

# Représentation des nombres réels en virgule flottante

"mantisse" "exposant"  $\in \{-2^{m-1} \dots 2^{m-1} - 1\}$

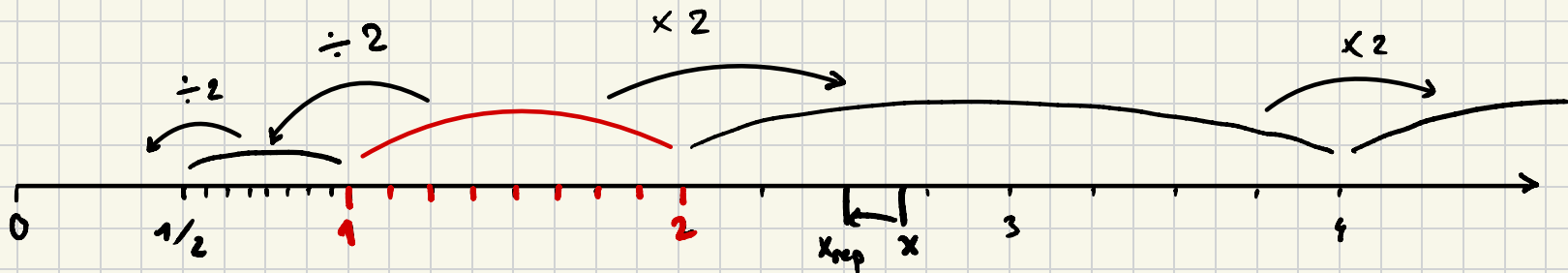
$\downarrow$   $\downarrow$

$X_{\text{rep}} = \pm (1, \overbrace{b_1 \dots b_n}) \cdot 2^{\overbrace{a_1 \dots a_m}}$

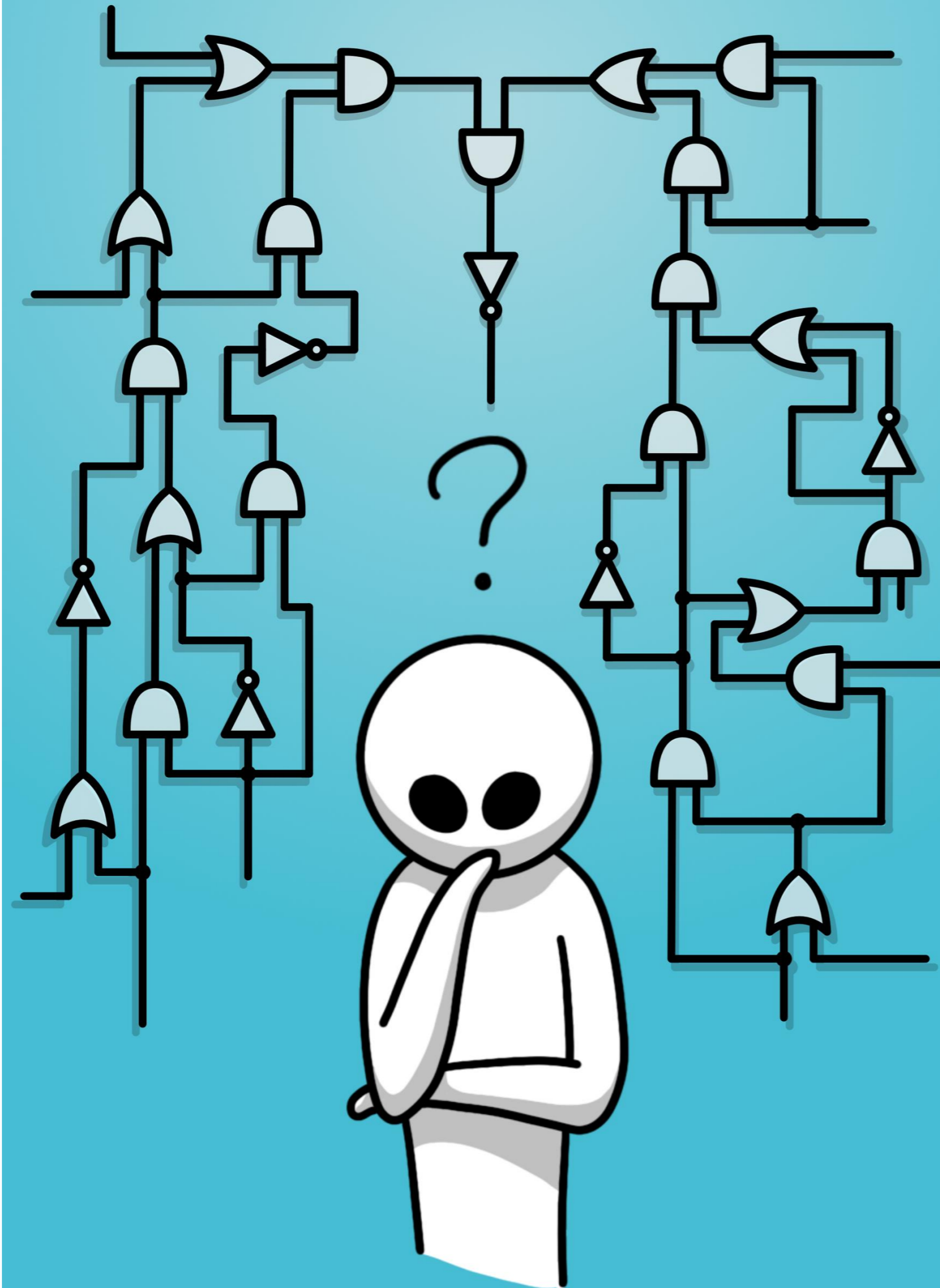
encodage binaire:  $c \ a_1 \dots a_m \ b_1 \dots b_n$

$\uparrow$   
bit de signe

float:  $m=8, n=23$



- 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

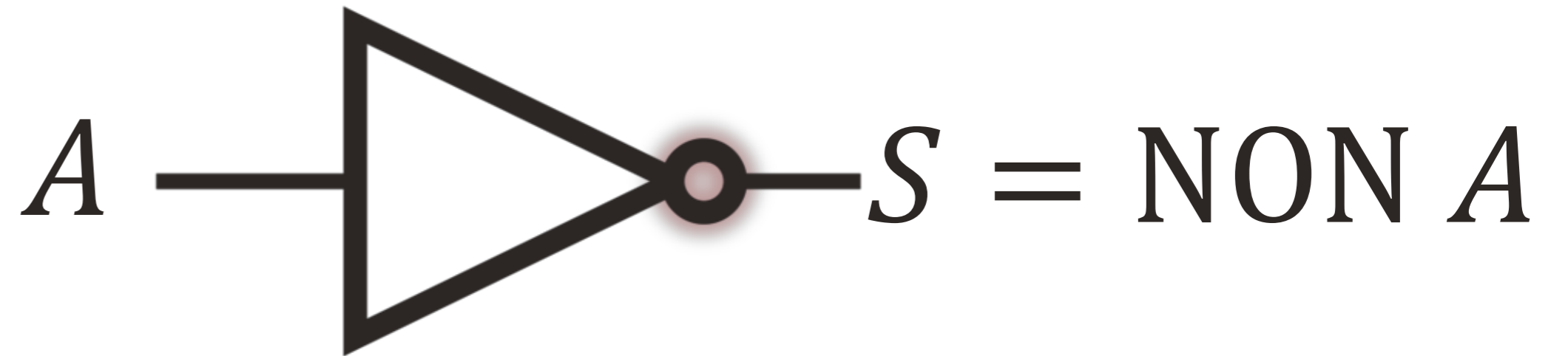
Circuits logiques

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
- 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*)

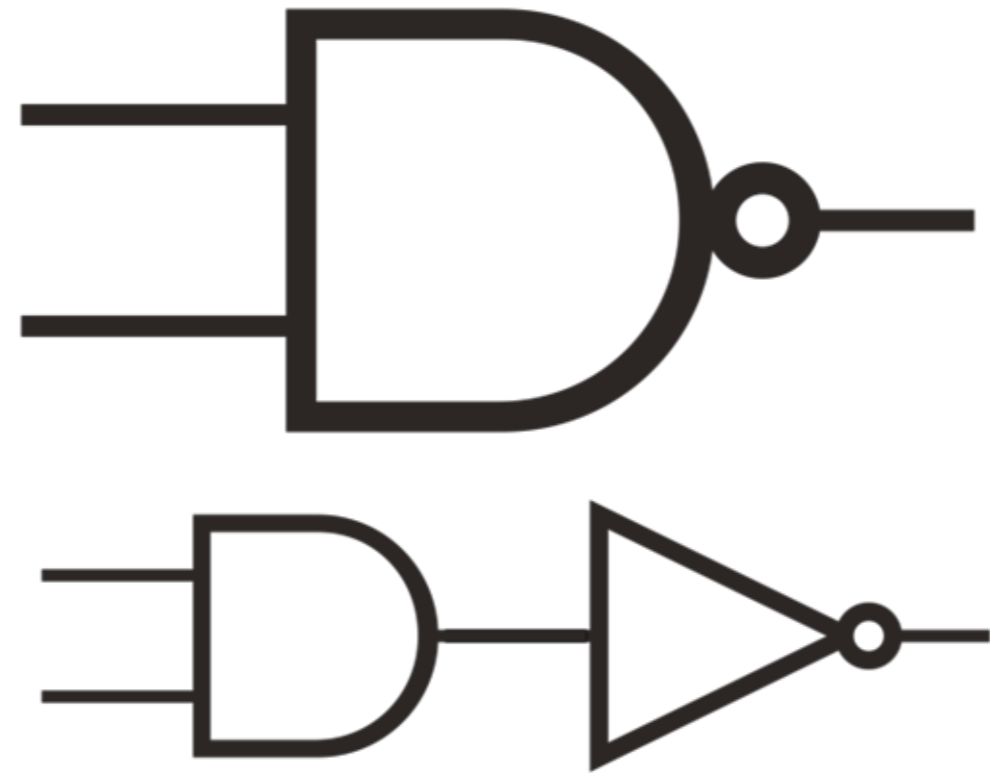
- Elle comporte deux ou plusieurs entrées.
- La porte OU génère un 1 en sortie si au moins un des bits en entrée vaut 1. La sortie vaut donc 0 en sortie si et seulement si les deux bits en entrée valent 0.
- Notez que la valeur de sortie  $S$  vaut 1 quand  $A + B \geq 1$  (mais n'est donc **pas** égale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



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

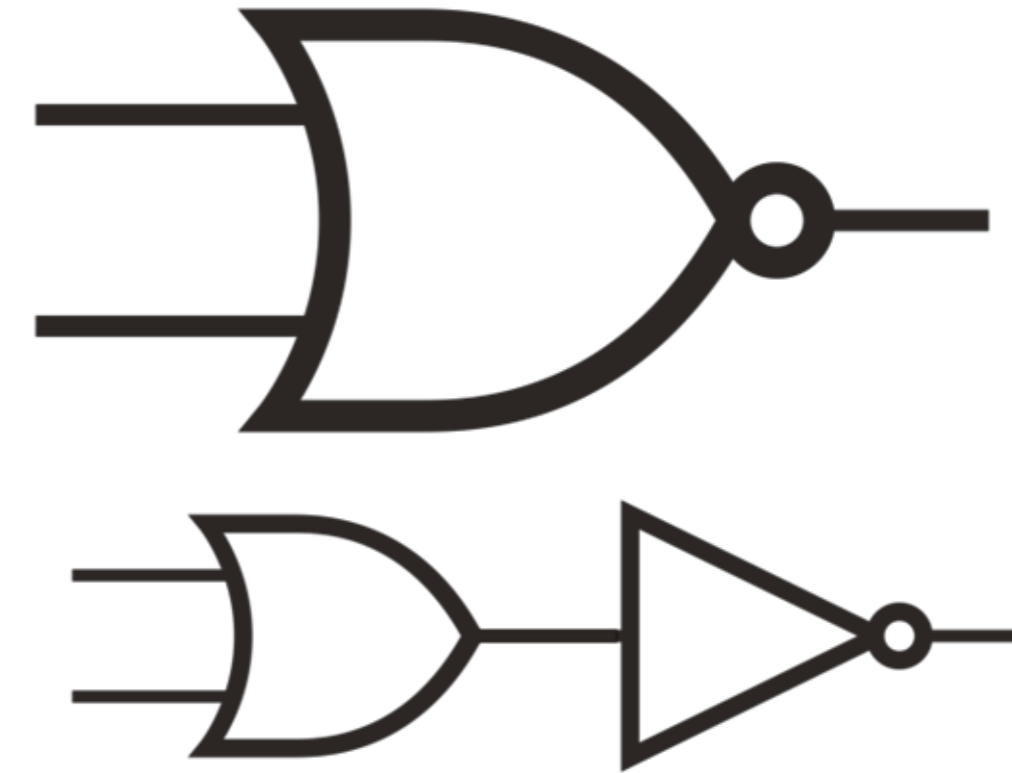
# Les portes NON ET (*NAND*) et NON OU (*NOR*)

## ■ 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*) = addition modulo 2

- 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

"Si et seulement si"

$$S = 1 \quad \underline{\underline{\text{ssi}}} \quad (A=0 \text{ et } B=1) \quad \text{ou} \quad (A=1 \text{ et } B=0)$$

$$S = 1 \quad \underline{\underline{\text{ssi}}} \quad (\text{non } A=1 \text{ et } B=1) \quad \text{ou} \quad (A=1 \text{ et } \text{non } B=1)$$

$$S = (\text{non } A \text{ et } B) \quad \text{ou} \quad (A \text{ et } \text{non } B)$$

# Additionner deux bits (*sans retenue*)

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

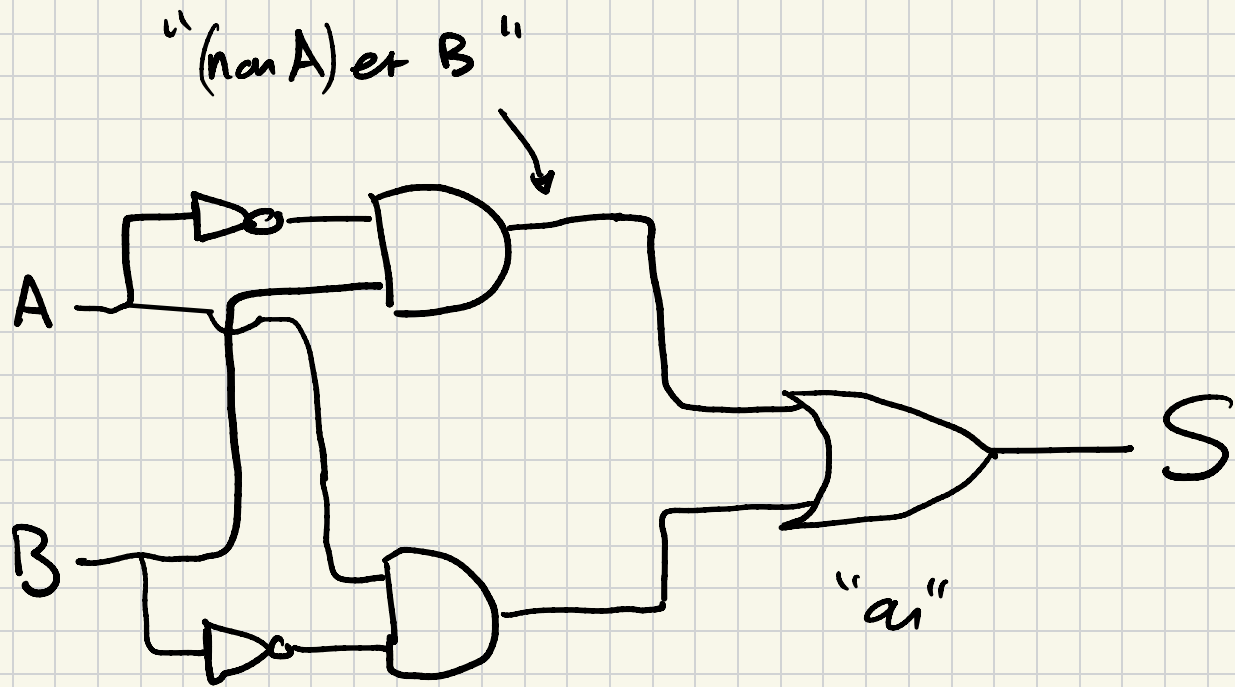
Additionneur		
$A$	$B$	$S$
0	0	0
0	1	1
1	0	1
1	1	0

- 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 et B"

"A et non B"

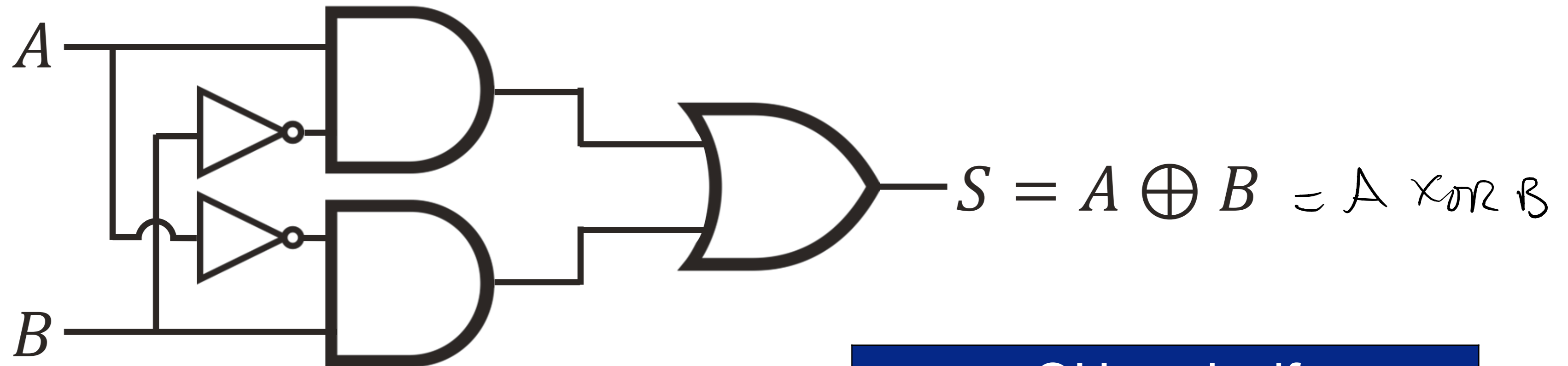
"a<sub>i</sub>"

"a<sub>i</sub> exclusif"

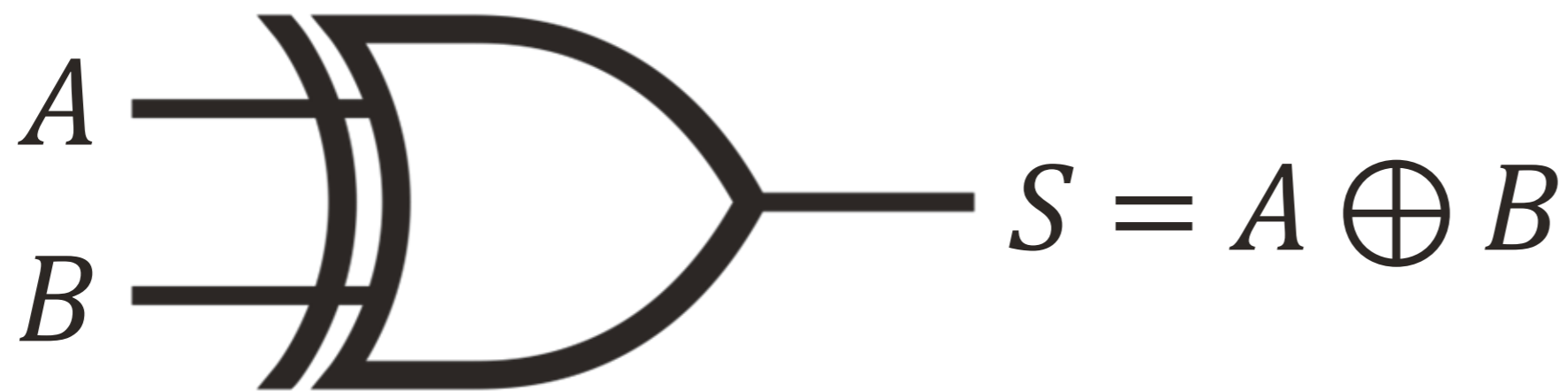
additionneur modulo 2 = porte XOR

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

- Circuit correspondant :



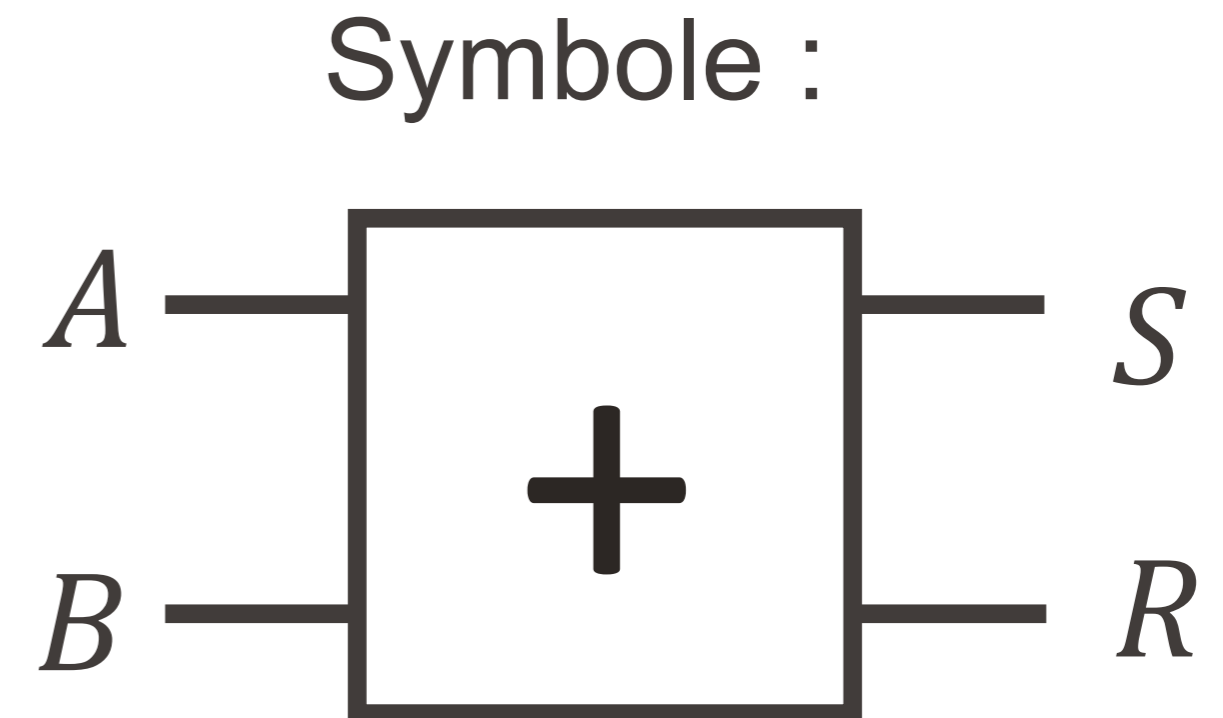
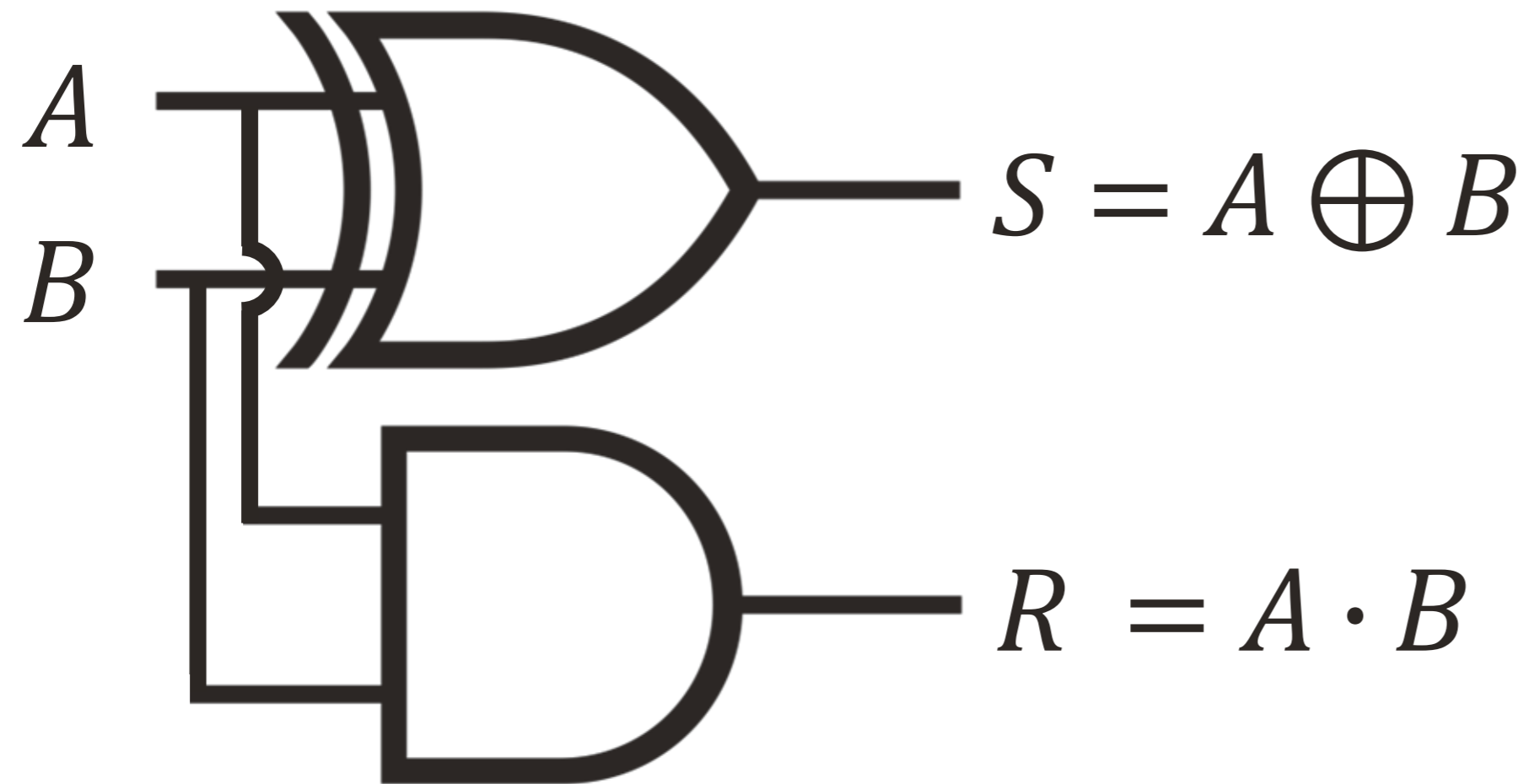
- Symbole résumant ce nouveau circuit :



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)*

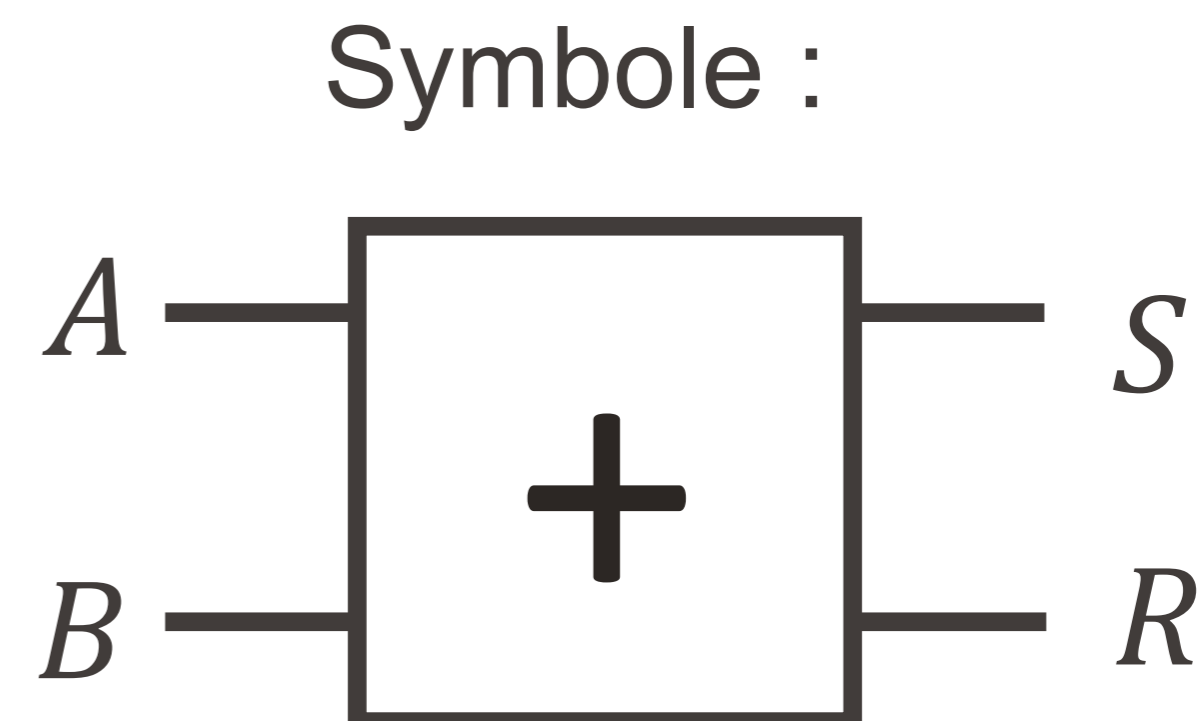
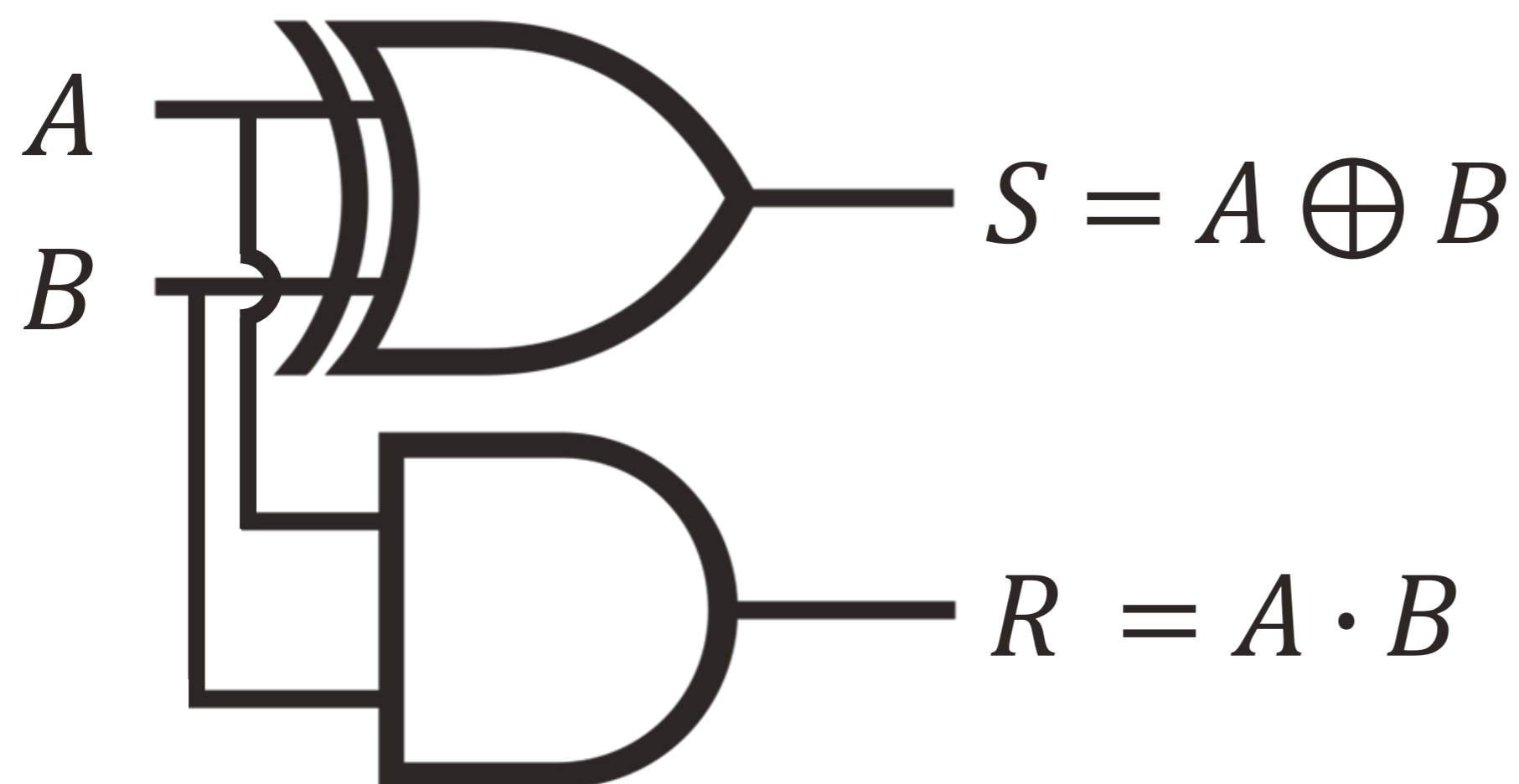
- On aimerait maintenant créer un circuit avec entrées  $A$  et  $B$  et sortie  $S = A \oplus B$ , ainsi qu'une retenue  $R = 1$  si et seulement si  $A = 1$  et  $B = 1$ .



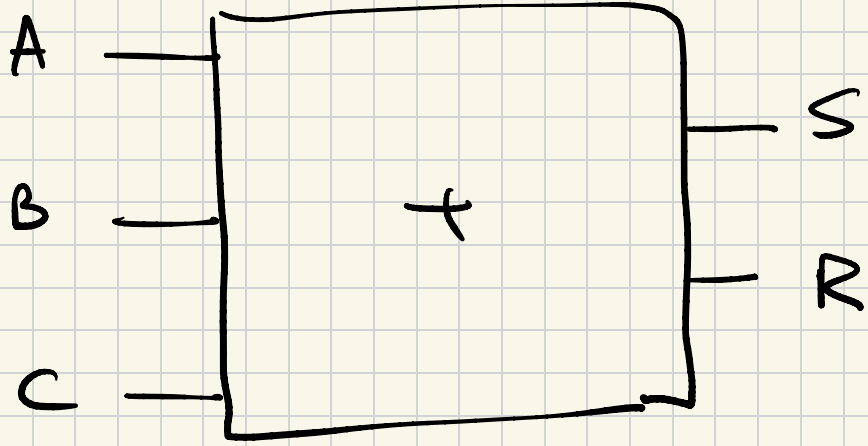
$A$	$B$	$R$	$S$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

# 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  $R_1$ .



# 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.

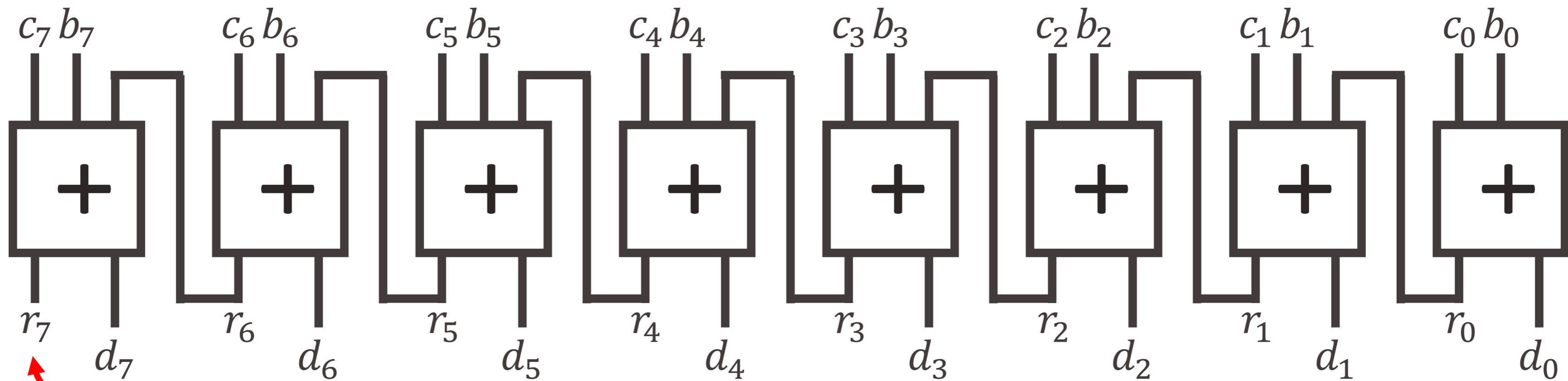
# Additionneur sur 8 bits

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 :

$$\begin{array}{r}
 11111 \\
 00111001 \\
 + 00101011 \\
 \hline
 = 01100100
 \end{array}$$

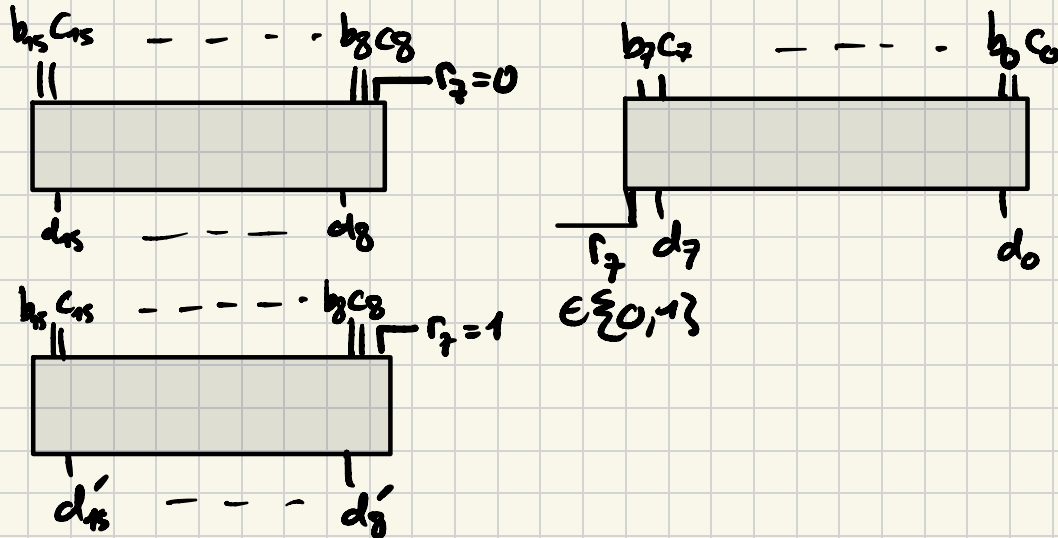


Si  $r_7 = 1 \Rightarrow$  **overflow !**

# Calcul parallèle

deux nombres  $x$  et  $y$  représentés sur 16 bits

$b_{15} \dots b_0$        $c_{15} \dots c_0$





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

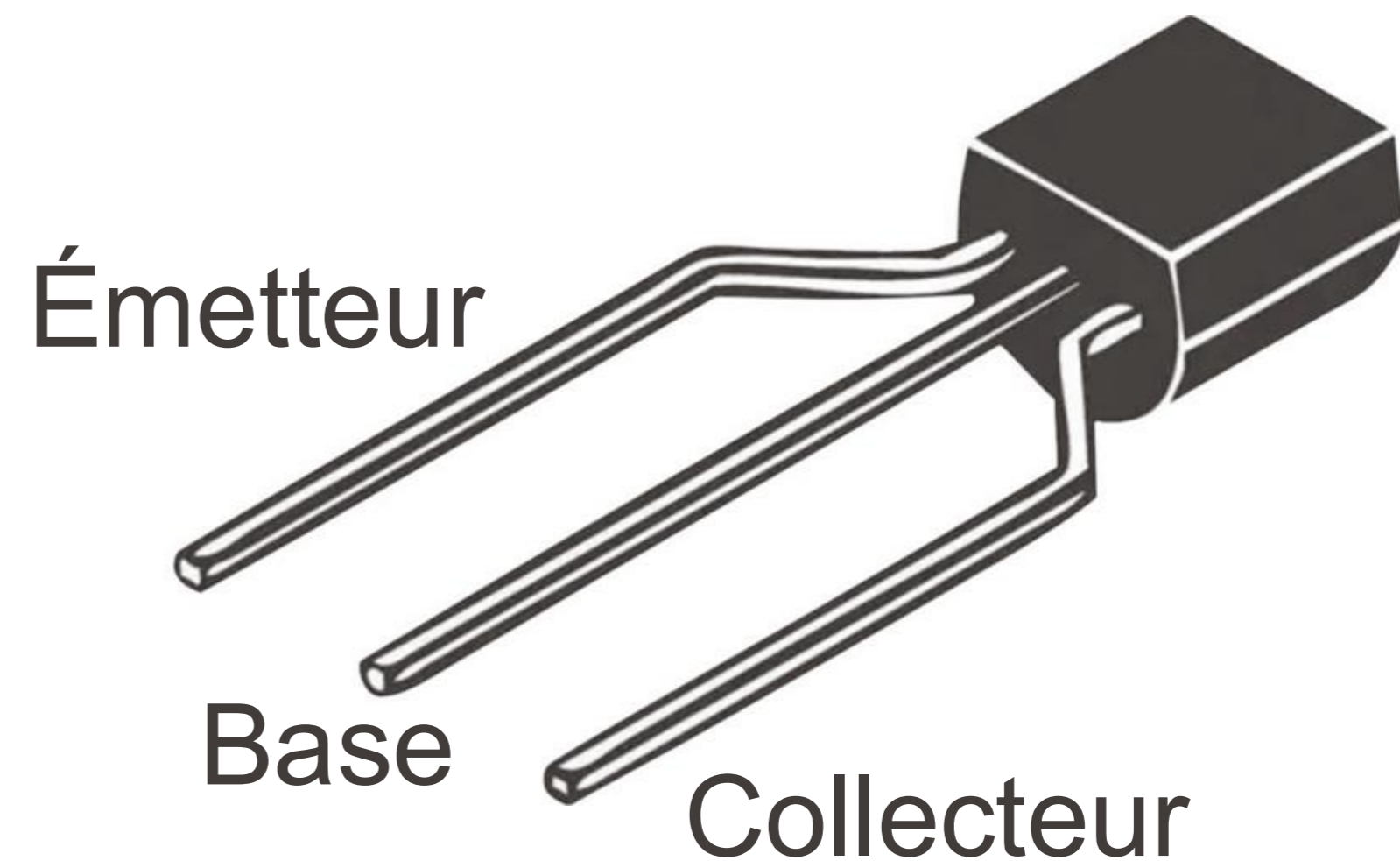
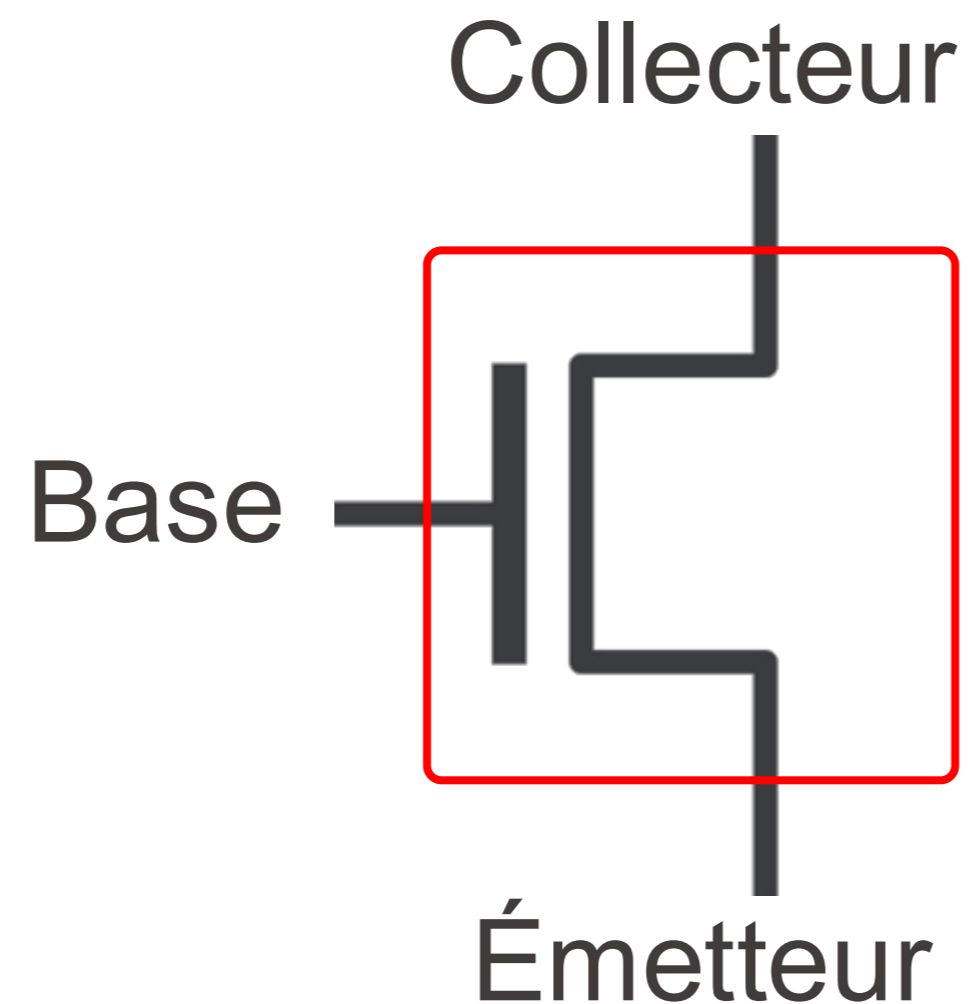
# Information, Calcul et Communication

**Transistors**

Olivier Lévêque

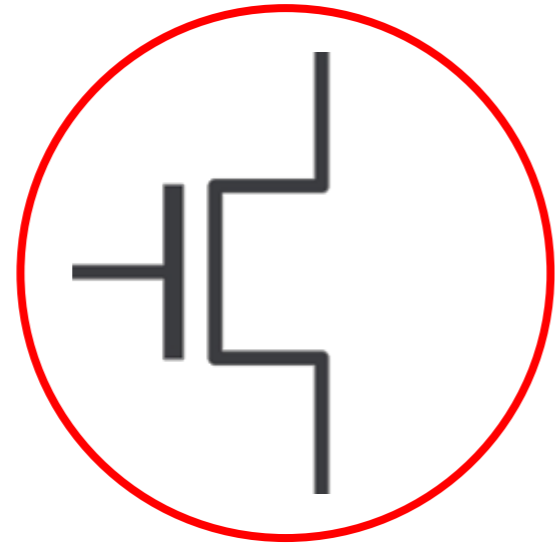
- Inventé en 1947 par trois américains : Bardeen, Shockley & Brattain
- Ce composant, qui est à la base de toute l'électronique moderne, a remplacé avantageusement les relais électromécaniques et les tubes à vide utilisés dans les premiers ordinateurs à la même époque → **miniaturisation**

- Schéma :

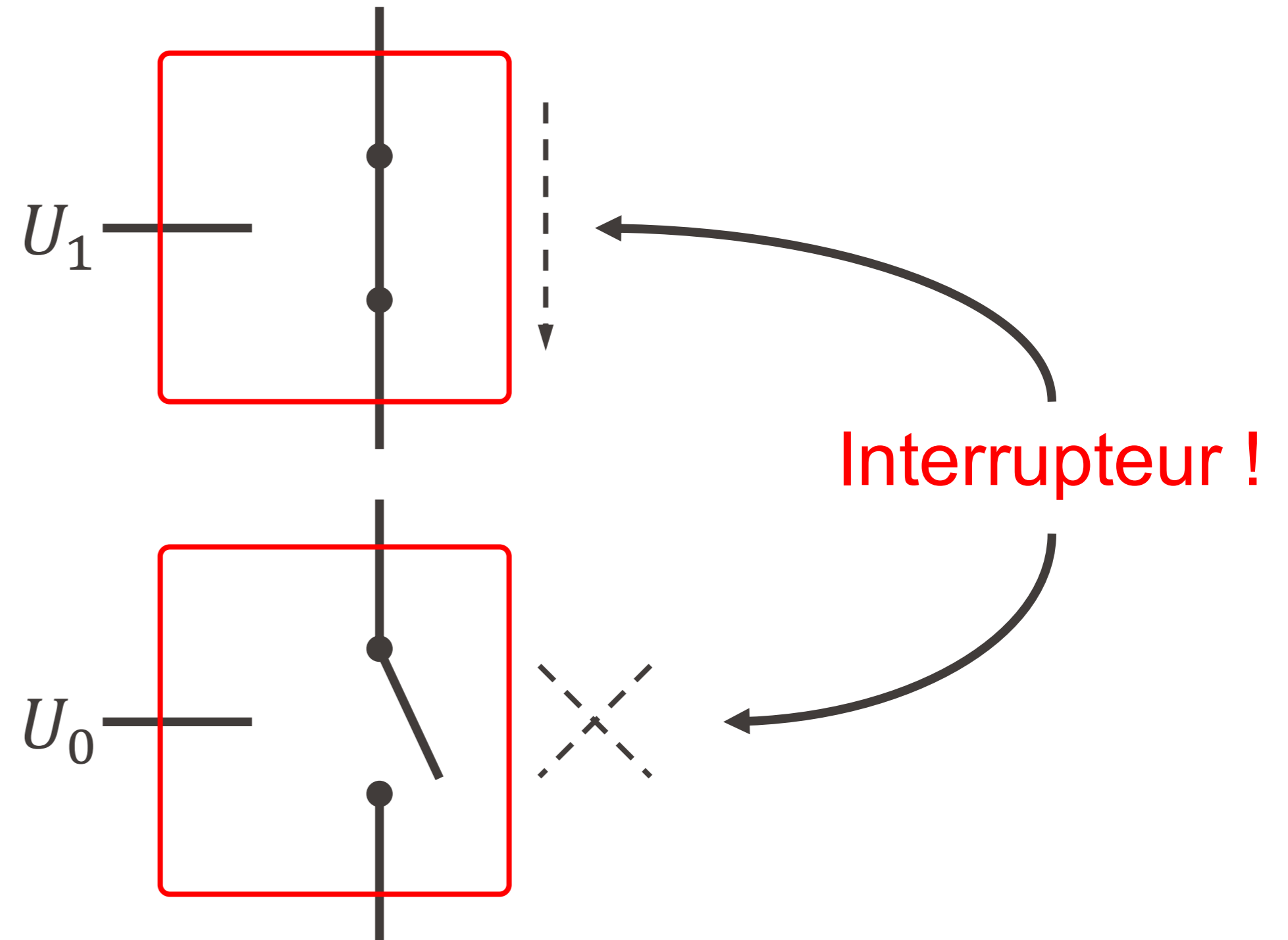


# Principe de fonctionnement (*n-mos*)

- Symbole :

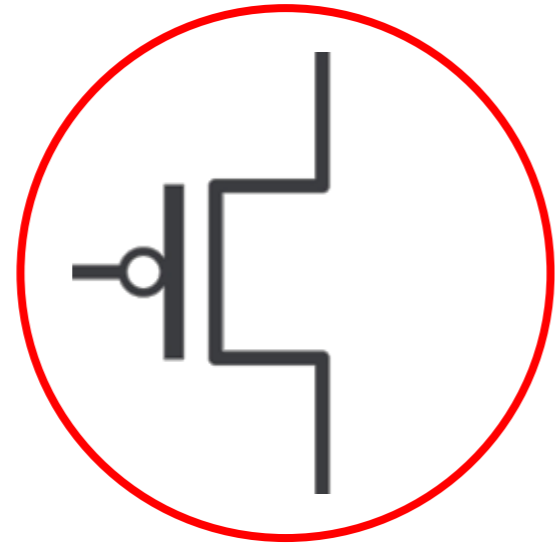


- Si la tension à la base est **haute** ( $U_1 = 5V$ ) alors le courant passe entre l'émetteur et le collecteur :
- Si la tension à la base est **basse** ( $U_0 = 0V$ ) alors le courant ne passe pas entre l'émetteur et le collecteur :

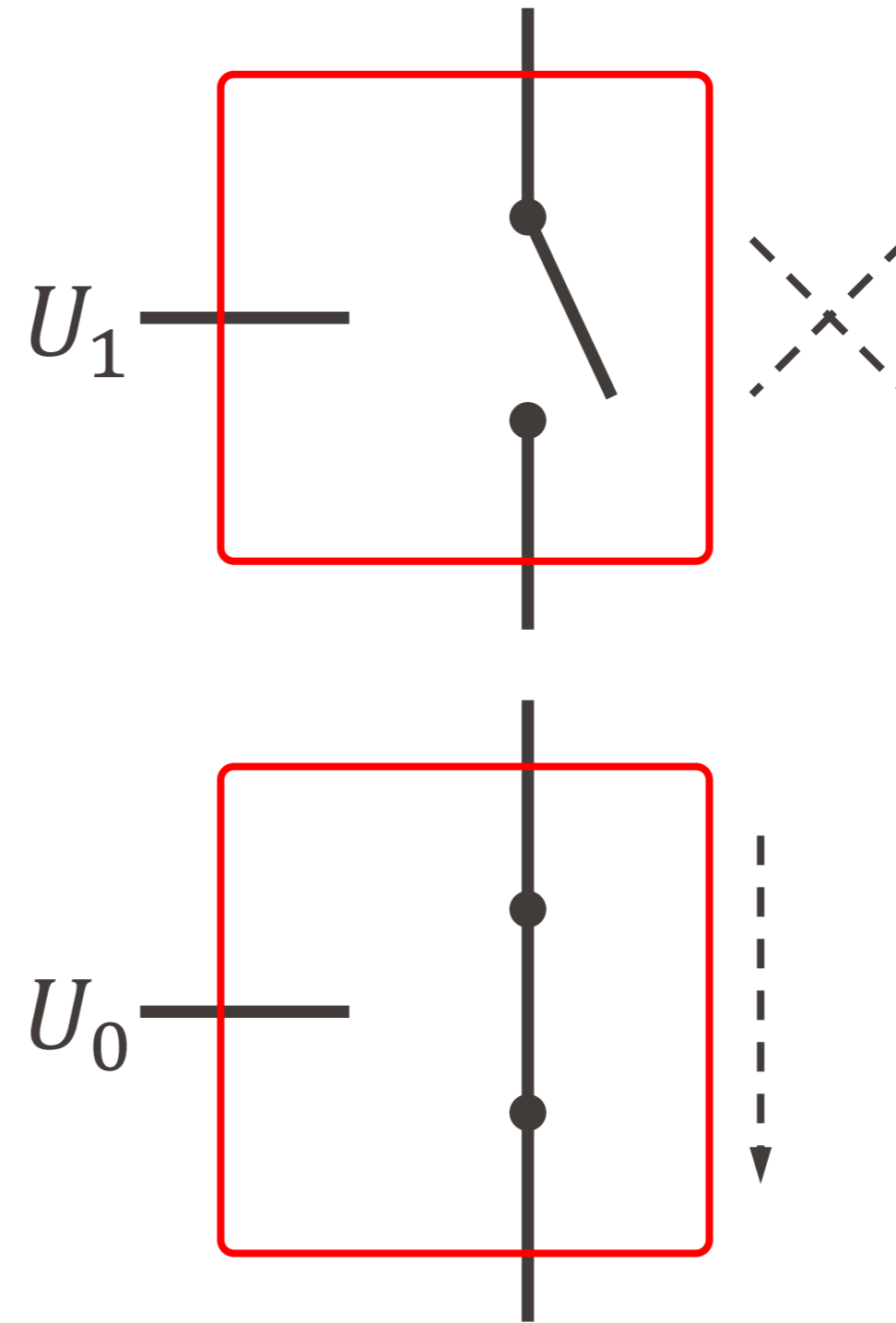


# Principe de fonctionnement (*p-mos*)

- Symbole :



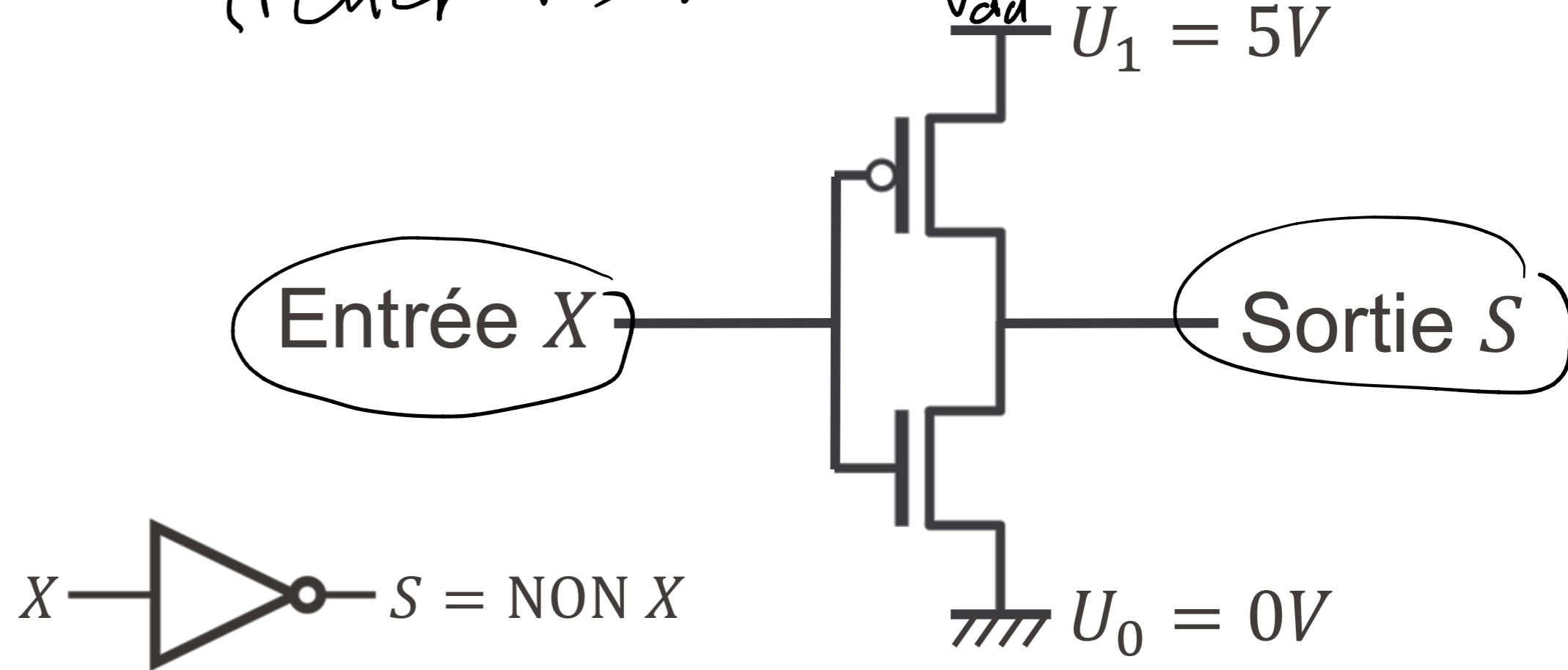
- Si la tension à la base est **haute** ( $U_1 = 5V$ ) alors le courant ne passe pas entre l'émetteur et le collecteur :
- Si la tension à la base est **basse** ( $U_0 = 0V$ ) alors le courant passe entre l'émetteur et le collecteur :



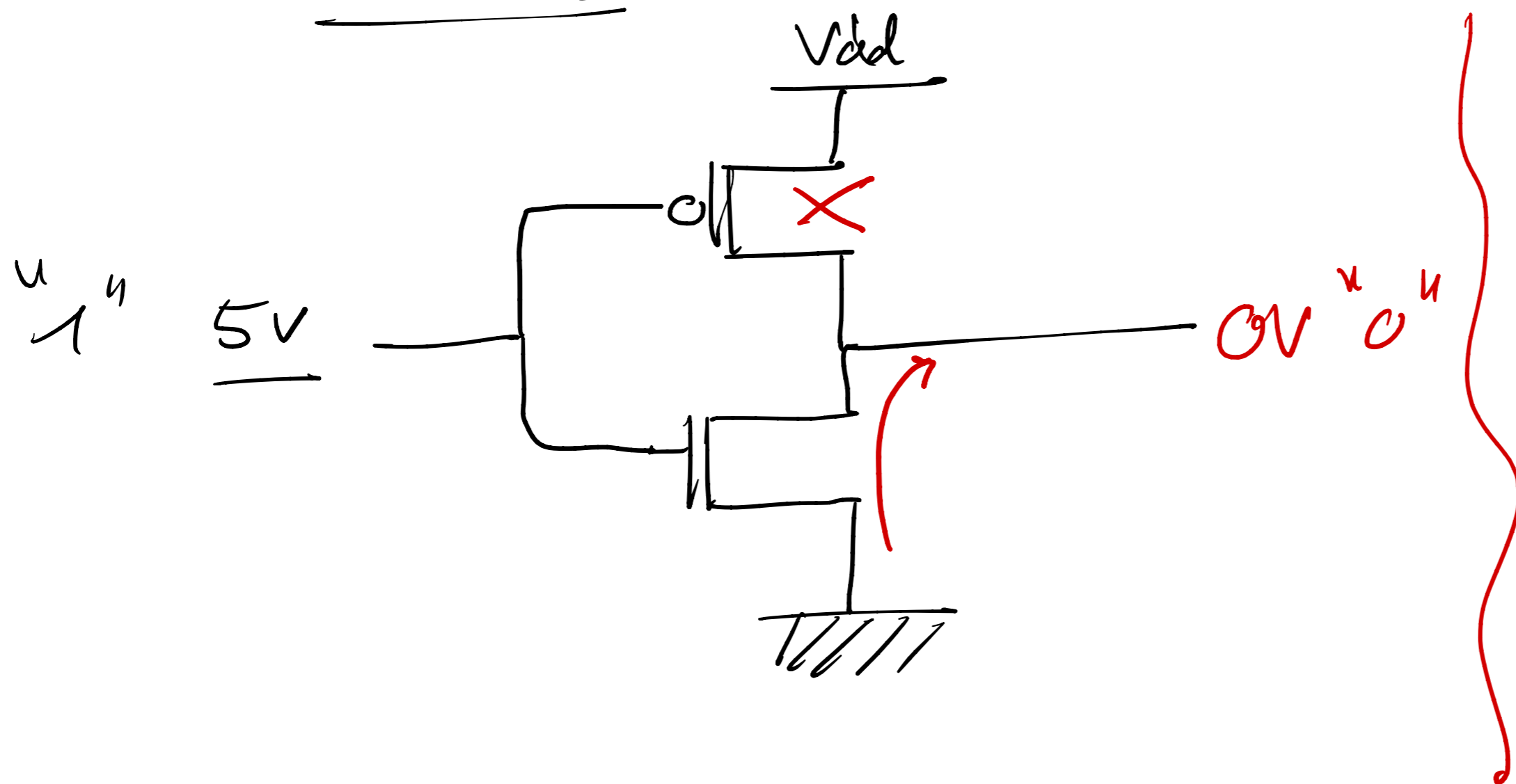
# Création d'un inverseur

- 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

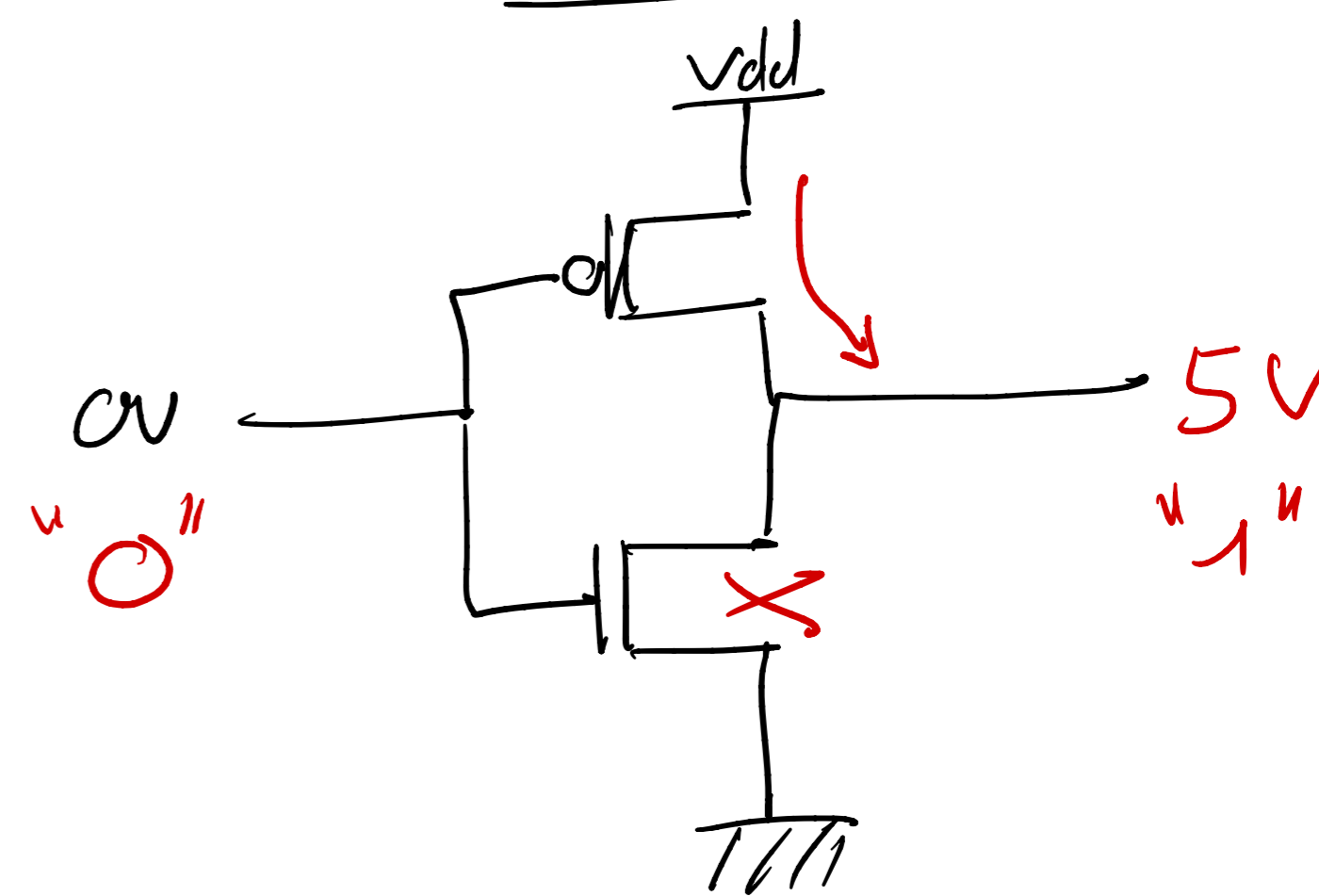
⚠️ { relier les p-mos à la partie sans tension ( $V_{dd}$ )  
relier les n-mos à la terre ( $U_0 = 0V$ )



1er cas

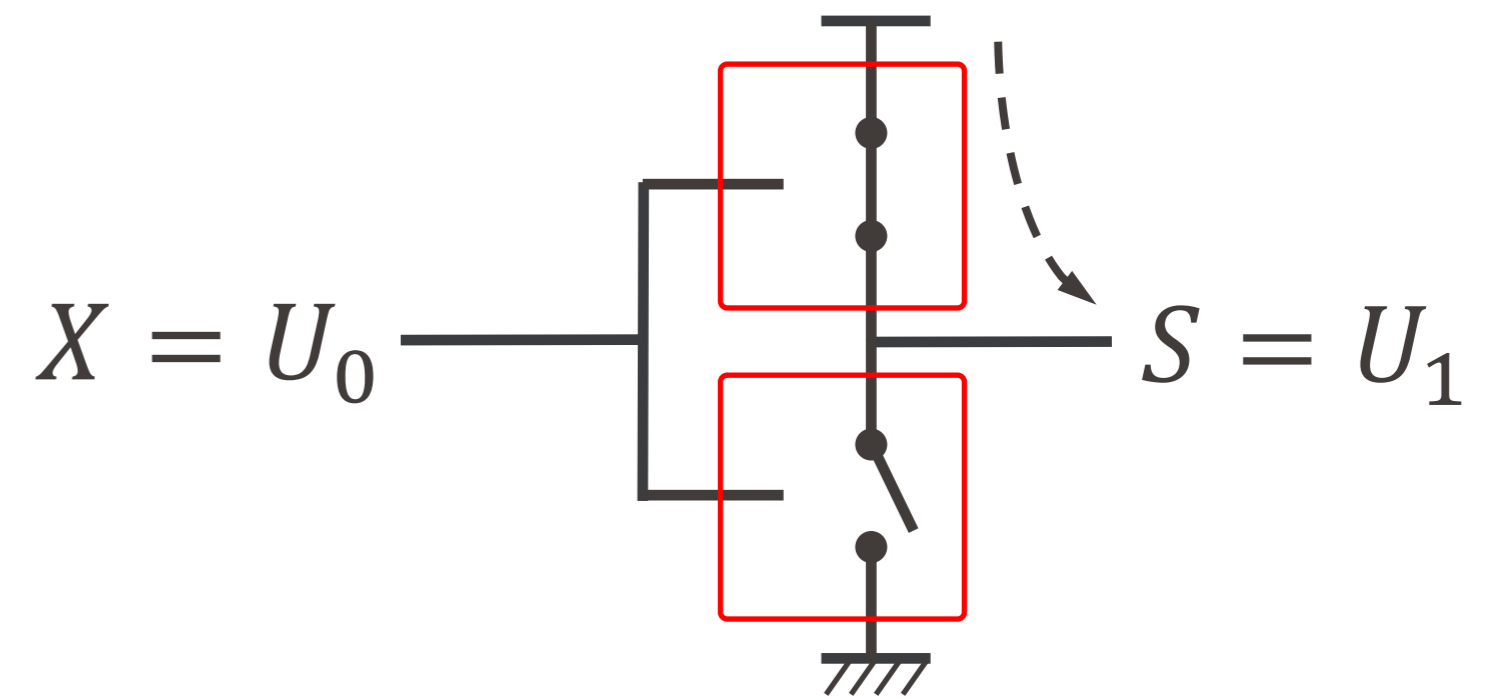
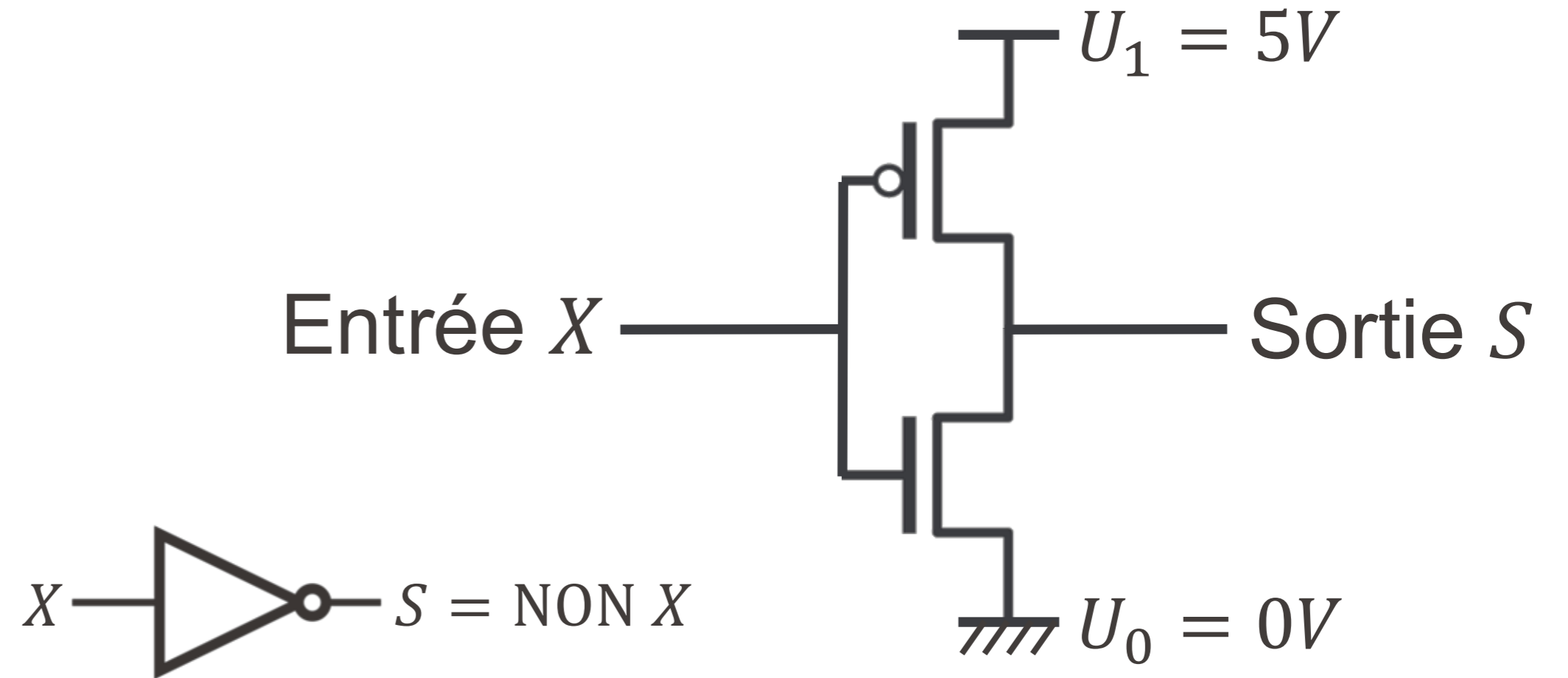
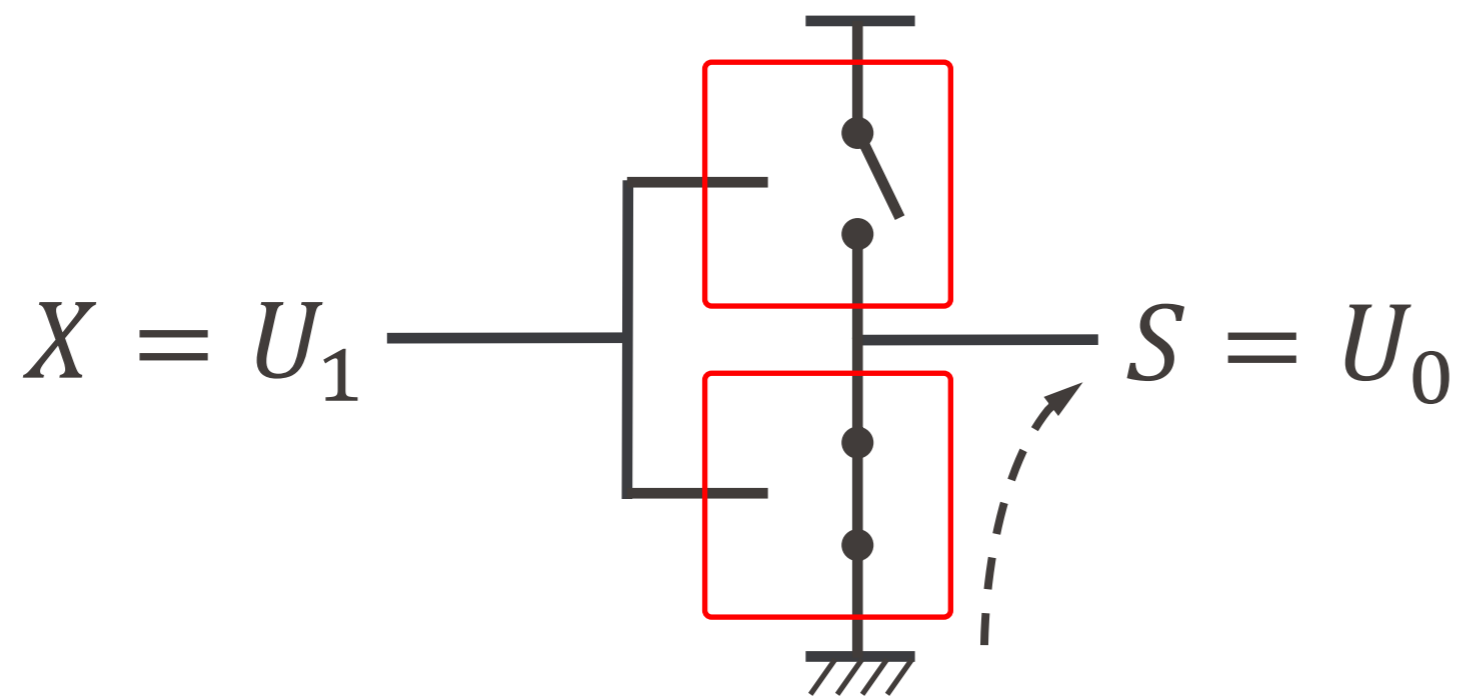


2e cas



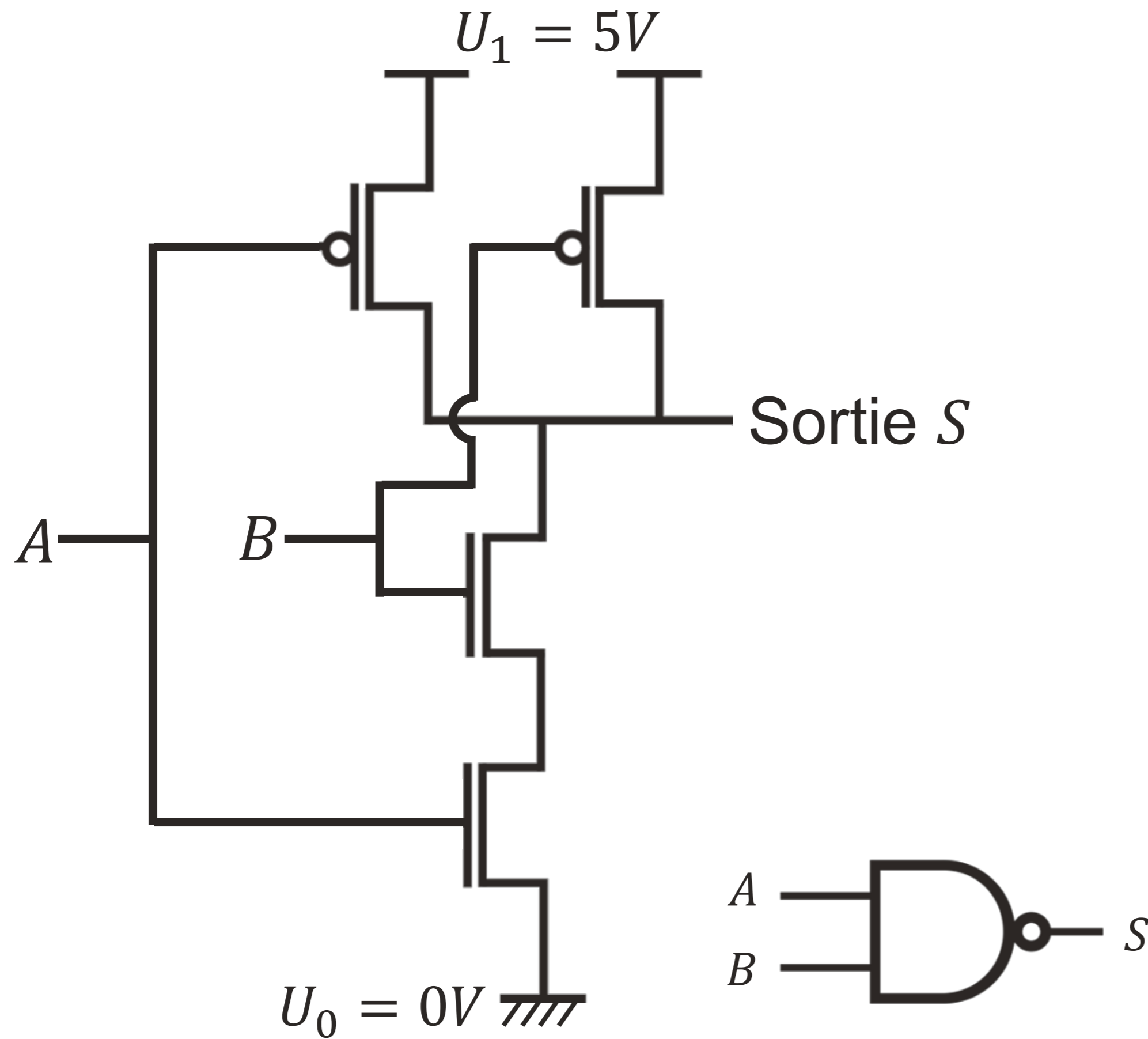
# Création d'un inverseur

- 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

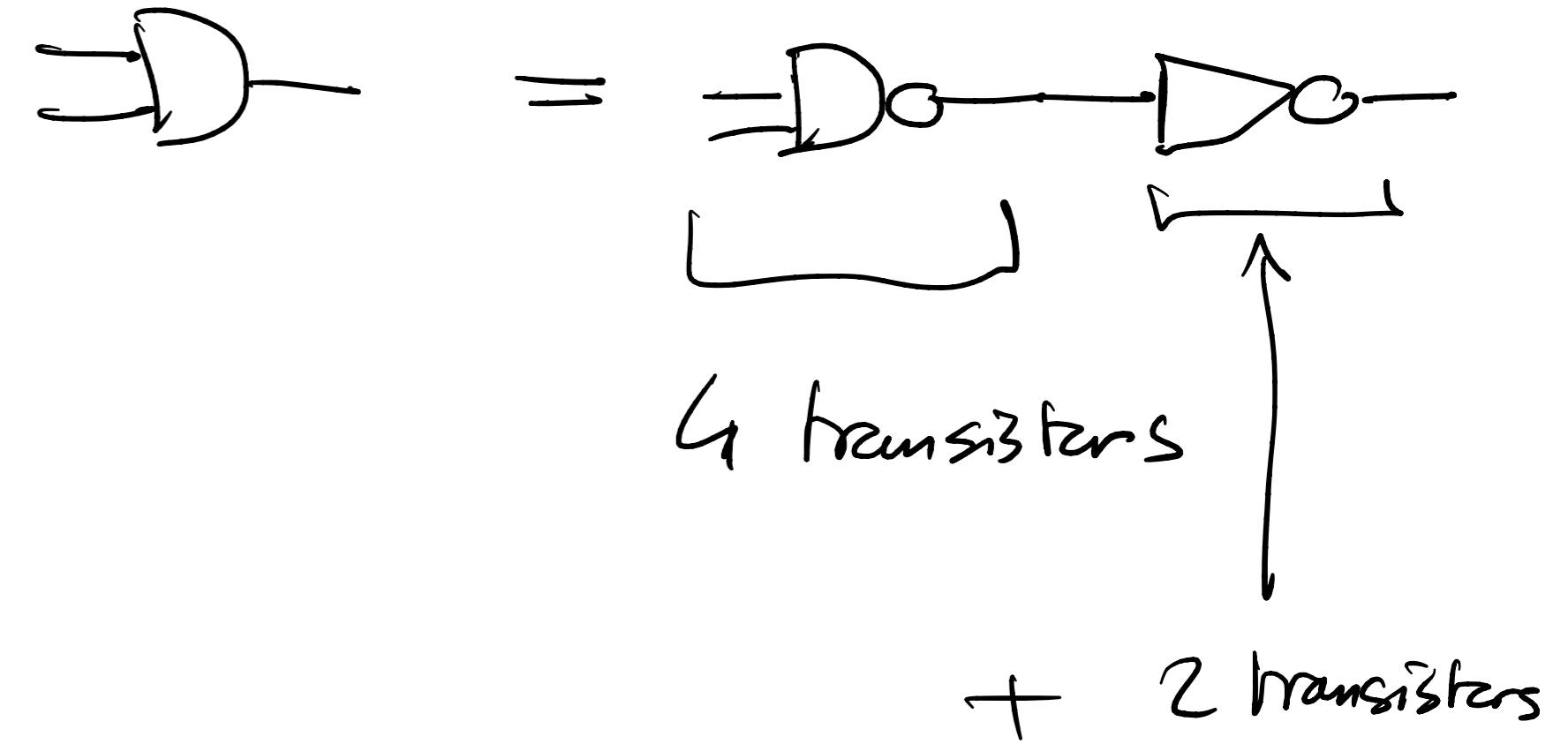


- En exercice : on peut généraliser aux portes AND et OR, il faut **6 transistors** pour créer ces portes (contre **4** pour les portes NAND et NOR).

# Création de la porte NAND

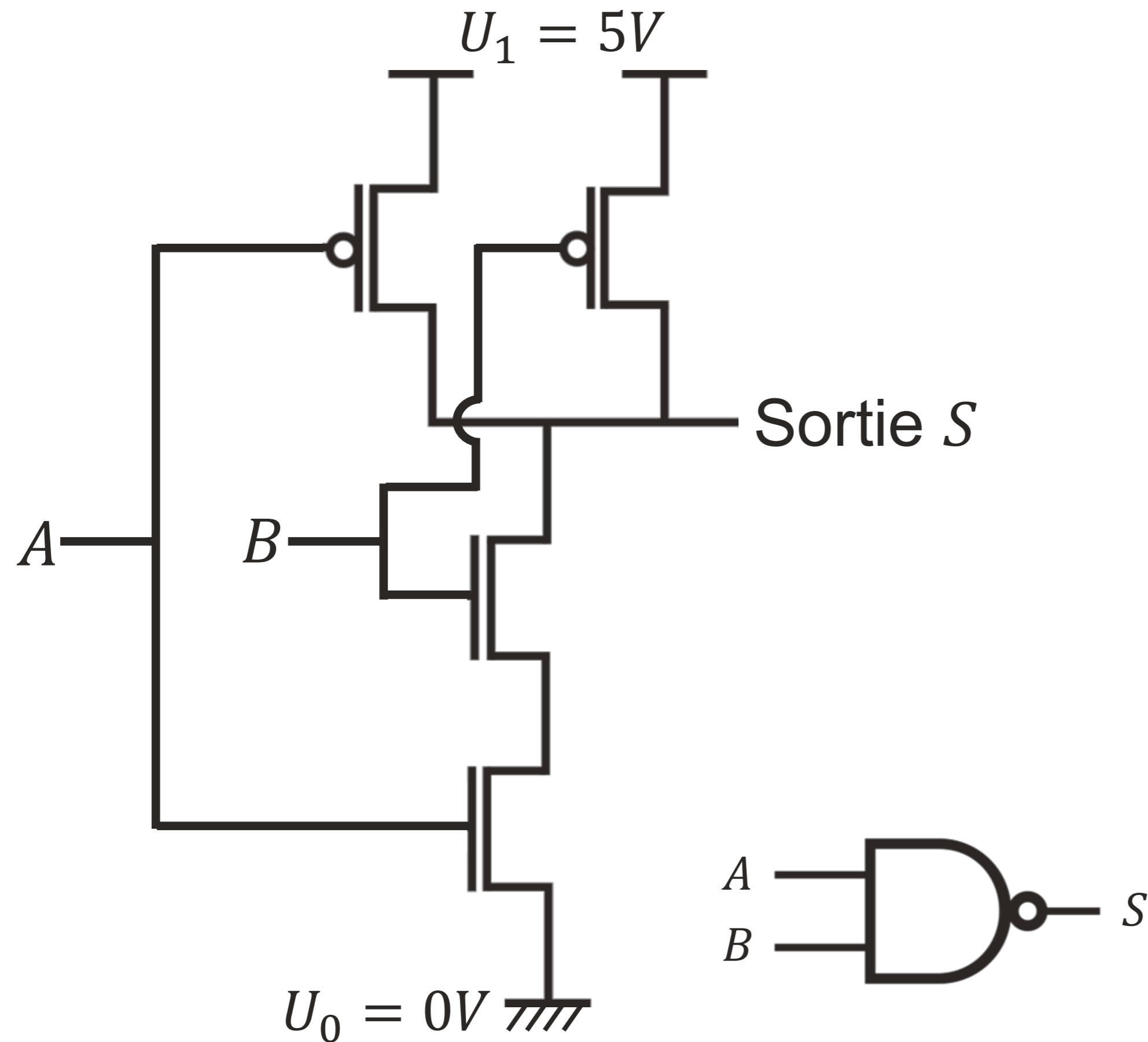


PORTE AND



= 6 transistors

# Création de la porte NAND



*Note :* Pour des raisons techniques non expliquées dans ce cours, on ne peut pas relier un transistor n-mos à la partie du circuit sous tension (5V), ni relier un transistor p-mos à la partie du circuit reliée à la terre (0V).