

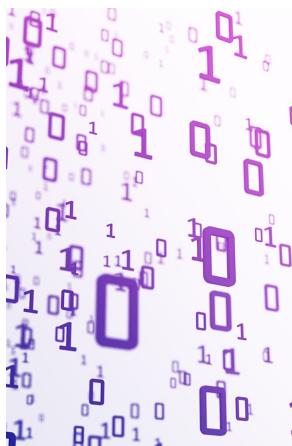
# **Information, Calcul et Communication**

## **CS-119(k) ICC – Théorie Semaine 7**

Rafael Pires

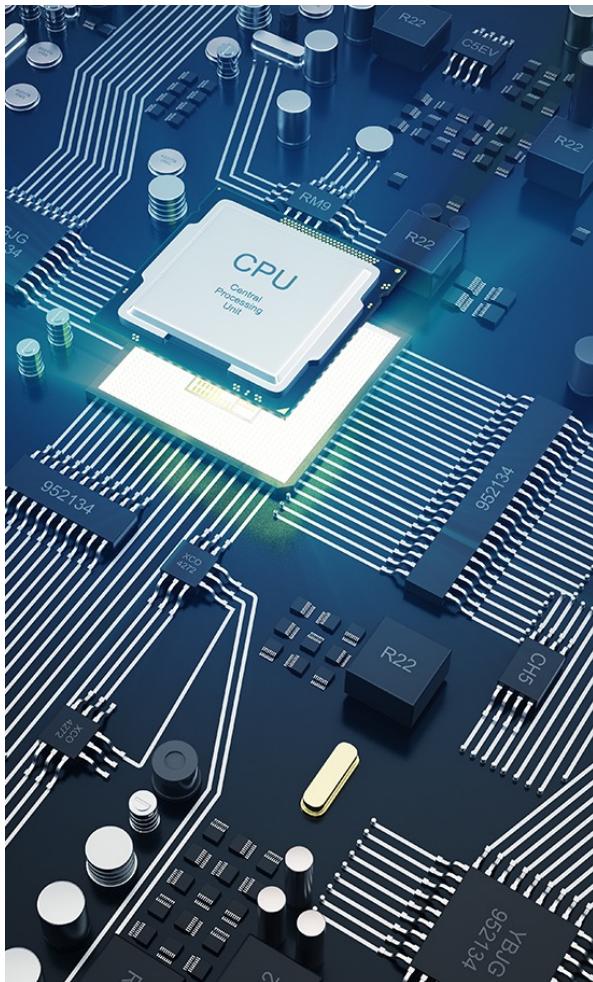
[rafael.pires@epfl.ch](mailto:rafael.pires@epfl.ch)

# Précédemment, dans ICC-T 06



- Suite du cours et représentation de l'information
- Représentation binaire des nombres entiers
  - Nombres positifs, négatifs (**complément à deux**)
  - Opérations (addition, soustraction, multiplication, division)
  - **Dépassemement de capacité**
- Représentation binaire des nombres réels
  - Virgule fixe : erreur relative **inévitable**
  - Virgule **flottante** : signe, exposant, mantisse

# Introduction : Transistors et circuits logiques

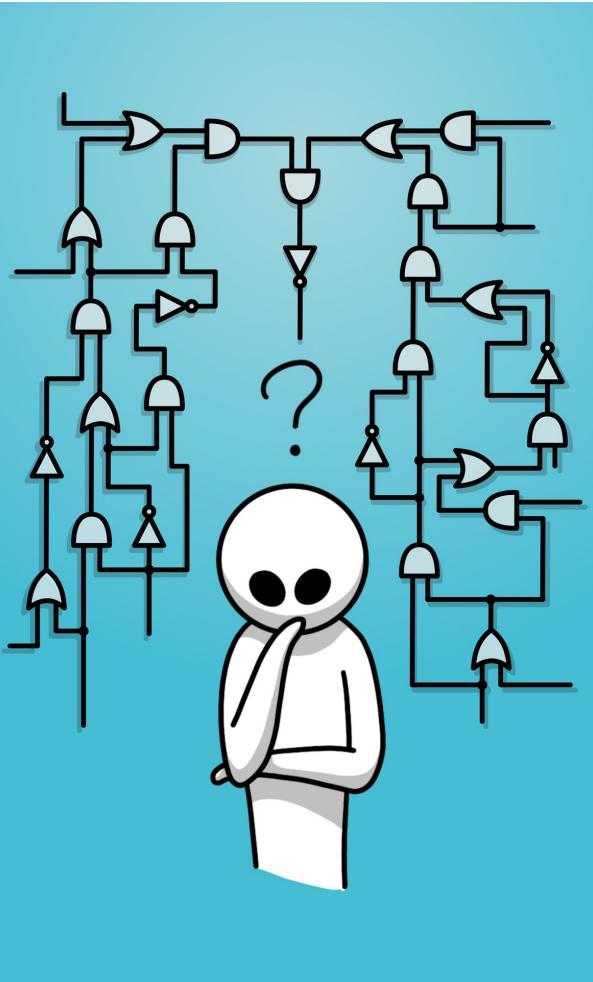


- **Le transistor** : brique élémentaire, agit comme un interrupteur pour représenter 0 et 1
- **Portes logiques** : associations de transistors pour effectuer des opérations binaires (ET, OU, NON...)
- **Processeurs** : assemblage géant de ces portes logiques (à l'échelle de milliards de transistors), réalisant calculs et contrôles

# Aujourd'hui

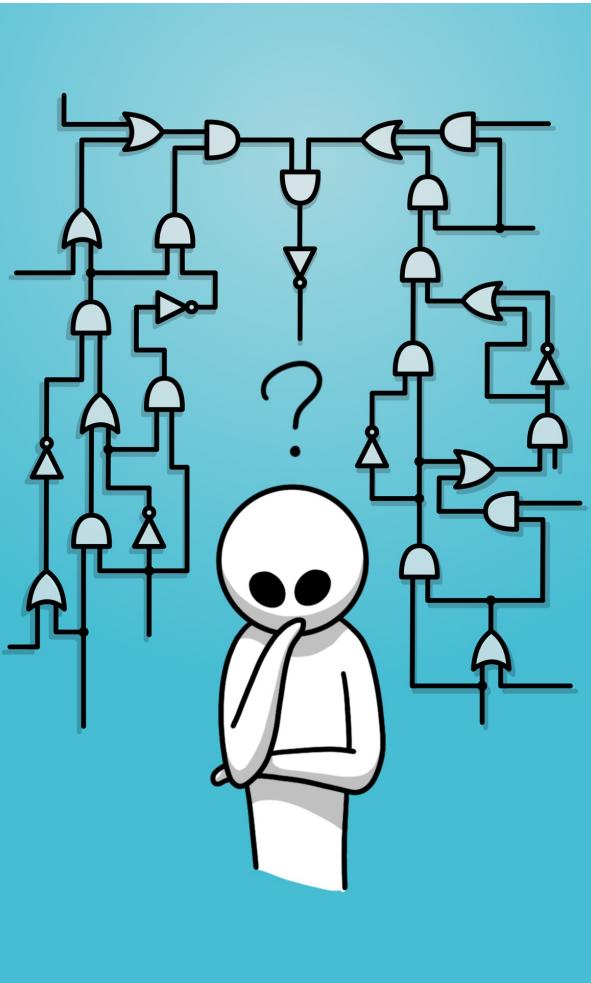
- Circuits logiques
- Transistors

# Introduction : Circuits logiques



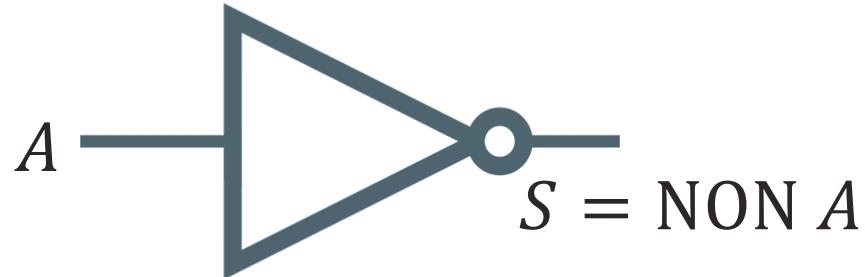
- Un circuit logique est un ensemble de **portes logiques** reliées entre elles.
- Ces portes logiques permettent de réaliser des **opérations élémentaires** sur des bits.
- Chaque porte logique est caractérisée par une **table de vérité** établissant une correspondance entre les entrées et les sorties de cette porte.

# Introduction : Circuits logiques

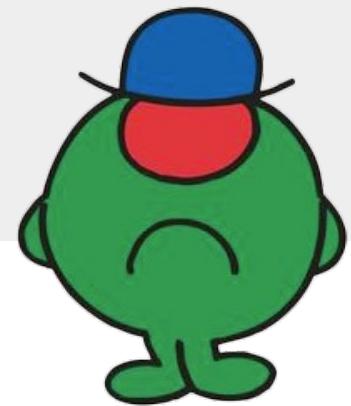


- Chaque porte logique est également représentée par un **symbole**.
- Nous verrons que l'on peut combiner plusieurs portes logiques ensemble pour faire tout type d'opération, comme un **additionneur**, par exemple.

## La porte NON (NOT)



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



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

## La porte ET (AND)



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

- Elle comporte **deux ou plusieurs** entrées.
- La porte ET génère un 1 en sortie si et seulement si **tous les 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  $A \cdot B$ .

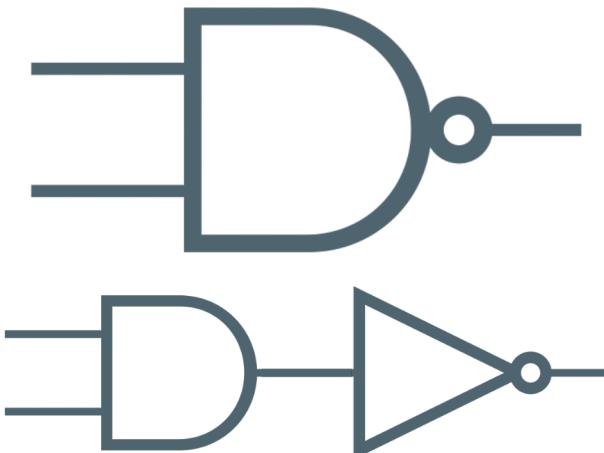
## La porte OU (OR)



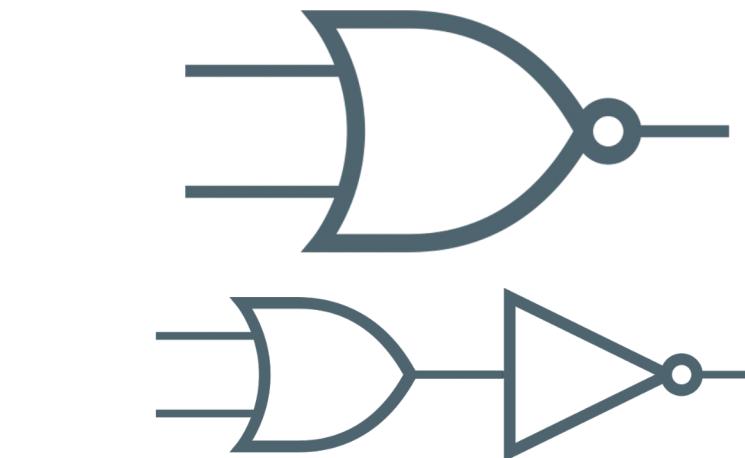
A	B	$S = A \text{ OR } B$
0	0	0
0	1	1
1	0	1
1	1	1

- 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 tous les 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  $A+B$ ).

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

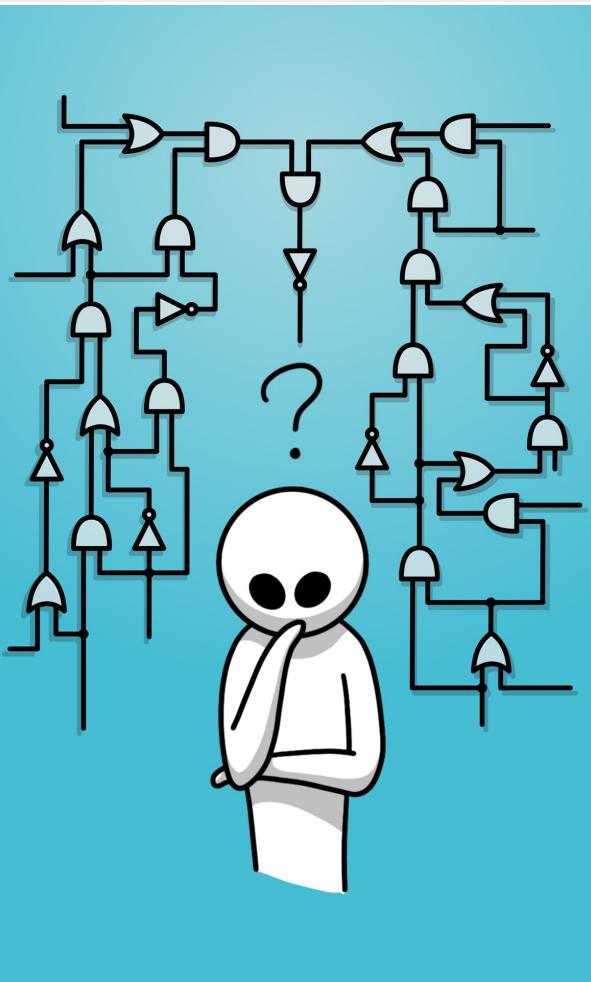


A	B	$S = A \text{ NAND } B$
0	0	1
0	1	1
1	0	1
1	1	0



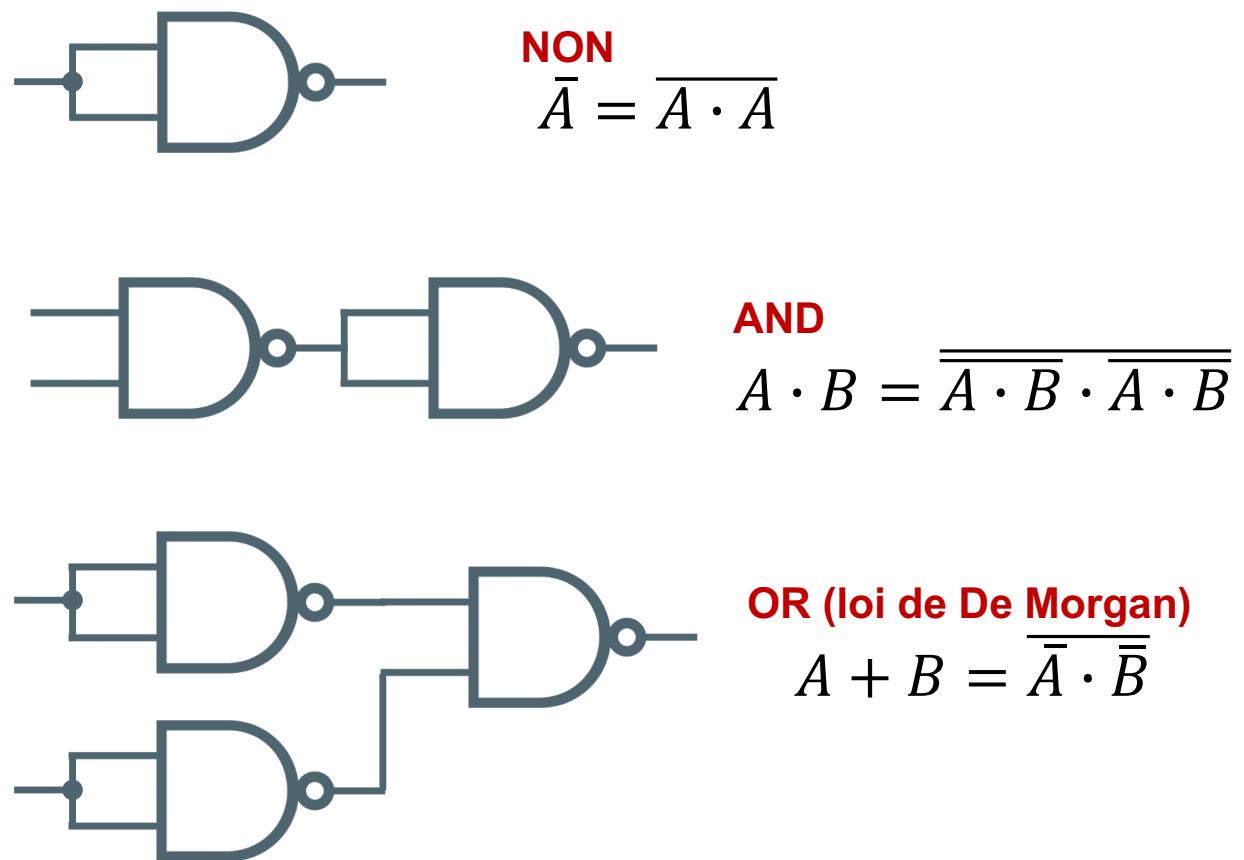
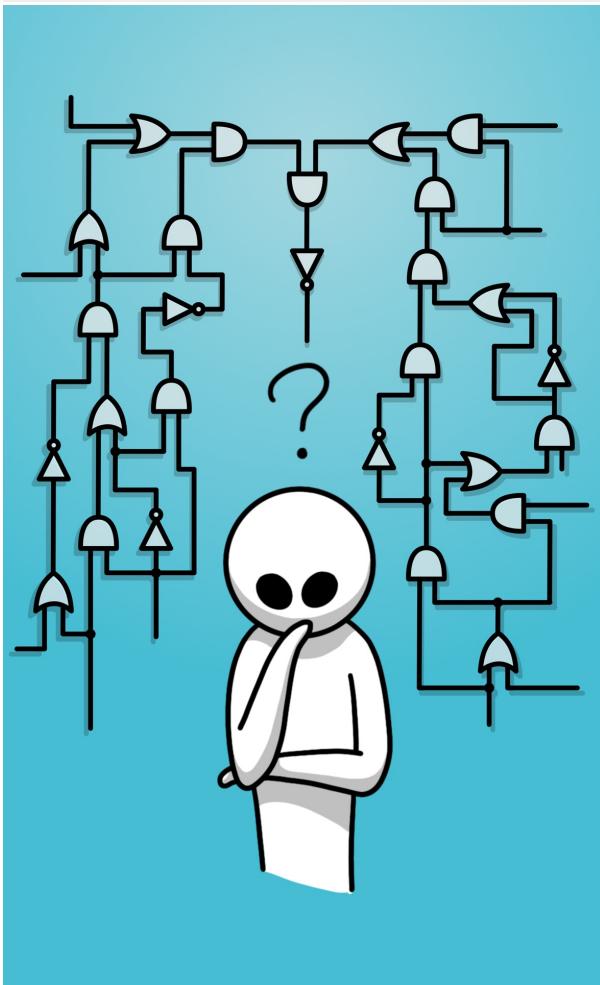
A	B	$S = A \text{ NOR } B$
0	0	1
0	1	0
1	0	0
1	1	0

## En pratique

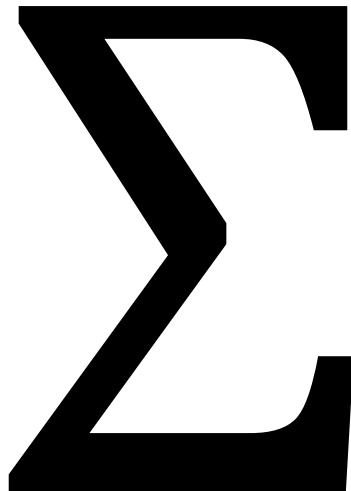


- Avec les **trois portes de base (NON, ET, OU)**, on peut créer tous les circuits possibles et donc effectuer toutes les opérations possibles.
- Il est possible de représenter une porte logique comme étant la **composition d'autres portes logiques**.
- En électronique, la porte **NON ET** est la plus 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.

## En pratique



# Additionner deux bits (sans retenue)



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

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

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

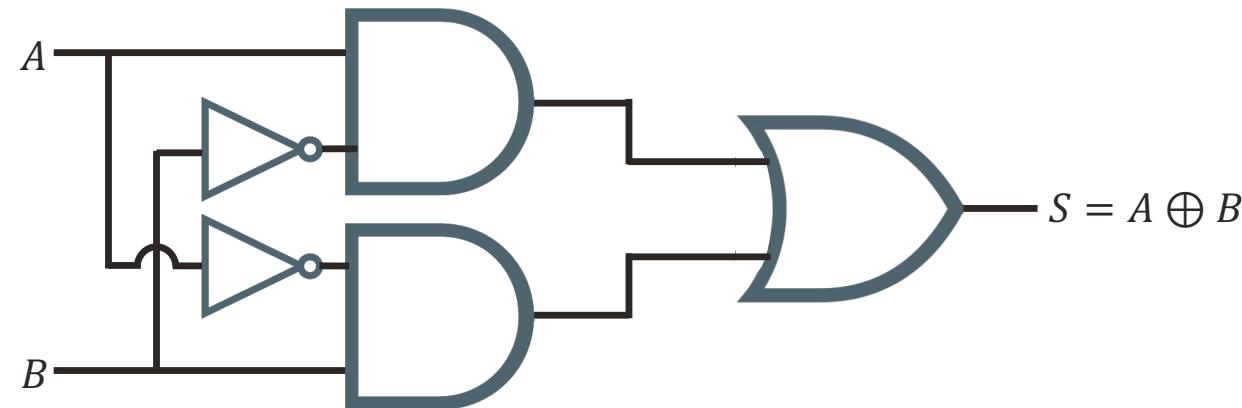
- Autrement dit:

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

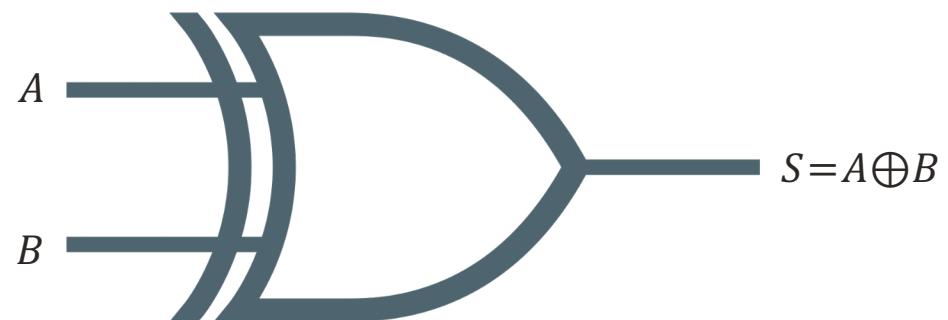
A	B	$(A+B)\%2$
0	0	0
0	1	1
1	0	1
1	1	0

## Additionner deux bits (sans retenue)

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



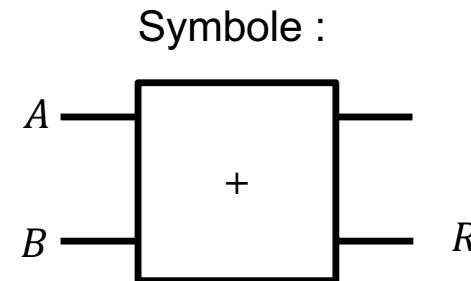
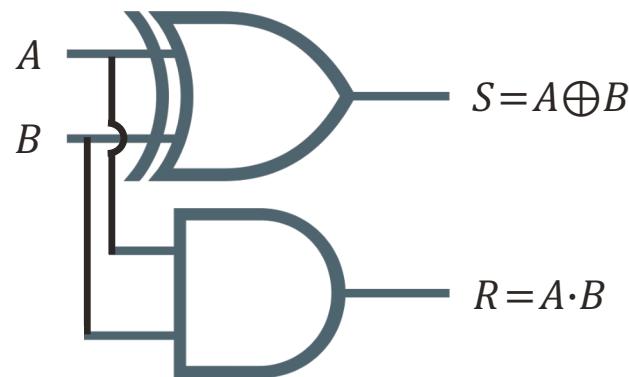
# La porte OU Exclusif (XOR)



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

## Additionner 2 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  $B=1$ .



## ICC-T 06 : Op. binaires : addition et soustraction

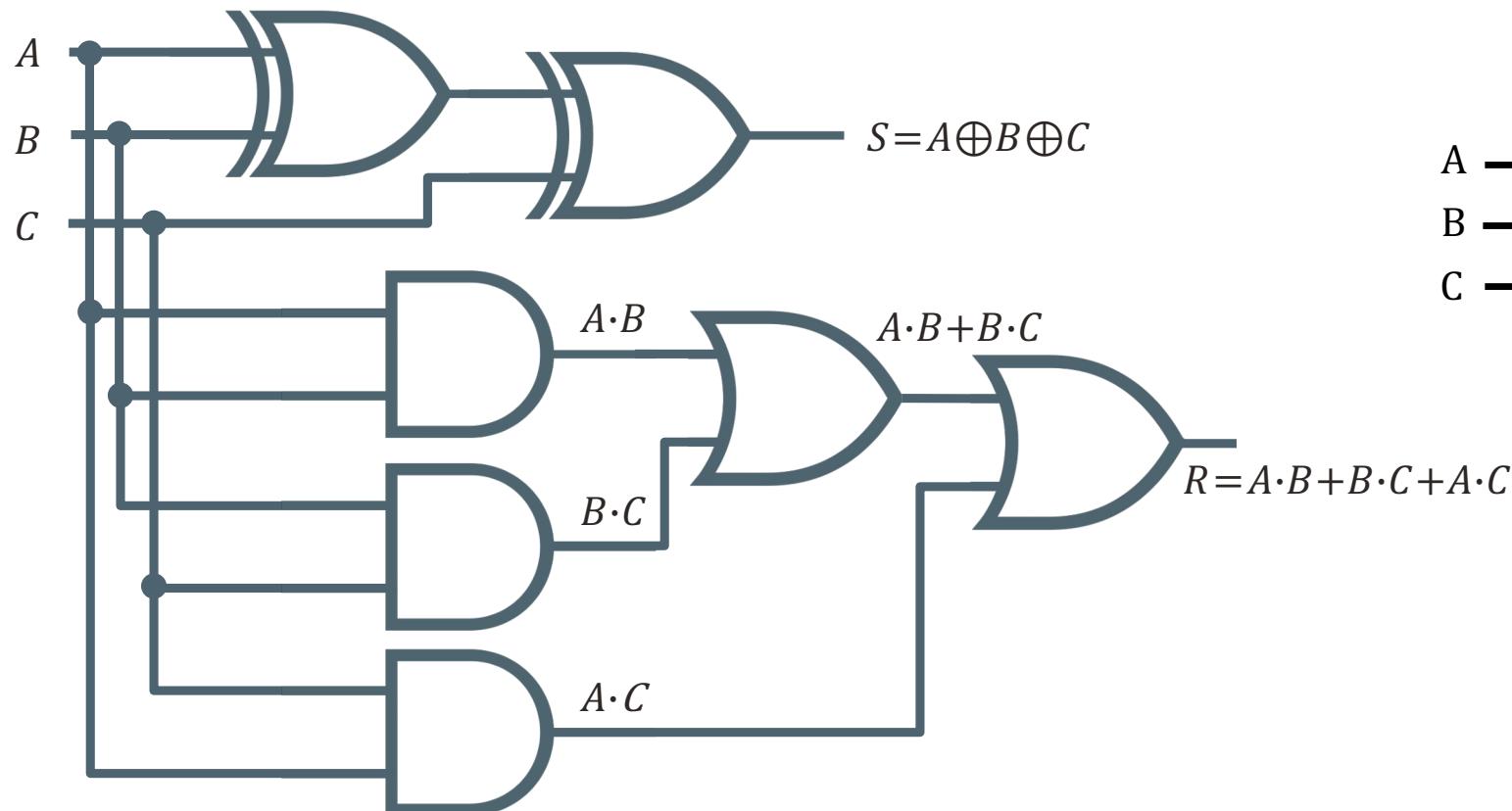
- Addition :

$$\begin{array}{r} 111 \\ 0111 \\ + 0011 \\ \hline 1010 \end{array} \quad \begin{array}{r} 7 \\ 3 \\ \hline 10 \end{array}$$

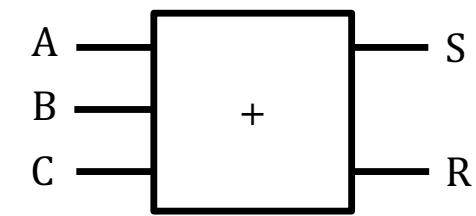


$0+0=0$	$0+1=1$
$1+1=10$	$1+1+1=11$

# Additionner 3 bits (avec retenue)



Symbol :



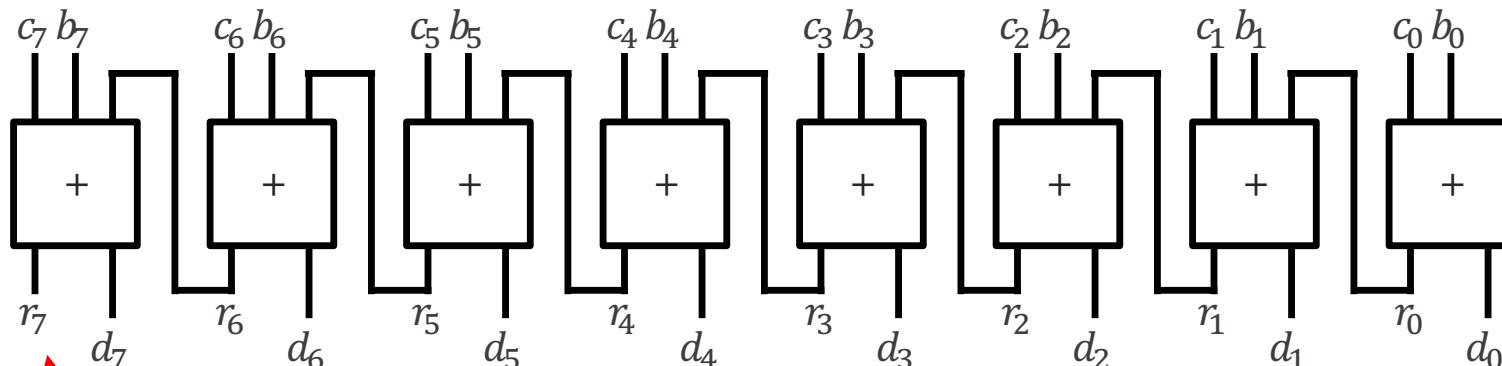
# Additionneur sur 8 bits

Effectuer :

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

Exemple :

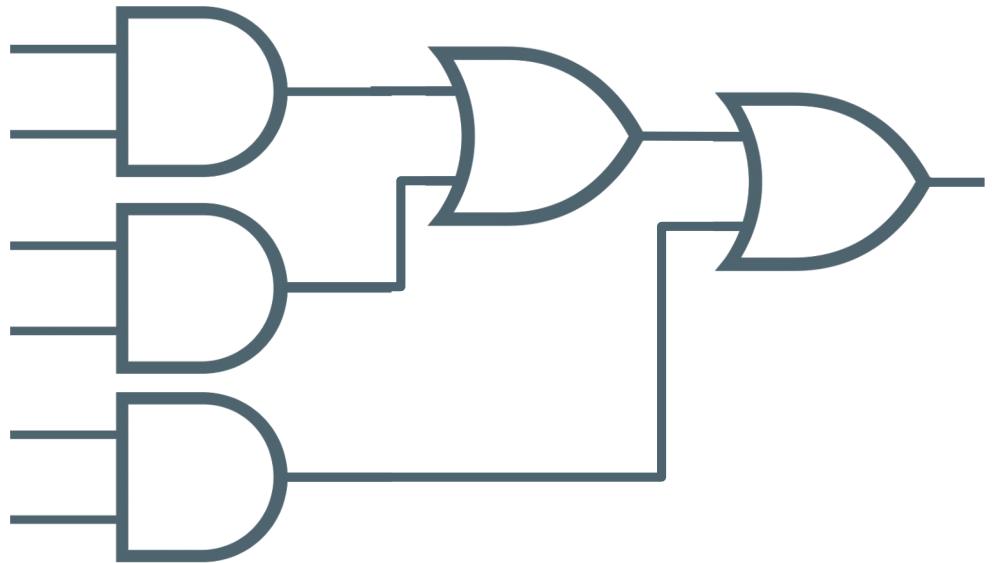
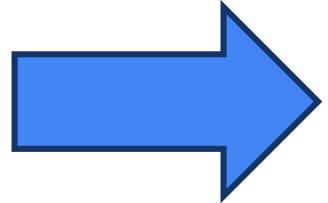
$$\begin{array}{r} 111 \ 11 \\ 00111001 \\ + 00101011 \\ \hline = 01100100 \end{array}$$



Si  $r_7 = 1 \Rightarrow \text{overflow !}$

# Carte de Karnaugh : de la table au circuit

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

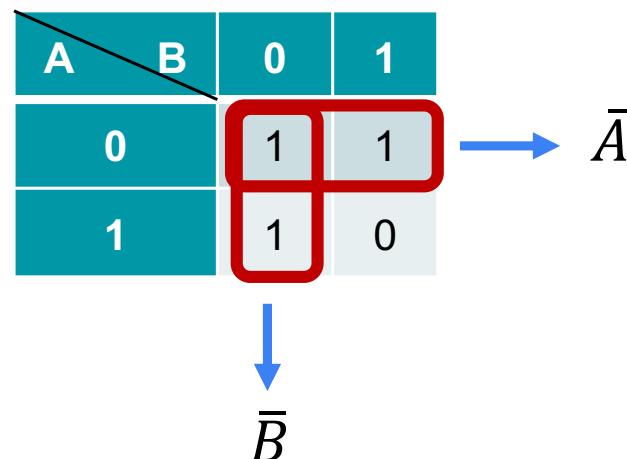


# Carte de Karnaugh : de la table au circuit

- Table de vérité

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

- K-Map



$$F = \bar{A} + \bar{B}$$

- Circuit

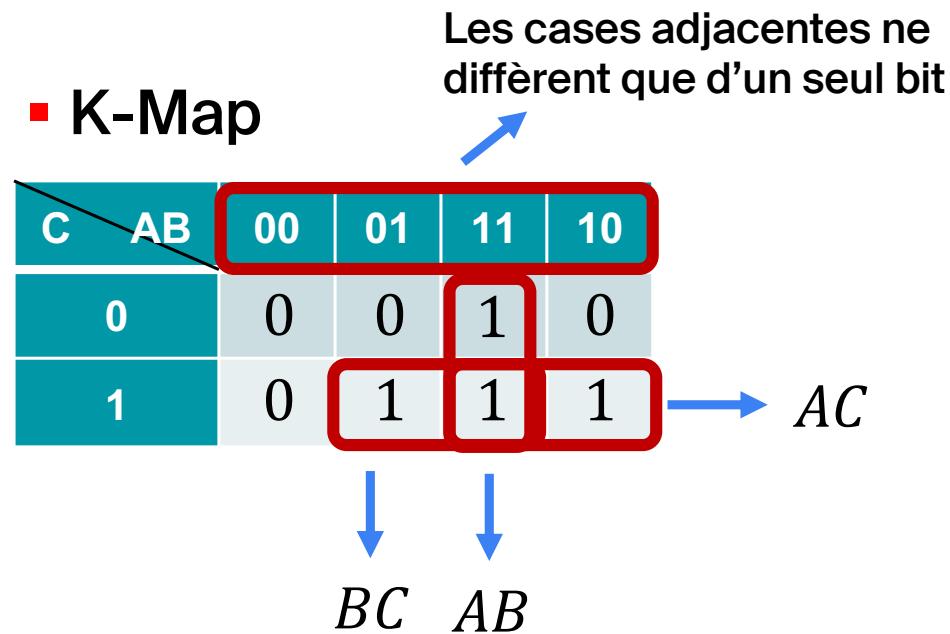


# Carte de Karnaugh : de la table au circuit

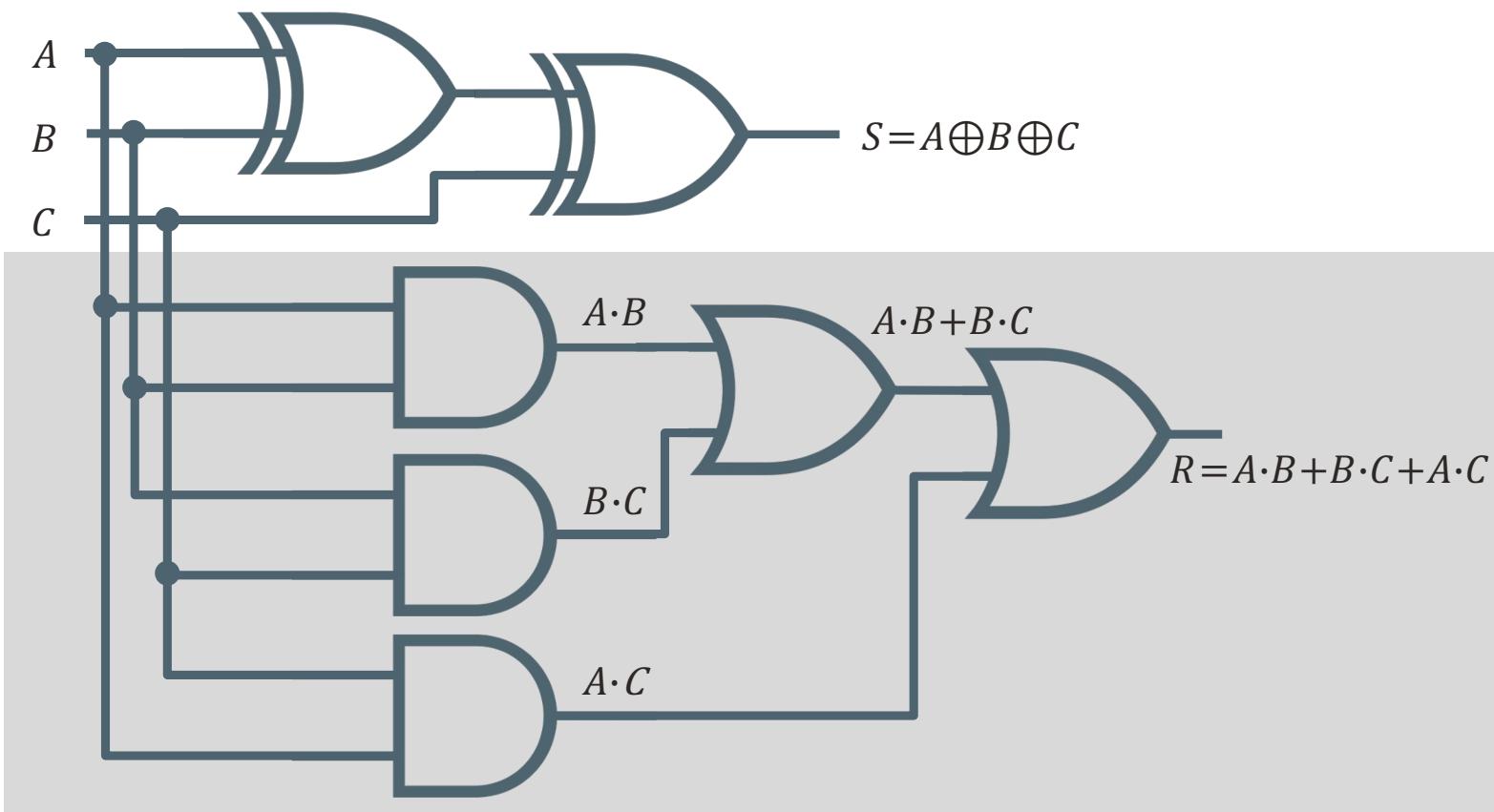
## ■ Table de vérité

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

## ■ K-Map



# Carte de Karnaugh : de la table au circuit



# Carte de Karnaugh : de la table au circuit

- 1.** Repérer le nombre de variables (n).
- 2.** Choisir la taille de la K-Map :
  - $2 \times 2$  (pour  $n=2$ )
  - $2 \times 4$  (pour  $n=3$ )
  - $4 \times 4$  (pour  $n=4$ )
- 3.** Placer les variables en lignes et colonnes.
- 4.** Écrire les indices en code Gray :
  - Pour 2 bits :  $00 \rightarrow 01 \rightarrow 11 \rightarrow 10$ .
- 5.** Remplir la K-Map avec les sorties (0 ou 1) selon la table de vérité.
- 6.** Faire les groupements de cases voisines pour la simplification.

# Mémoire vive

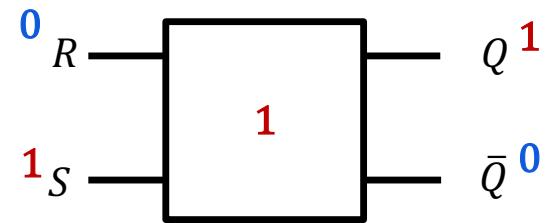
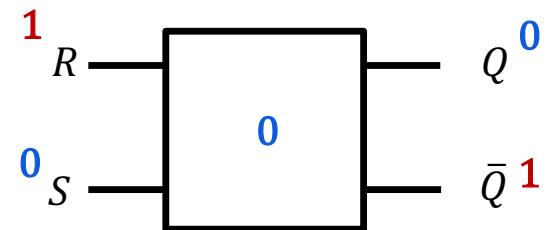
Jusqu'ici...

- Nous avons étudié des circuits **combinatoires** : leur sortie dépend uniquement des entrées à l'instant donné.
- Aucune **mémoire** : impossible de « retenir » un état.

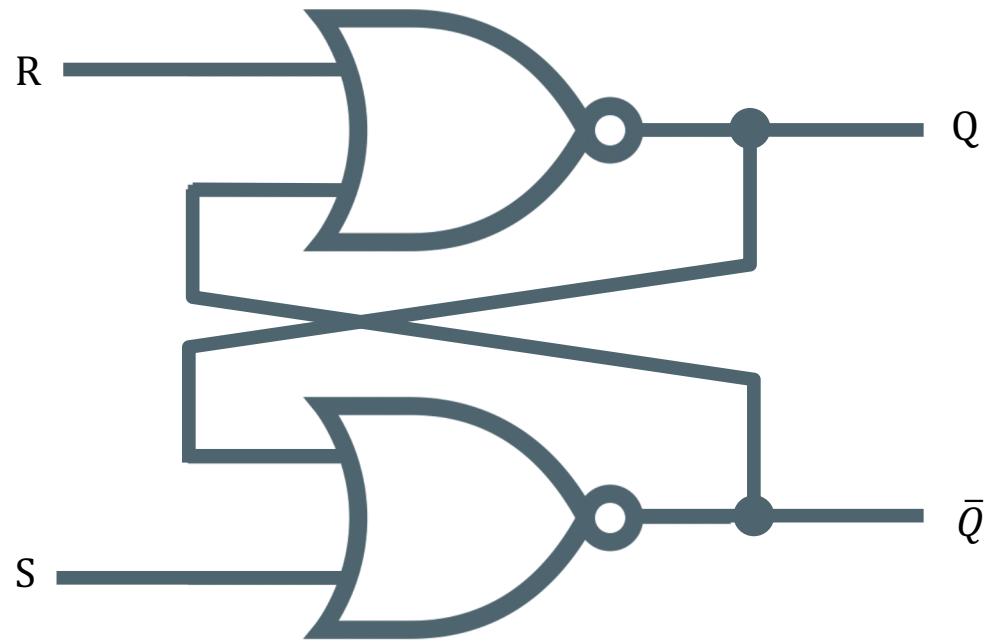
Et maintenant...

- Nous avons parfois besoin de conserver un bit de donnée (pour une commande, un signal d'erreur, etc.).
- Pour cela, on utilise des circuits **séquentiels**, c'est-à-dire avec mémoire.

# Mémoire vive : SR Latch

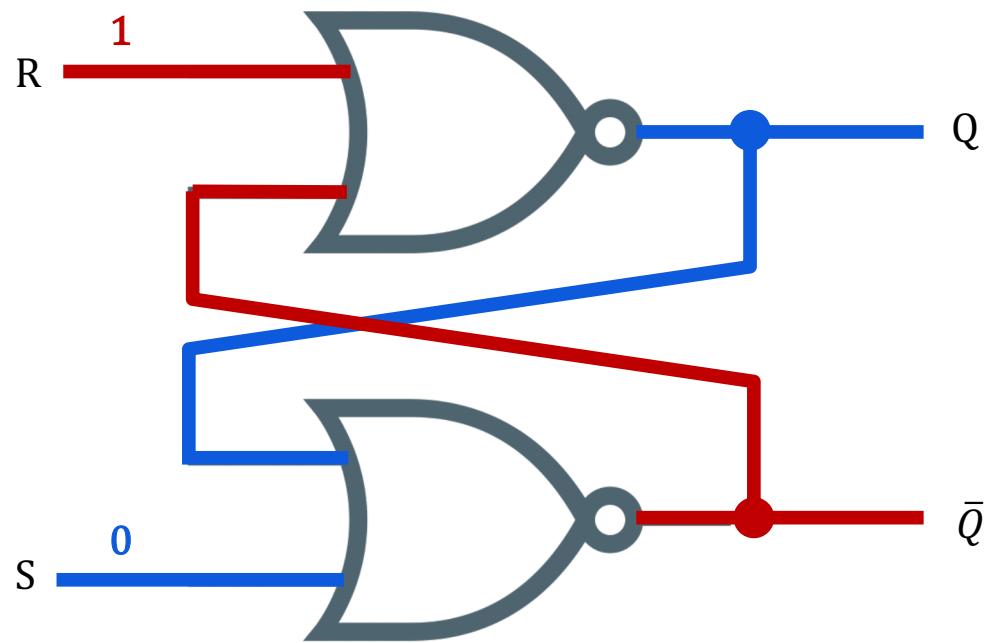


# Mémoire vive : SR Latch



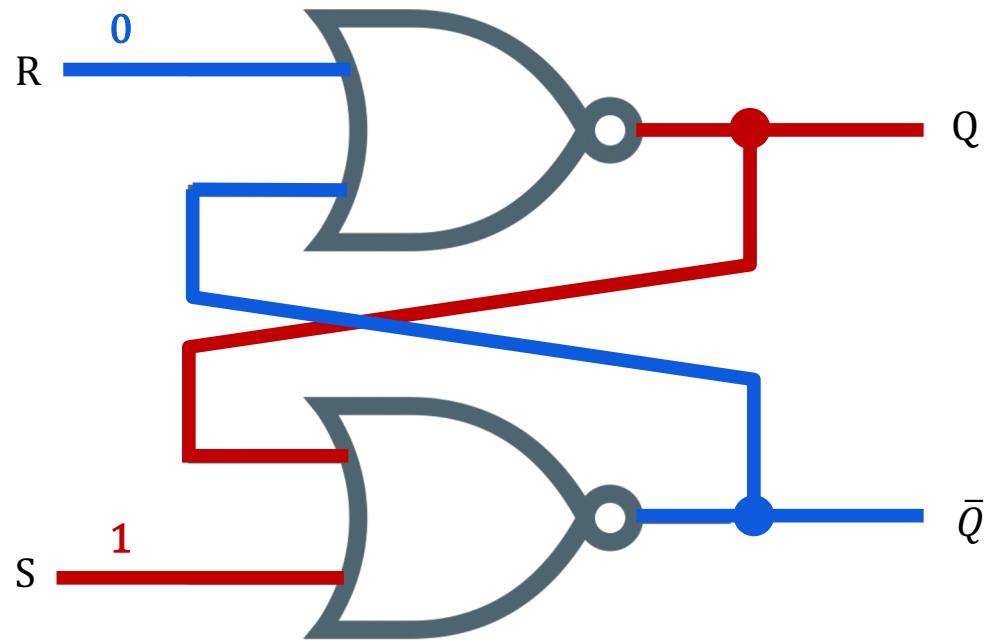
S	R	Q	$\bar{Q}$
0	0	1	0
		0	1
0	1	0	1
1	0	1	0
1	1	indéfini	

# Mémoire vive : SR Latch



S	R	Q	$\bar{Q}$
0	0	1	0
		0	1
0	1	0	1
1	0	1	0
1	1	indéfini	

# Mémoire vive : SR Latch

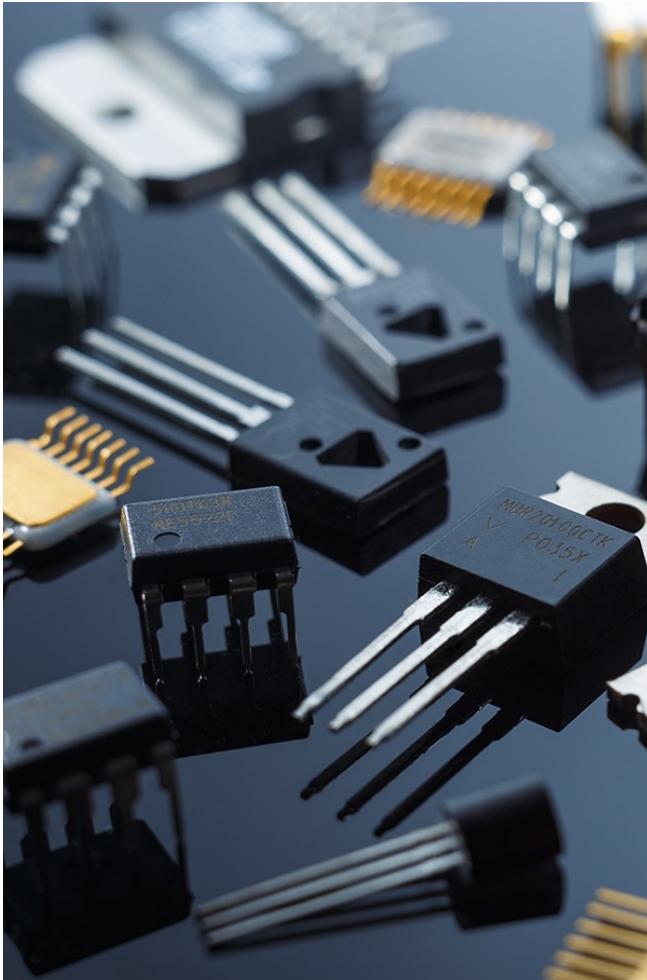


S	R	Q	$\bar{Q}$
0	0	1	0
		0	1
0	1	0	1
1	0	1	0
1	1	indéfini	

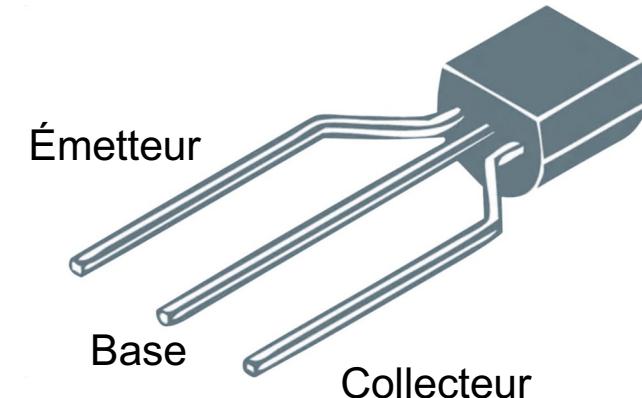
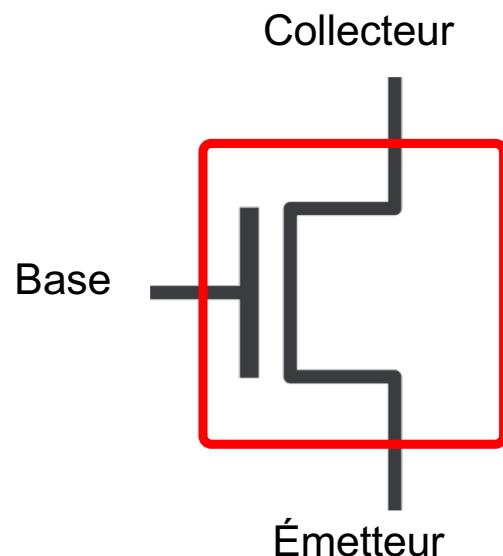
# Aujourd'hui

- Circuits logiques
- Transistors

# Le transistor



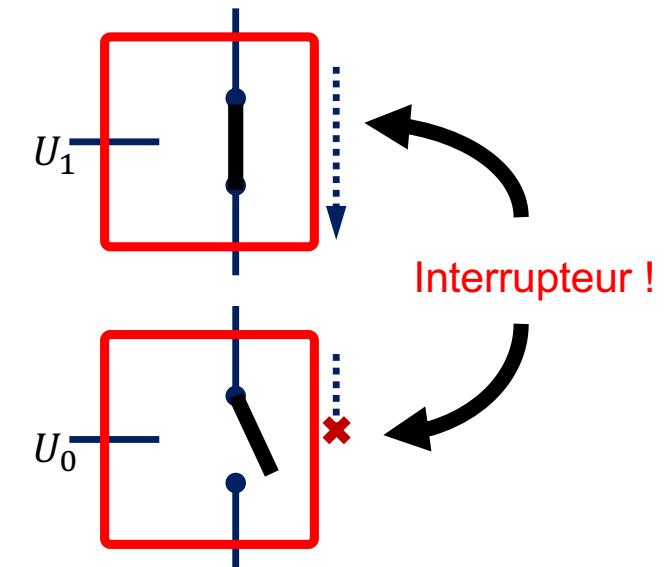
- Inventé en 1947 par trois américains : Bardeen, Shockley & Brattain
- Ce composant, qui est à la base de toute l'électronique moderne, a remplacé avantageusement les **relais électromécaniques** et les **tubes à vide** utilisés dans les premiers ordinateurs à la même époque → **miniaturisation**



# Principe de fonctionnement (n-mos)

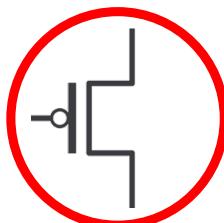


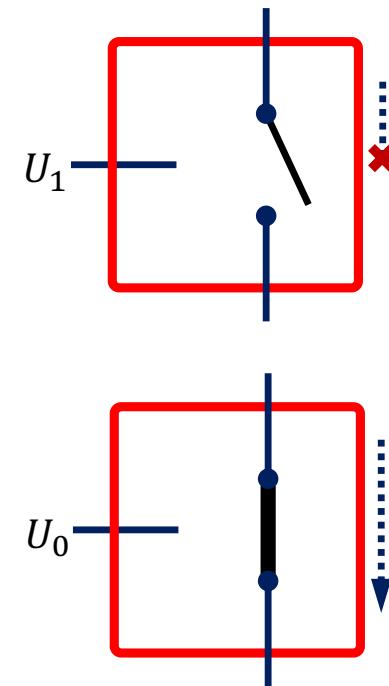
- Symbole :
- Si la tension à la base est **haute** ( $U_1=5V$ ) alors **le courant passe entre l'émetteur et le collecteur** :
- Si la tension à la base est **basse** ( $U_0=0V$ ) alors **le courant ne passe pas entre l'émetteur et le collecteur** :



# Principe de fonctionnement (p-mos)



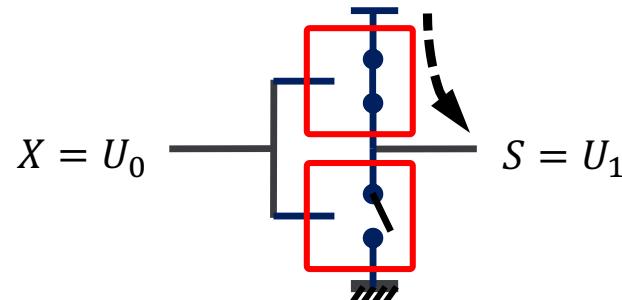
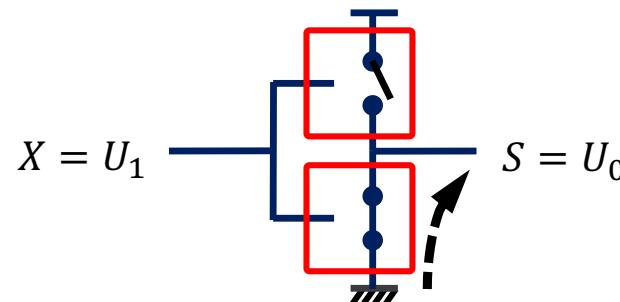
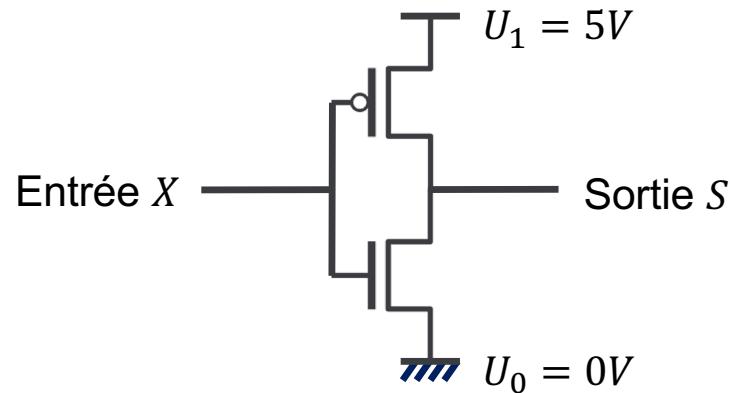
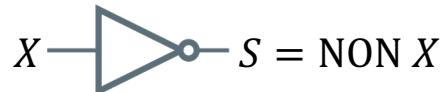
- Symbole : 
- Si la tension à la base est **haute** ( $U_1=5V$ ) alors **le courant ne passe pas** entre l'émetteur et le collecteur :
- Si la tension à la base est **basse** ( $U_0=0V$ ) alors **le courant passe** entre l'émetteur et le collecteur :



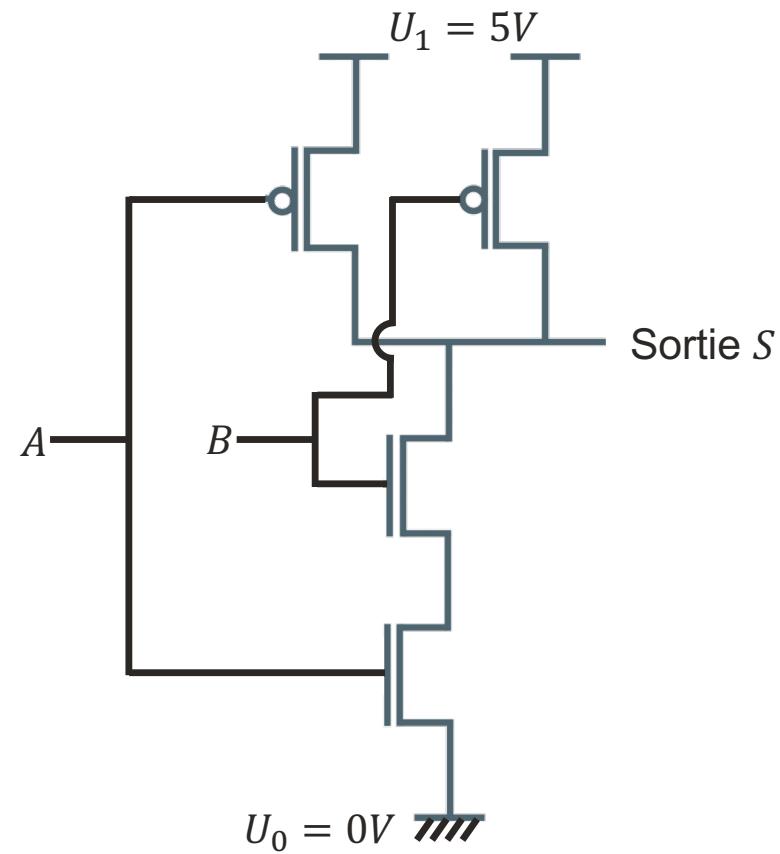
# Création d'un inverseur



- 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 de la porte NAND



# Aujourd'hui

- Circuits logiques
- Transistors

## Résumé Cours 7 – ICC-T

- Les **transistors** n-mos et p-mos forment la base de l'électronique numérique en se comportant comme des **interrupteurs** commandés par la tension de grille.
- Les **portes logiques** (telles que l'inverseur ou la porte NAND) s'obtiennent en **combinant ces transistors**, et constituent les briques essentielles du calcul binaire.
- À partir de la **table de vérité**, on utilise les **cartes de Karnaugh** pour repérer les groupements de 1 et simplifier les fonctions logiques, définissant ainsi plus efficacement le circuit souhaité.
- Un **additionneur 8 bits** repose sur la **porte XOR** pour effectuer la somme binaire, et sur d'autres portes (AND, OR ...) pour gérer la retenue.

rafael.pires@epfl.ch

**EPFL**

Merci

