
0	0		0	$f(0)$
0	1	$\rightarrow$	0	$1 \oplus f(0)$
1	0		1	$f(1)$
1	1		1	$1 \oplus f(1)$

Circuit de Deutsch-Josza: étape 1

- Partons de la superposition d'états $(0,0) - (0,1) + (1,0) - (1,1)$, et passons celle-ci à travers la porte O_f. Ceci donne :

$$(0, f(0)) - (0,1 \oplus f(0)) + (1, f(1)) - (1,1 \oplus f(1))$$

Porte O_f				
X	Y		X	$Y \oplus f(X)$
0	0		0	$f(0)$
0	1	→	0	$1 \oplus f(0)$
1	0		1	$f(1)$
1	1		1	$1 \oplus f(1)$

Circuit de Deutsch-Josza: étape 2

- Après ça, nous passons encore le premier des deux qubits à travers une porte de Hadamard, ce qui donne :

$$(0, f(0)) - (0,1 \oplus f(0)) + (1, f(1)) - (1,1 \oplus f(1))$$
$$\downarrow$$
$$(0, f(0)) + (1, f(0)) - \big(0,1 \oplus f(0)\big) - \big(1,1 \oplus f(0)\big)$$
$$+ \big(0, f(1)\big) - \big(1, f(1)\big) - \big(0,1 \oplus f(1)\big) + \big(1,1 \oplus f(1)\big)$$

Porte H	
X	$H(X)$
0	“$0 + 1$”
1	“$0 - 1$”

Circuit de Deutsch-Josza: étape 3

- En regroupant les termes par la valeur de leur premier qubit, nous trouvons :

$$(0, f(0)) + \big(0, f(1)\big) - \big(0{,}1 \oplus f(0)\big) - \big(0{,}1 \oplus f(1)\big)$$
$$+ (1, f(0)) - \big(1, f(1)\big) - \big(1{,}1 \oplus f(0)\big) + \big(1{,}1 \oplus f(1)\big)$$

- Si $f(1) = f(0)$, alors les termes dont le premier qubit vaut 1 (seconde ligne) s'annulent, et donc il ne reste plus que les termes de la première ligne.

- Si $f(1) \neq f(0)$, autrement dit : $f(1) = 1 \oplus f(0)$, alors ce sont les termes de la première ligne qui s'annulent ; il ne reste plus que les termes de la seconde.

- En conclusion, nous obtenons la réponse en mesurant la valeur du premier qubit en sortie : si celle-ci vaut 0, alors $f(1) = f(0)$; sinon, $f(1) \neq f(0)$.

Conclusion

- La circuit quantique de Deutsch-Josza permet de déterminer si $f(1) = f(0)$ ou au contraire si $f(1) \neq f(0)$ en une seule évaluation de la function f (via l'oracle O_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 !