



# **ARCHITECTURE DES ORDINATEURS (ADO)**

## **COURS 2 ALGÈBRE DE BOOLE CIRCUITS LOGIQUES**

**MAROUA MASMOUDI KOTTI**

**Avec les contributions de : TAISA GUIDINI GONCALVES**

# Algèbre de Boole

- 1. OPÉRATEURS LOGIQUES**
- 2. THÉORÈMES DE BOOLE**

# Algèbre de Boole

- L'information est manipulée sous la forme ***codée en binaire***.
- Les variables manipulées n'ont pas une signification numérique, mais ***logique*** et ne peuvent prendre que deux états dits ***logiques*** : 0 et 1 (true et false dans certains langages de programmation).
- Pour transformer l'information codée en binaire, on va utiliser les outils offerts par ***l'algèbre booléenne***.

- Exemple

Un test exécuté dans une condition est une variable booléenne.

SI (A=B) ALORS ...

Le test (A=B) peut avoir deux valeurs :

- 1 si A est réellement égal à B ;
- et 0 sinon.

On dit que c'est la variable booléenne associée au test A=B.

# Algèbre de Boole

- L'algèbre booléenne définit un certain nombre :
  - **d'opérations** pour construire des expressions booléennes ;
  - **de propriétés** pour manipuler ces expressions.
- Une expression booléenne est représentée par une **fonction booléenne (logique)**.
- Une fonction booléenne (ou logique) est une fonction définie sur  **$2^n$  combinaisons de  $n$  variables** logiques.
  - Une fonction logique est donc une fonction de  **$n$  variables logiques**.
  - Une fonction logique peut prendre en sortie **2 valeurs notées 0 et 1**.

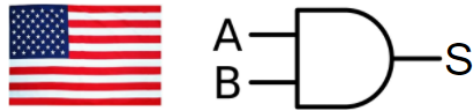
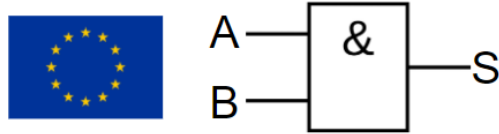
# Algèbre de Boole

- **Exemple :**

Une lampe possède 2 états : **allumée "1"** ou **éteinte "0"**. Cet état est la fonction de la position **fermée "1"** ou **ouverte "0"** des différents interrupteurs, a, b et c.

- La fonction logique est allumer ou éteindre.
  - Les interrupteurs sont les variables logiques.
  - Le résultat de la fonction logique est l'état de la lampe, qui possède bien 2 valeurs : **allumée "1"** ou **éteinte "0"**.
- La définition la plus simple d'une fonction booléenne est sa **table de vérité**. Elle donne pour chaque valeur possible, des entrées et les valeurs de sortie correspondantes.

# Algèbre de Boole



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

$$S = A . B$$

- Dans une technologie donnée, on sait implémenter certaines opérations de l'algèbre par des opérateurs, appelés des **portes logiques**.
- En assemblant ces portes, on peut construire des **circuits logiques** qui implémentent une fonction booléenne donnée.

# Opérateurs / Portes Logiques

---

- L'algèbre de Boole utilise plusieurs opérateurs (ET, OU, etc.)
- Chaque porte logique est représentée par :
  - son symbole ;
  - sa table de vérité (correspondance entre la sortie et toutes les combinaisons de valeurs que peuvent prendre la/les entrée(s)) ;
  - son circuit logique (schéma électrique).
- Une **table de vérité** est un tableau qui représente des entrées (en colonne) et des états binaires (0 et 1). Le résultat (la sortie), exprimé aussi sous forme binaire, en général, se lit dans la dernière colonne.

# Opérateurs / Portes Logiques

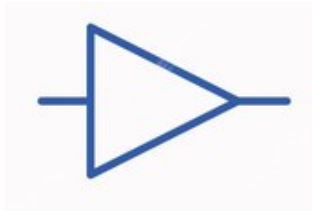
Entrées logiques					Sortie
$e_1$	$e_2$	$e_3$	...	$e_n$	$s$
0	0	0	...	0	0
0	0	0	...	1	1
...	...	...	...	...	...
1	1	1	....	0	1
1	1	1	....	1	0

Combinaisons d'entrées

Pour **n entrées**, il y a  **$2^n$  combinaisons** différentes d'entrées, soit  **$2^n$  lignes** dans la table de vérité.

# Opérateurs / Portes Logiques

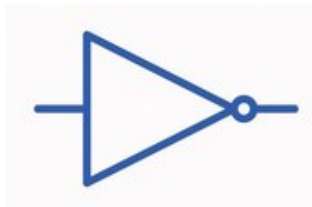
- **L'opérateur OUI** est symbolisé par la variable d'entrée.



**Equation :  $s = a$**

<i>Entrée</i>	<i>Sortie</i>
$a$	$s = a$
0	<b>0</b>
1	<b>1</b>

- **L'opérateur NON** est symbolisé par une barre au dessus de(s) variable(s). En maths on utilise le symbole  $\neg$



**Equation :  $s = \bar{a}$**

<i>Entrée</i>	<i>Sortie</i>
$a$	$s = \bar{a}$
0	<b>1</b>
1	<b>0</b>

# Opérateurs / Portes Logiques

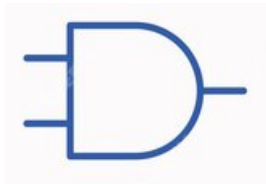
- **L'opérateur ET** est symbolisé par un AND ou &&.

En algèbre de Boole, il est symbolisée par un point "."

En maths on utilise le symbole  $\wedge$

La sortie vaut 1 si TOUTES les entrées valent 1.

La sortie vaut 0 si AU MOINS une entrée vaut 0.



**Equation :  $s = a . b$**

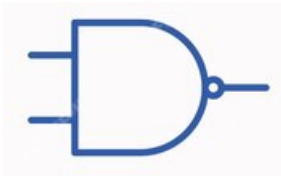
<i>Entrées</i>		<i>Sortie</i>
<i>a</i>	<i>b</i>	<i>s = a . b</i>
0	0	<b>0</b>
0	1	<b>0</b>
1	0	<b>0</b>
1	1	<b>1</b>

# Opérateurs / Portes Logiques

## ● L'opérateur NON-ET

La sortie vaut 1 si AU MOINS une entrée vaut 0.

La sortie vaut 0 si TOUTES les entrées valent 1.



$$\text{Equation : } s = \overline{a \cdot b}$$

<i>Entrées</i>		<i>Sortie</i>
<i>a</i>	<i>b</i>	$s = \overline{a \cdot b}$
0	0	<b>1</b>
0	1	<b>1</b>
1	0	<b>1</b>
1	1	<b>0</b>

# Opérateurs / Portes Logiques

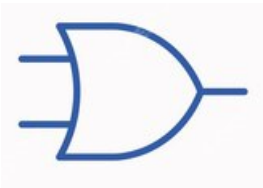
- **L'opérateur OU** est symbolisé par un OR ou  $||$ .

En algèbre de Boole, il est symbolisée par un "+"

En maths on utilise le symbole  $\vee$

La sortie vaut 1 si AU MOINS une entrée vaut 1.

La sortie vaut 0 si TOUTES les entrées valent 0.



**Equation :  $s = a + b$**

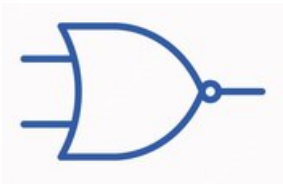
<i>Entrées</i>		<i>Sortie</i>
<i>a</i>	<i>b</i>	$s = a + b$
0	0	<b>0</b>
0	1	<b>1</b>
1	0	<b>1</b>
1	1	<b>1</b>

# Opérateurs / Portes Logiques

- **L'opérateur NON-OU**

La sortie vaut 0 si AU MOINS une entrée vaut 1.

La sortie vaut 1 si TOUTES les entrées valent 0.



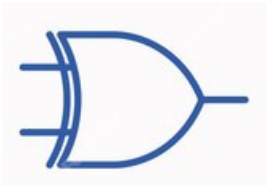
**Equation :  $s = \overline{a + b}$**

<i>Entrées</i>		<i>Sortie</i>
<i>a</i>	<i>b</i>	$s = \overline{a + b}$
0	0	<b>1</b>
0	1	<b>0</b>
1	0	<b>0</b>
1	1	<b>0</b>

# Opérateurs / Portes Logiques

- **L'opérateur XOR/OU EXCLUSIF**

La sortie vaut 1 si UNE SEULE entrée vaut 1.



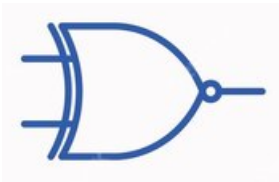
**Equation :  $s = a \oplus b$**

<i>Entrées</i>		<i>Sortie</i>
<i>a</i>	<i>b</i>	$s = a \oplus b$
0	0	<b>0</b>
0	1	<b>1</b>
1	0	<b>1</b>
1	1	<b>0</b>

# Opérateurs / Portes Logiques

- **L'opérateur XNOR/NON-XOR**

La sortie vaut 1 si LES DEUX entrées sont identiques.



**Equation :**  $s = \overline{a \oplus b}$

<i>Entrées</i>		<i>Sortie</i>
<i>a</i>	<i>b</i>	$s = \overline{a \oplus b}$
0	0	<b>1</b>
0	1	<b>0</b>
1	0	<b>0</b>
1	1	<b>1</b>

# Relations et propriétés

<b>propriétés de la somme</b>		<b>propriétés du produit</b>		<b>négation</b>
$0 + 0 = 0$	$a + 1 = 1$	$0 . 0 = 0$	$a . 1 = a$	$\overline{0} = 1$
$0 + 1 = 1$	$a + 0 = a$	$0 . 1 = 0$	$a . 0 = 0$	$\overline{1} = 0$
$1 + 0 = 1$	$a + a = a$	$1 . 0 = 0$	$a . a = a$	$\overline{\overline{a}} = a$
$1 + 1 = 1$	$a + \overline{a} = 1$	$1 . 1 = 1$	$a . \overline{a} = 0$	

<b>commutativité</b>	<b>associativité</b>	<b>distributivité</b>
$a . b = b . a$ $a + b = b + a$	$a . (b . c) = (a . b) . c$ $a + (b + c) = (a + b) + c$	$a . (b + c) = a . b + a . c$ $(a + b) . (c + d) = ac + ad + bc + bd$

<b>propriétés combinées</b>		<b>théorème de Morgan</b>
$a . (a + b) = a$	$(a + b) . (a + \overline{b}) = a$	$\overline{a + b + c} = \overline{a} . \overline{b} . \overline{c}$
$a + a . b = a$	$(a + b) . (a + c) = a + b . c$	$\overline{a . b . c} = \overline{a} + \overline{b} + \overline{c}$
$a + \overline{a} . b = a + b$	$(a + b) . (\overline{a} + c) = a . c + \overline{a} . b$	

# Théorèmes de l'Algèbre de Boole

- Théorème d'involution :  $\overline{\overline{A}} = A$   
 $\overline{\overline{\overline{A}}} = \overline{A}$

- Théorème d'inclusion :  $A.B + A.\overline{B} = A$   
 $[A + B].[A + \overline{B}] = A$

Démonstration : mettre A en facteur [distributivité « à l'envers »] :

$$A.B + A.\overline{B} = A.[B + \overline{B}] = A$$

$$[A + B].[A + \overline{B}] = A + B.\overline{B} = A$$

- **Théorème d'allégement** :  **$A.(\overline{A} + B) = A.B$**   
 **$A + \overline{A}.B = A + B$**

Démonstration : utiliser la distributivité [du ET et du OU] :

$$A.(\overline{A} + B) = A.\overline{A} + A.B = A.B$$

$$A + \overline{A}.B = [A + \overline{A}].[A + B] = A + B$$

# Théorèmes de l'Algèbre de Boole

- **Théorème d'absorption :**  $A.(A + B) = A$   
 $A + (A.B) = A$

Démonstration par la distributivité du ET (*utilisée dans les 2 sens*) :

$$\begin{aligned} A.(A + B) &= A.A + A.B \quad \text{[distributivité du ET]} \\ &= A + A.B \quad \text{[2<sup>ème</sup> forme du théorème d'absorption]} \\ &= A.(B + 1) \quad \text{[mise en facteur de A : distributivité du ET « à l'envers »]} \\ &= A.1 \\ &= A \end{aligned}$$

Démonstration par la distributivité du OU (*utilisée dans les 2 sens*) :

$$\begin{aligned} A + A.B &= (A + A).(A + B) \quad \text{[distributivité du OU]} \\ &= A.(A + B) \quad \text{[1<sup>ère</sup> forme du théorème d'absorption]} \\ &= (A + 0).(A + B) \quad \text{[pour y voir plus clair dans ce qui va suivre ...]} \\ &= A + (B.0) \quad \text{[distributivité du OU à l'envers : « factorisation par l'addition »]} \\ &= A + 0 \\ &= A \end{aligned}$$

- **Théorème de De Morgan :**  $\overline{A.B} = \overline{A} + \overline{B} \rightarrow$  porte ET-NON  
 $\overline{A + B} = \overline{A} . \overline{B} \rightarrow$  porte OU-NON

# Fonction booléenne à partir de la table de vérité

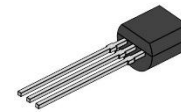
Entrées			Sortie
<i>S1</i>	<i>S2</i>	<i>S3</i>	<i>H</i>
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

$$H = \overline{S1} . \overline{S2} . \overline{S3} + \overline{S1} . S2 . \overline{S3} + S1 . S2 . S3$$

# TABLEAU DE KARNAUGH

# Tableau de Karnaugh : Simplifications de fonctions booléennes

- Les fonctions booléennes sont implantées à l'aide de **portes logiques** constituées de **transistors**.
- Afin d'économiser de l'espace, de l'énergie et de l'argent, l'objectif est d'utiliser le moins de transistors possibles.
- Nous utilisons les **théorèmes** et les **propriétés de l'algèbre de Boole**, afin de passer d'une expression à une autre plus simple.
- Une autre méthode très efficace est l'utilisation des **tableaux de Karnaugh**.



# Tableau de Karnaugh

- Méthode graphique et simple pour trouver ou simplifier une fonction logique.
- Elle utilise le **code de Gray** qui a comme propriété principale de ne faire varier qu'un seul bit entre deux variables (mots successifs).
- Chaque case du tableau correspond à une combinaison des variables d'entrées, donc à une ligne de la **table de vérité**.
- Le **tableau de Karnaugh** aura autant de cases que la table de vérité possède de lignes.

# Table de Karnaugh : procédure à suivre

---

- Construire la table de vérité.
- Construire la table de Karnaugh, en changeant un seul chiffre (une seule valeur de variable) entre les cases adjacentes.
- On remplit la table de Karnaugh avec les valeurs de la table de vérité.

# Tableau de Karnaugh

## Table de vérité

<i>Entrées</i>			<i>Sortie</i>
<i>S1</i>	<i>S2</i>	<i>S3</i>	<i>H</i>
0	0	0	<b>1</b>
0	0	1	<b>0</b>
0	1	0	<b>1</b>
0	1	1	<b>0</b>
1	0	0	<b>0</b>
1	0	1	<b>0</b>
1	1	0	<b>0</b>
1	1	1	<b>1</b>

## Tableau de Karnaugh

<i>S3</i> <i>S1 S2</i>	0	1
00	<b>1</b>	<b>0</b>
01	<b>1</b>	<b>0</b>
11	<b>0</b>	<b>1</b>
10	<b>0</b>	<b>0</b>

# Table de Karnaugh : procédure à suivre

---

- Ensuite, on cherche à recouvrir **tous les 1** du tableau en formant des regroupements. Chaque regroupement :
  - ne contient que des 1 adjacents ;
  - doit former un rectangle ;
  - et le nombre d'éléments d'un regroupement doit être une puissance de 2 ( $2^n$ , ... ,  $2^4 = 16$ ,  $2^3 = 8$ ,  $2^2 = 4$ ,  $2^1 = 2$ ,  $2^0 = 1$ ).
- On choisit toujours :
  - les plus grands regroupements possibles ;
  - et le plus petit nombre possible de regroupements recouvrant les 1 du tableaux.

# Table de Karnaugh: procédure à suivre

---

- Une même case peut faire partie de plusieurs regroupements.
- Les regroupements peuvent se faire au delà des bords. Les côtés/coins ont des codes Gray voisins.
- Toute case contenant un “1” doit faire partie d’au moins un regroupement, mais **aucun “0” ne doit y être.**

# Tableau de Karnaugh

<i>Entrées</i>			<i>Sortie</i>
<i>S1</i>	<i>S2</i>	<i>S3</i>	<i>H</i>
0	0	0	<b>1</b>
0	0	1	<b>0</b>
0	1	0	<b>1</b>
0	1	1	<b>0</b>
1	0	0	<b>0</b>
1	0	1	<b>0</b>
1	1	0	<b>0</b>
1	1	1	<b>1</b>

<i>S1 S2</i> \ <i>S3</i>	0	1
00	<b>1</b>	<b>0</b>
01	<b>1</b>	<b>0</b>
11	<b>0</b>	<b>1</b>
10	<b>0</b>	<b>0</b>

# Tableau de Karnaugh

AB \ CD		AB			
		00	01	11	10
CD	10	1	1	1	1
	11	1	1	1	
	01	1	1	1	1
	00	1		1	1

AB \ CD		AB			
		00	01	11	10
CD	10	1	1		1
	11	1	1		1
	01	1	1		1
	00	1	1		

Un tableau de Karnaugh peut se représenter sous les formes suivantes :

(X)		$\bar{a}$	$a$
$\bar{b}$			
$b$			

(X)		0	$a$	1
0				
1				

(X)		$a$	
$b$			

# Table de Karnaugh: procédure à suivre

---

- Pour chaque rectangle, on élimine les variables qui changent d'état. On conserve que les variables qui restent fixes (ne changent pas de valeur).
- On multiplie les variables fixes avec “.” afin d'obtenir des mintermes (des variables booléennes liées par des ET logique) de H.
- Les produits obtenus sont ensuite sommés avec “+” pour avoir la fonction booléenne de H.

# Tableau de Karnaugh

<i>Entrées</i>			<i>Sortie</i>
<i>S1</i>	<i>S2</i>	<i>S3</i>	<i>H</i>
0	0	0	<b>1</b>
0	0	1	<b>0</b>
0	1	0	<b>1</b>
0	1	1	<b>0</b>
1	0	0	<b>0</b>
1	0	1	<b>0</b>
1	1	0	<b>0</b>
1	1	1	<b>1</b>

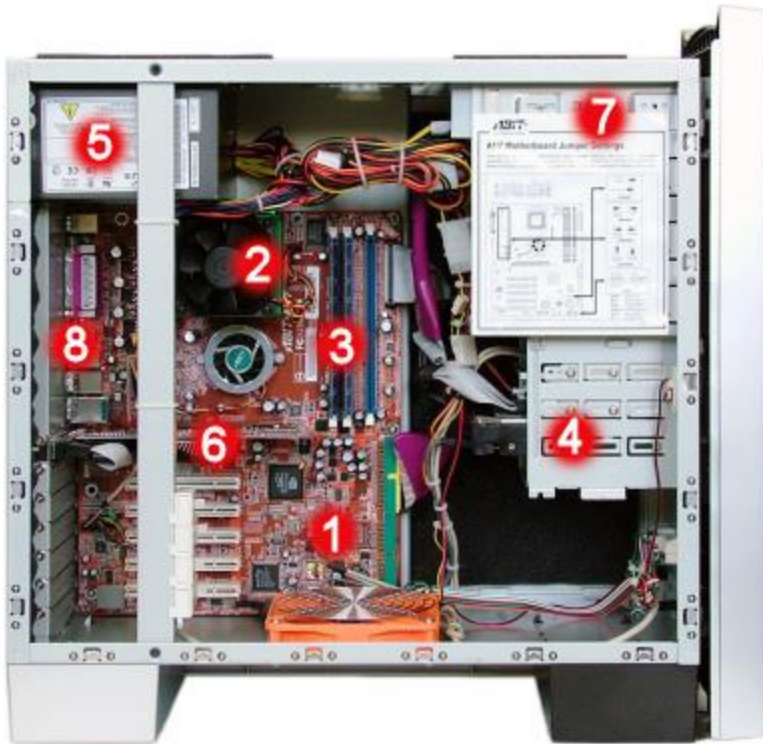
<i>S1 S2 \ S3</i>	0	1
00	<b>1</b>	<b>0</b>
01	<b>1</b>	<b>0</b>
11	<b>0</b>	<b>1</b>
10	<b>0</b>	<b>0</b>

$$H = \overline{S1} . \overline{S3} + S1 . S2 . S3$$

# Circuit Logique

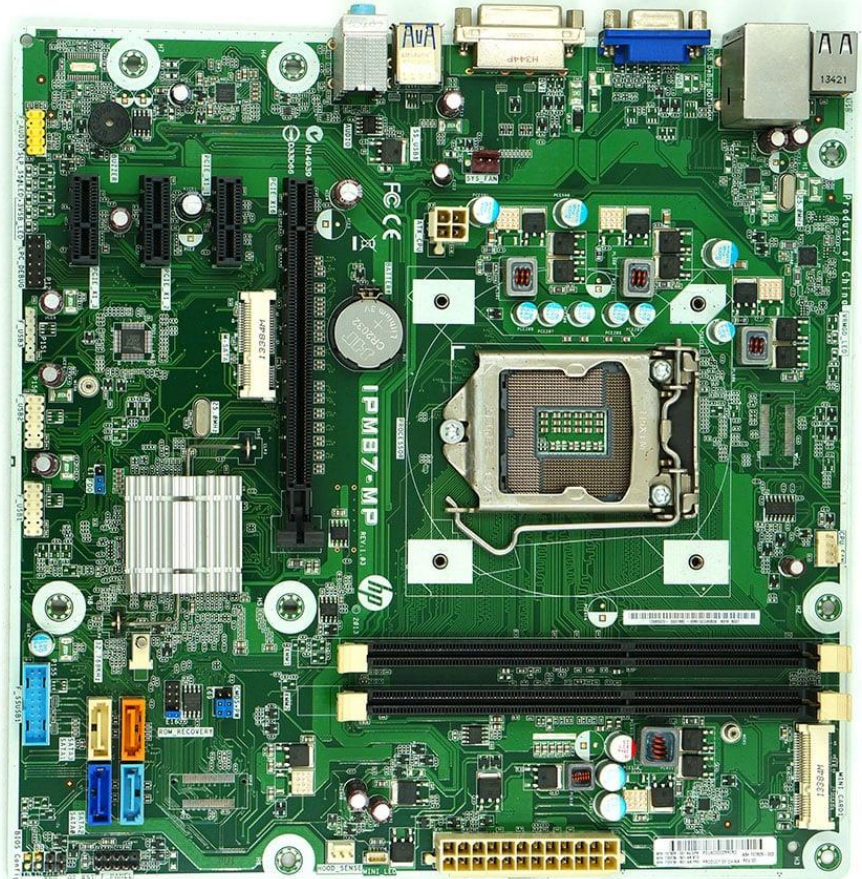
1. **CIRCUIT COMBINATOIRE**
2. **CIRCUIT SÉQUENTIEL**

# Circuit logique dans l'ordinateur



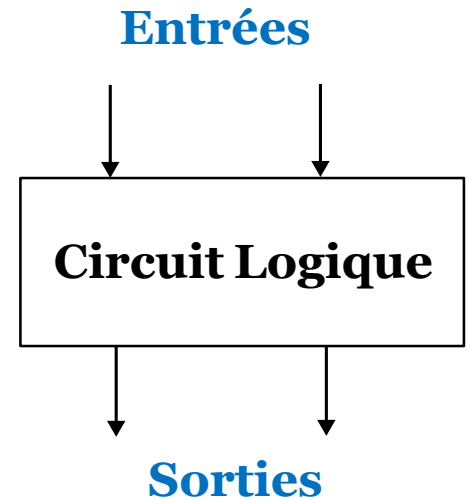
**Ordinateur  
(desktop)**

**Carte mère**



# Circuit logique

- Un circuit logique (ou digital) est un réseau de composants qui traitent des **données** à valeurs discrètes et produit des résultats, également à valeurs discrètes.
- Il peut être présenté comme une boîte noire avec les indications suivantes :
  - ses entrées ;
  - ses sorties ;
  - la spécification (non ambiguë de sa fonction).
- Les entrées et sorties ne peuvent prendre que « 0 » ou « 1 » comme valeur.
- Les circuits logiques sont de deux types : **combinatoires** et **séquentiels**.

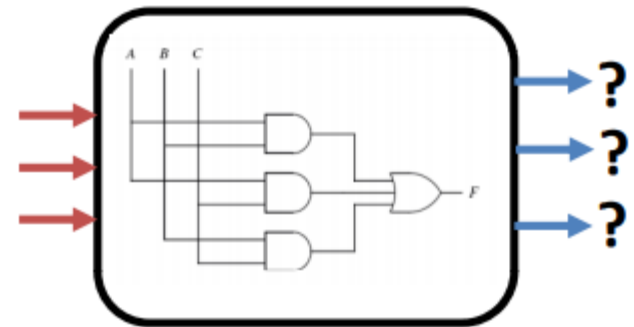


# Circuit logique

Les questions qui se posent au sujet des circuits logiques, qu'ils soient combinatoires ou séquentiels sont de deux types : Analyse ou Conception.

## ❖ Analyse

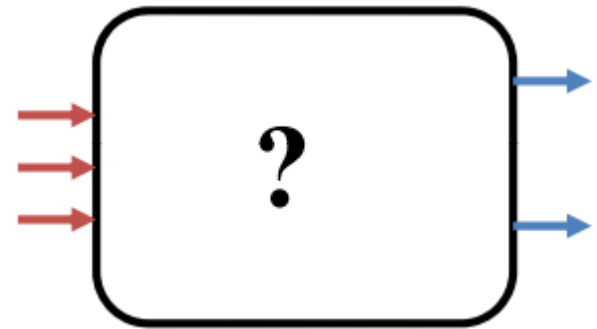
- Etant donné un circuit, trouver sa fonction.
- La fonction peut être exprimée par :
  - Une équation Booléenne.
  - Une table de vérité.



# Circuit logique

## ❖ Conception (ou synthèse)

- Partant d'une fonction recherchée, déterminer le circuit.
- La fonction peut être exprimée par :
  - Une équation Booléenne.
  - Une table de vérité.
  - Un schéma de réalisation-simulation.



# Circuits Combinatoires

- 1. LES ADDITIONNEURS**
- 2. LES DÉCODEURS**
- 3. LES MULTIPLEXEURS**
- 4. LES DÉMULTIPLEXEURS**

# Circuits combinatoires

- Un circuit combinatoire est un circuit logique où **les sorties ne sont fonctions que des entrées.**
- Les **sorties** à l'instant  $t$  ne dépendent que des **entrées** à l'instant  $t$ . Autrement dit, ce sont des circuits **sans état**, ils n'ont **aucune idée de leur passé**, ils sont dits sans mémoire.
- Ils sont souvent *asynchrones* :
  - dès que leur entrées changent, leur sorties changent en conséquence (après un certain délai) ;
  - ils n'attendent pas un quelconque signal d'autorisation.
- Les circuits les plus simples sont appelés **portes logiques.**

# Circuits combinatoires

---

## **Un circuit combinatoire :**

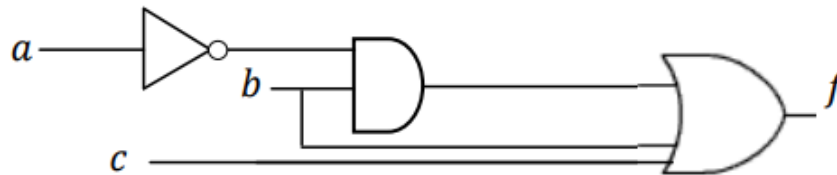
- est défini par un ensemble de portes reliées les unes aux autres ;
- les sorties des portes sont reliées aux entrées d'autres portes (définissant une orientation des connexions) ;
- en suivant l'orientation des connections, il est impossible qu'en partant de la sortie d'une porte, on revienne à l'une des ses entrées (graphe acyclique).

# Exemple de synthèse d'un circuit combinatoire

- L'objectif est de générer le logigramme du circuit à partir de la fonction logique correspondante :

$$f(a, b, c) = \bar{a} . b + b + c$$

- Représentation par la porte logique **NON**, la porte logique **ET** et la porte logique **OU** :

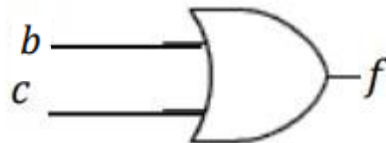


# Exemple de synthèse d'un circuit combinatoire

- On peut aussi réduire ou simplifier l'équation précédente en utilisant les propriétés de l'algèbre de Bool, sous la forme :

$$f(a, b, c) = \bar{a} \cdot b + b + c = b(\bar{a} + 1) + c = b + c$$

- Le logigramme correspondant est le suivant :



# Exemple de synthèse d'un circuit combinatoire

---

- Cette solution est la plus économique en termes de portes logiques utilisées.
- Donc, avant d'effectuer la synthèse d'un circuit combinatoire, on doit d'abord poser les questions suivantes :
  - Quelles portes logiques utilisées pour la représentation schématique ?
  - Est-ce que la fonction est simplifiée ?

# Additionneur

---

- L'addition de deux nombres binaires s'accomplit en additionnant les bits les moins forts, en allant vers ceux des rangs les plus forts.
- Deux exemples d'additionneur :
  - Demi-additionneur ;
  - Additionneur complet.

# Demi-additionneur

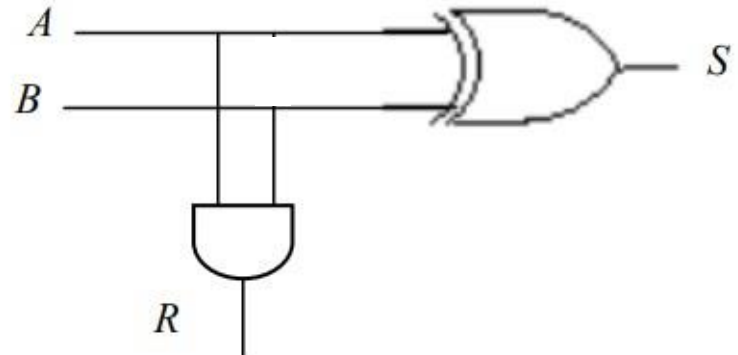
- Un demi-additionneur est un circuit qui prend en entrée deux bits, et qui produit la somme (addition) de ces deux nombres **S** et la retenue éventuelle **R**.
- C'est un additionneur sans tenir compte de la retenue précédente.

## Table de vérité

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

$$S = A \oplus B$$
$$R = A \cdot B$$

## Logigramme



# Additionneur complet

- C'est un additionneur qui prend en compte la retenue précédente.

## Table de verité

A	B	Re	S	Rs
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

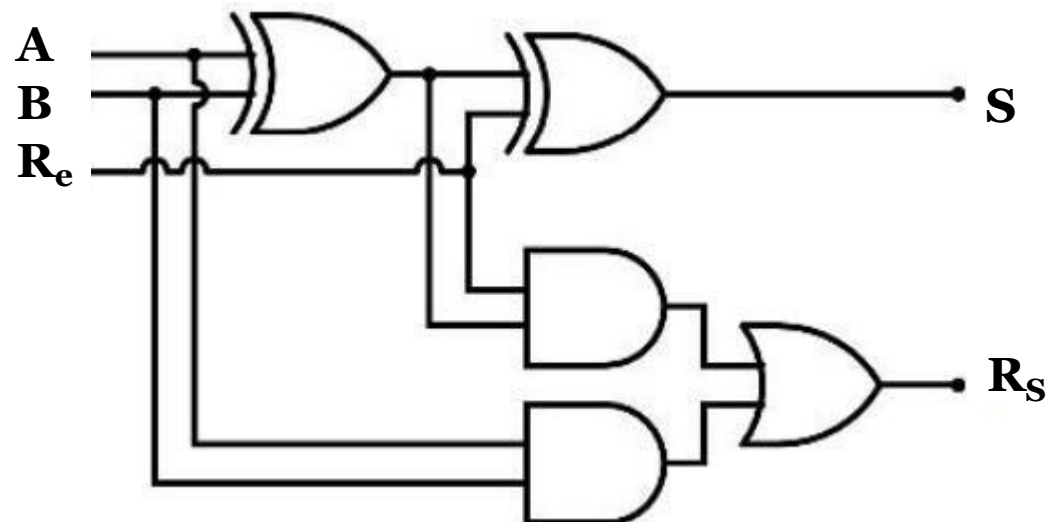
Re: Retenue en Entrée

Rs: Retenue en Sortie

$$S = A \oplus B \oplus Re$$

$$R_s = (A \cdot B) + Re \cdot (A \oplus B)$$

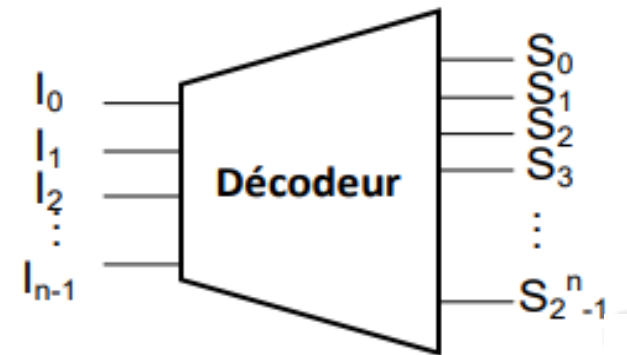
## Logigramme



# Décodeur

- Le décodeur est un circuit à  **$n$  entrées** et  **$2^n$  sorties** (au plus) dont une seule est active à la fois.
- C'est un circuit qui permet de sélectionner **une seule sortie** à partir de son adresse binaire (la combinaison spécifique des bits des entrées).
- **Application d'un décodeur** : la sélection de bloc mémoire. En effet, lorsque l'ordinateur fait référence à une case mémoire, il la désigne par son adresse binaire.

Celle-ci doit être décodée pour qu'un et un seul bloc mémoire soit sélectionné.



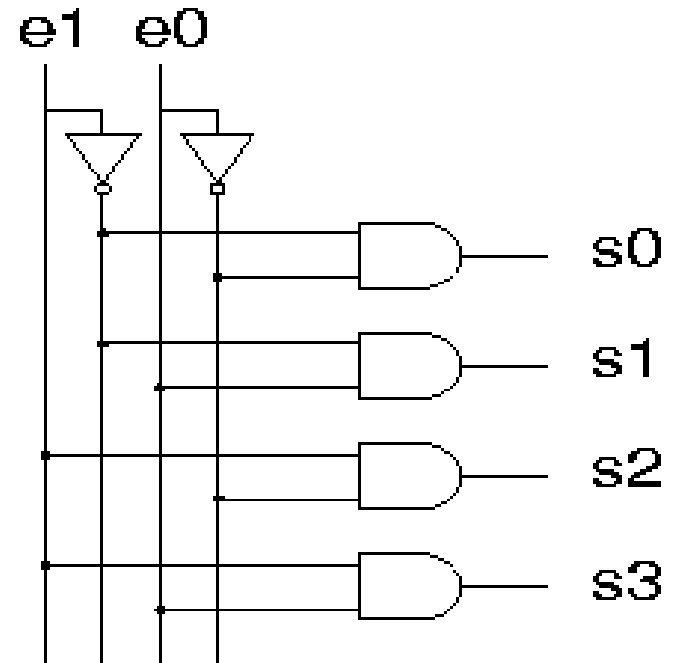
# Décodeur 2 vers 4

- Deux entrées **e1** et **e0**
- Quatre sorties **s0**, **s1**, **s2** et **s3**

**Table de vérité**

<b>e0</b>	<b>e1</b>	<b>s0</b>	<b>s1</b>	<b>s2</b>	<b>s3</b>
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

**Logigramme**



# Multiplexeur (MUX)

- Le multiplexeur est un circuit combinatoire qui comporte  **$2^n$  entrées** et une **seule sortie**.
- Il permet de choisir **une entrée** parmi les  $2^n$  et de la recopier sur la sortie.
- Pour la sélection de l'entrée, il y a  **$n$  entrées** de sélection qui codent le numéro en binaire de l'entrée voulue.
- **L'intérêt :** pouvoir connecter **plusieurs entrées** à un même circuit et **sélectionner l'entrée** voulue simplement en indiquant son adresse sur les lignes de commande. On peut ainsi relier plusieurs composants entre eux en minimisant les fils de connexion.

# Multiplexeur 4 vers 1

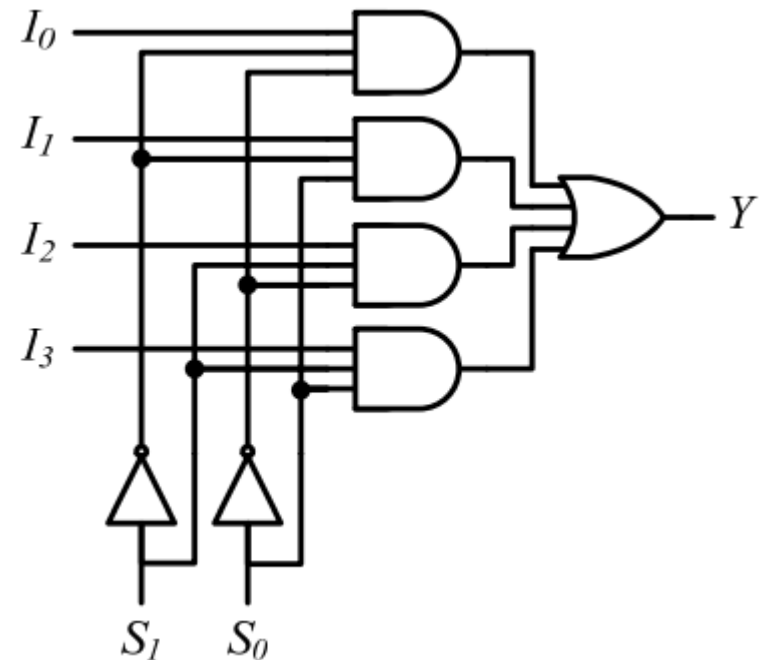
- 4 entrées  $I_0, I_1, I_2, I_3$
- 2 entrées de sélection  $S_1$  et  $S_0$
- 1 sortie  $Y$

## Table de vérité

$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

$$Y = \overline{S_1}.\overline{S_0}.I_0 + \overline{S_1}.S_0.I_1 + S_1.\overline{S_0}.I_2 + S_1.S_0.I_3$$

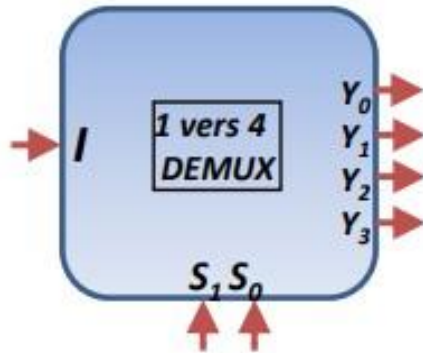
## Logigramme



# Démultiplexeur (Demux)

- Un démultiplexeur est un aiguilleur à **une entrée de donnée**.
- Il contient :
  - Une entrée de donnée ;
  - $n$  entrées d'adresse (bits de sélection) ;
  - $2^n$  sorties.
- La valeur de l'entrée se retrouve sur la sortie dont le numéro est codé par l'adresse.
- Dans cette fonction, le circuit joue le rôle inverse du multiplexeur.

# Demux 1 vers 4



**Table de vérité**

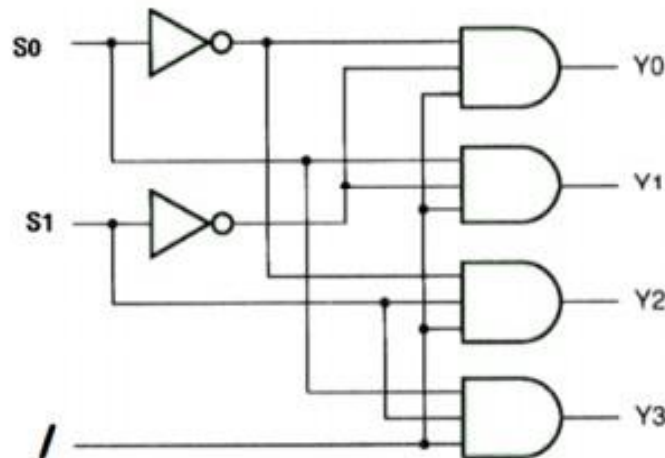
S1	S0	Y3	Y2	Y1	Y0
0	0	0	0	0	<i>I</i>
0	1	0	0	<i>I</i>	0
1	0	0	<i>I</i>	0	0
1	1	<i>I</i>	0	0	0

$$Y_0 = \overline{S_1} \cdot \overline{S_0} \cdot I$$

$$Y_1 = \overline{S_1} \cdot S_0 \cdot I$$

$$Y_2 = S_1 \cdot \overline{S_0} \cdot I$$

$$Y_3 = S_1 \cdot S_0 \cdot I$$



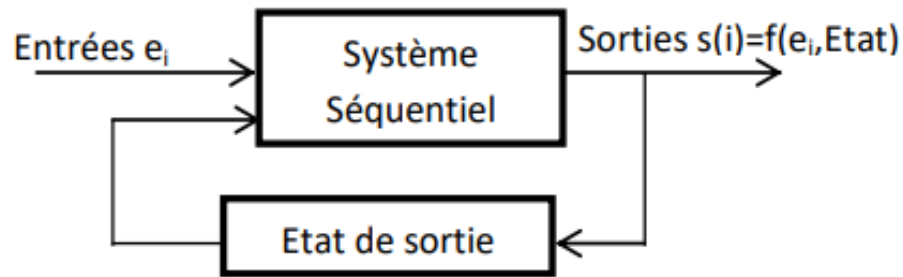
**Logigramme**

# Circuits séquentiels

1. **BASCULES RS**
2. **BASCULE JK**
3. **BASCULE D**

# Circuits séquentiels

- Ce sont ceux où **les sorties** à l'instant  $t$  dépendent non seulement des entrées à l'instant  $t$ , mais aussi des entrées précédentes ou de leur **état interne**.
- On peut représenter un système séquentiel par le schéma suivant :



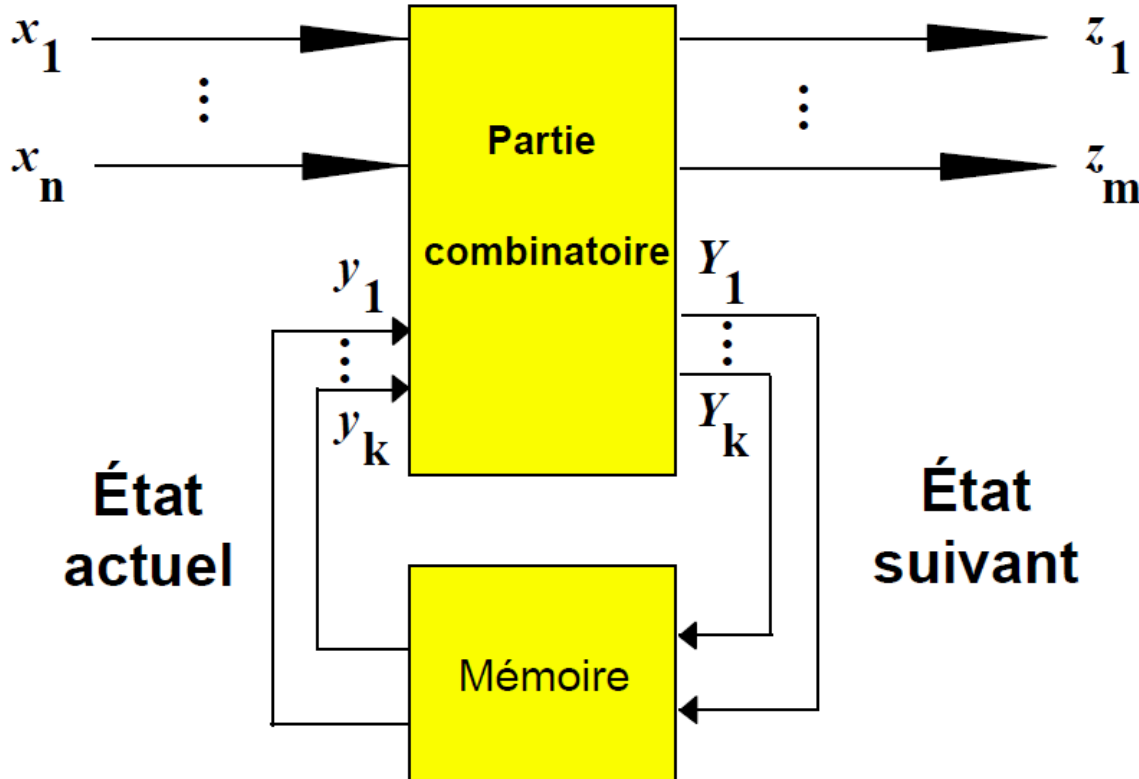
- Ils sont souvent **synchrones** : leurs sorties ne changent que sur **autorisation d'un signal** particulier qui marque les instants significatifs dans le temps.

# Circuits séquentiels

- Les circuits séquentiels sont construits au moyen des **circuits combinatoires** et de cellules élémentaires de mémorisation (**bascules**).
- Une bascule est un circuit pouvant prendre deux états logiques "0" et "1", qui sert à mémoriser un bit.
- **L'état de la bascule** peut être modifiée en agissant sur une ou plusieurs entrées. Le nouveau état de la bascule dépend de l'état précédent.
- La bascule peut conserver son état pendant une durée. Elle peut donc être utilisée comme mémoire.

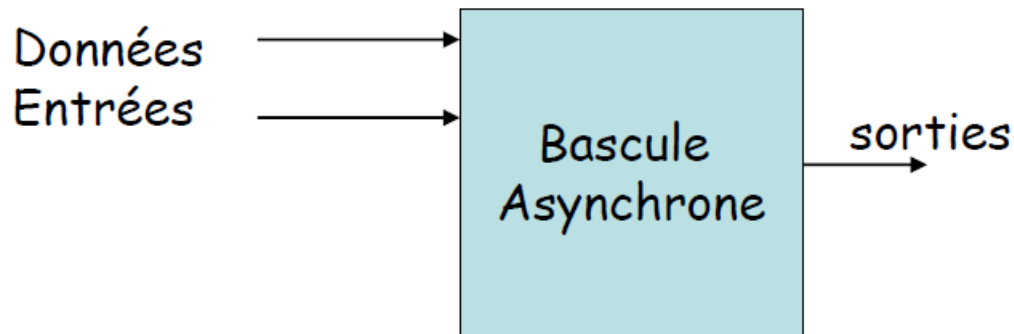
# Circuits séquentiels

**$n$  Variables d'entrée**



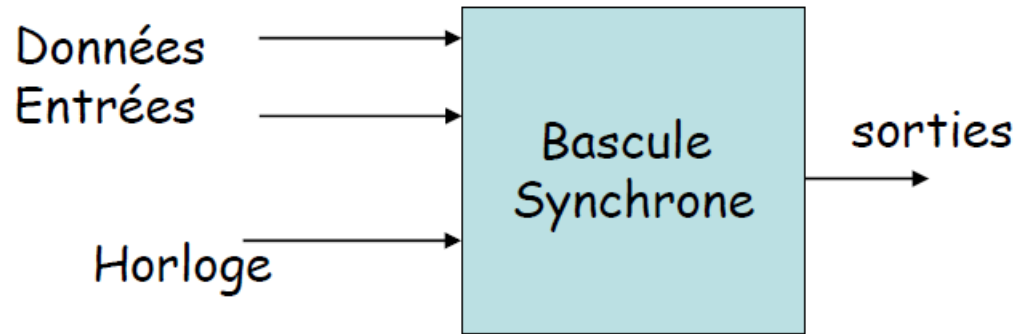
# Circuits séquentiels

- Il existe deux catégories de bascules :
  - Asynchrones
  - Synchrones
- Les **bascules asynchrones** sont des bascules dont la sortie ou l'état de mémorisation dépend à tout instant de l'état simultané des entrées.



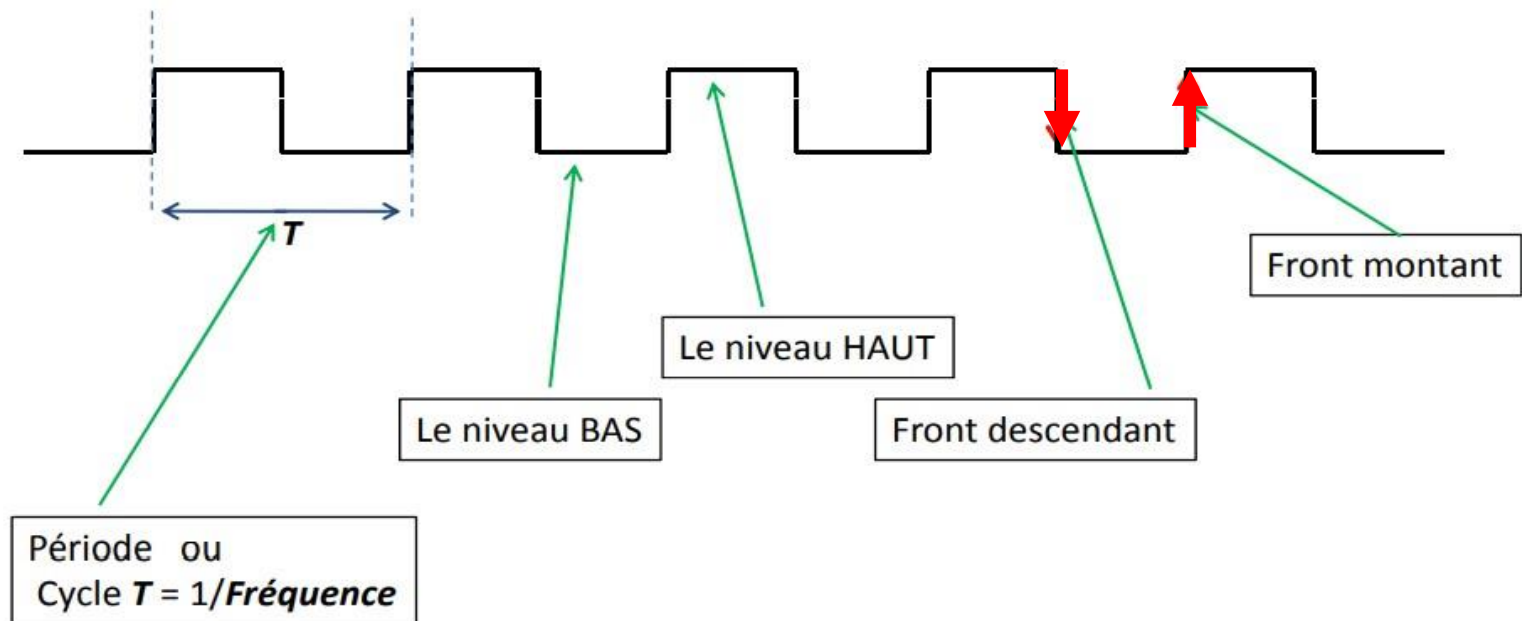
# Circuits séquentiels

- Les **bascules synchrones** possèdent une entrée d'horloge de synchronisation.



# Horloge (Clock)

- L'Horloge (**H** ou **Clk**) est un signal électrique périodique. Il passe alternativement et de manière périodique d'un niveau haut (1) à un niveau bas (0).



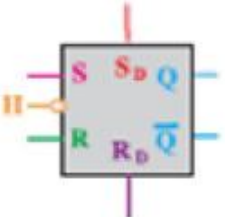
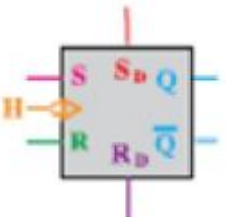


# Horloge (Clock)

- Fréquence = nombre de changement par seconde en hertz (Hz)

**Fréquence = 1/période ou cycle T**

Une horloge de 1 Hz a une période de 1 seconde  
 1 megaHz a une période de 1 microseconde  
 1 gigaHz a une période de 1 nanoseconde

Bascules à commande par niveau d'horloge		Bascules à commande par front d'horloge	
	Bascule synchrone RSH à niveau haut		Bascule synchrone RSH à front montant
	Bascule synchrone RSH à niveau bas et à commande asynchrone		Bascule synchrone RSH à front descendant et à commande asynchrone

# Bascules

---

- Il existe plusieurs types de bascules :
  - La bascule RS (asynchrone et synchrone)
  - La bascule JK
  - La bascule D

# Bascule RS asynchrone

- $Q_t$  est l'état de la sortie logique  $Q$  à l'instant  $t$ . Par convention, l'instant  $t$  est **l'instant actuel**.
- $Q_{t-1}$  était donc l'état de la sortie logique  $Q$  à l'instant  $t-1$ , c'est-à-dire à l'instant précédant l'évènement faisant évoluer la bascule.
- $Q_{t+1}$  sera donc l'état de la sortie logique  $Q$  à l'instant  $t+1$ , c'est-à-dire à l'instant suivant l'évènement et faisant donc évoluer la bascule.
- Les entrées  $R$  et  $S$  sont à la fois des entrées du type « état » et du type « temps ».

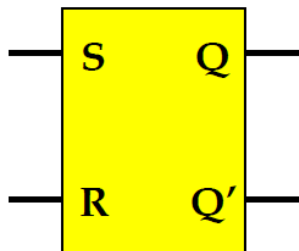
# Bascule RS asynchrone (NOR)

- Est constituée de deux entrées **S** et **R** et de deux sorties **Q** et  $\bar{Q}$  (ou **Q'**).
  - **S mise à 1** (“Set”) implique que **Q = 1** (Q est forcé à 1 par S) et  $\bar{Q} = 0$ .
  - **R mise à 0** (“Reset”) implique que **Q = 0** (Q est forcé à 0 par R) et  $\bar{Q} = 1$ .
  - Lorsque R et S sont à 0, **Q** et  $\bar{Q}$  ne changent pas d'état.
  - Lorsque R et S sont à 1, **Q** et  $\bar{Q}$  sont dans un état non logique.

# Bascule RS asynchrone (NOR)

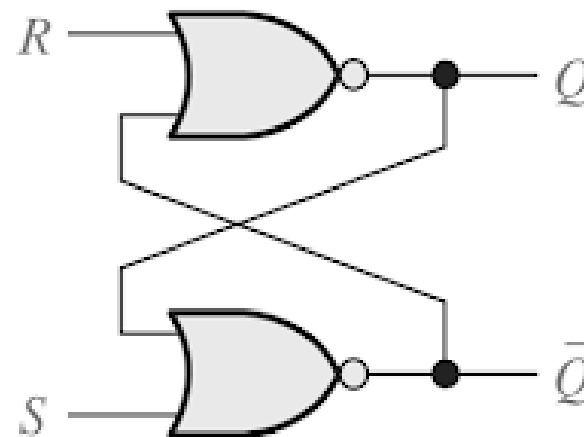
Table de vérité

Entrées		Sorties	
R	S	$Q_{t+1}$	$\overline{Q_{t+1}}$
0	0	$Q_t$ (état mémoire)	$\overline{Q_t}$ (état mémoire)
0	1	1 (Set)	0
1	0	0 (Reset)	1
1	1	X (interdit)	X (interdit)



Bascule RS d'entrée active en niveau HAUT.

Logigramme (portes NOR)



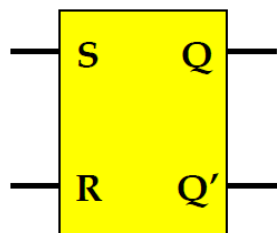
# Bascule RS asynchrone (NAND)

- Est constituée par deux entrées  $\bar{S}$  et  $\bar{R}$  et de deux sorties  $Q$  et  $\bar{Q}$  (ou  $Q'$ ).
  - $\bar{S}$  mise à 1 (“Reset”) implique que  $Q = 0$  ( $Q$  est forcé à 1 par  $\bar{S}$ ) et  $\bar{Q} = 1$ .
  - $\bar{R}$  mise à 0 (“Set”) implique que  $Q = 1$  ( $Q$  est forcé à 0 par  $\bar{R}$ ) et  $\bar{Q} = 0$ .
  - Lorsque  $\bar{S}$  et  $\bar{R}$  sont à 0,  $Q$  et  $\bar{Q}$  sont dans un état non logique.
  - Lorsque  $\bar{S}$  et  $\bar{R}$  sont à 1,  $Q$  et  $\bar{Q}$  ne changent pas d'état.

# Bascule RS asynchrone (NAND)

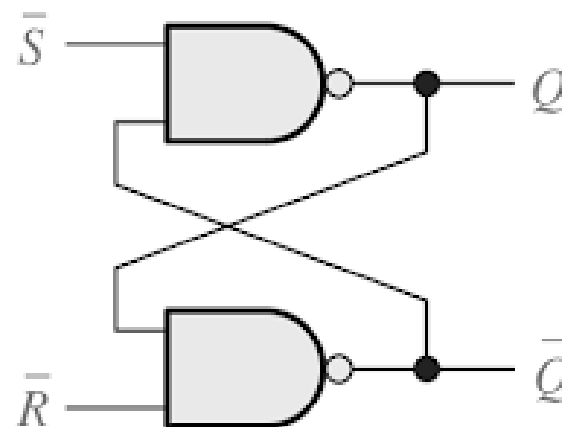
Table de vérité

Entrées		Sorties	
$\bar{R}$	$\bar{S}$	$Q_{t+1}$	$\overline{Q_{t+1}}$
0	0	X (interdit)	X (interdit)
0	1	0 (Reset)	1
1	0	1 (Set)	0
1	1	$Q_t$ (état mémoire)	$\bar{Q}_t$ (état mémoire)



Bascule  $\bar{R} \bar{S}$  d'entrée active en niveau BAS.

Logigramme (portes NAND)



# Bascule RS synchrone

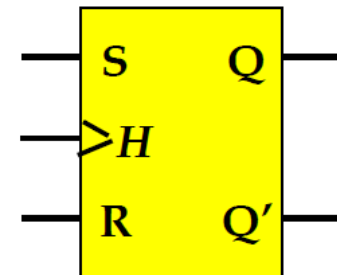
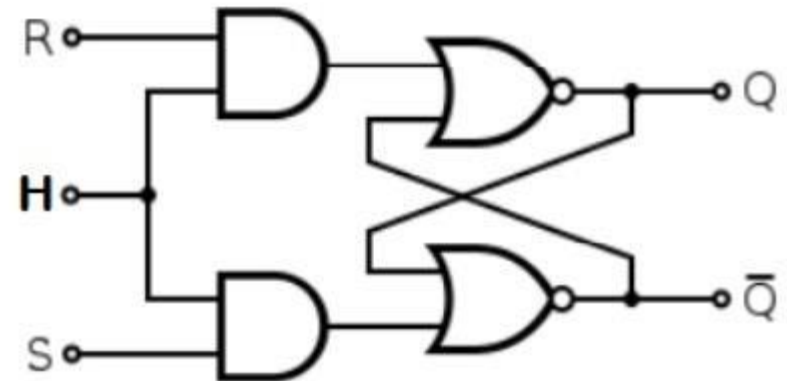
- Est une bascule RS dont la prise en compte de l'état des entrées est synchronisée par une **impulsion d'horloge**.
- Si  $H = 0$ , il y a la mémorisation de l'état précédent (mémoire figée).
- Si  $H = 1$ , la bascule RS répond normalement aux commandes appliquées à ses entrées (mémoire classique).

# Bascule RS synchrone

Table de vérité

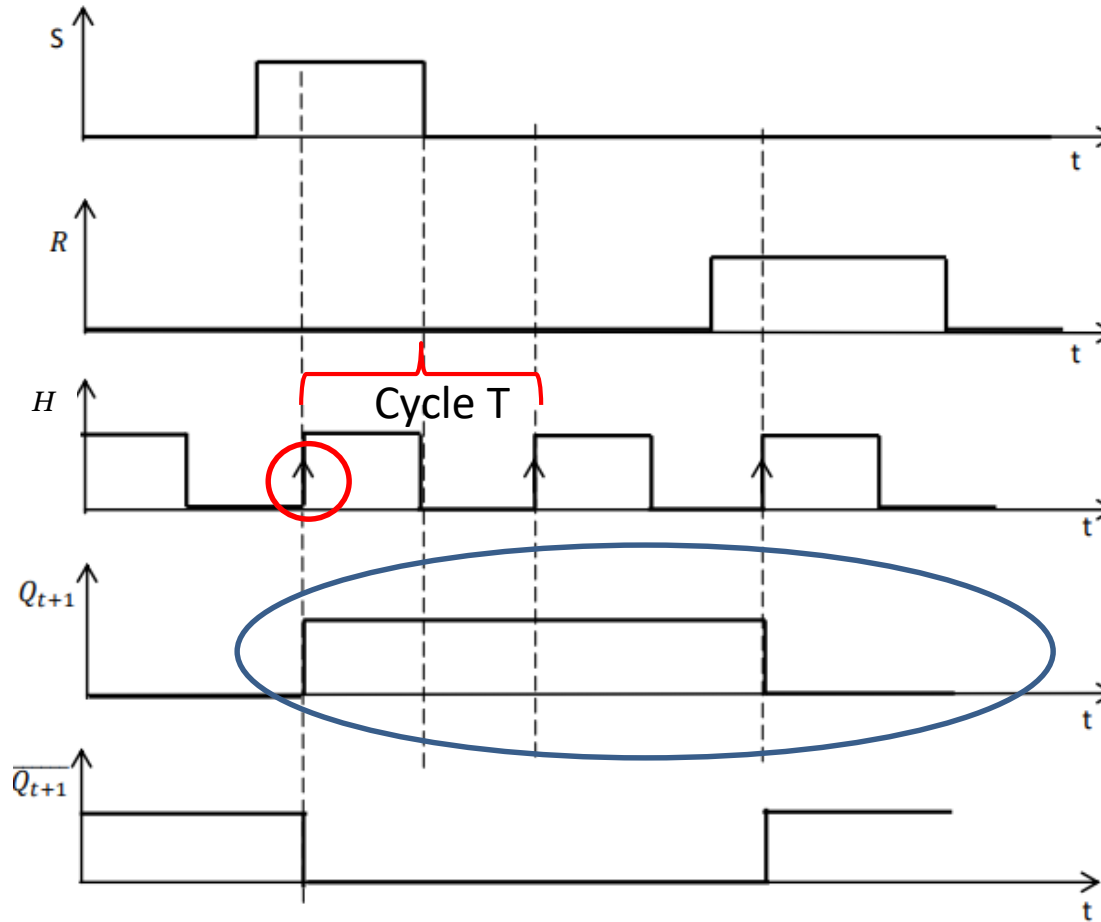
Entrées			Sorties	
H	R	S	$Q_{t+1}$	$\overline{Q}_{t+1}$
0	0	0	$Q_t$ (état mémoire)	$\overline{Q}_t$ (état mémoire)
1	0	0	$Q_t$ (état mémoire)	$\overline{Q}_t$ (état mémoire)
1	0	1	1 (Set)	0
1	1	0	0 (Reset)	1
1	1	1	X (interdit)	X (interdit)

Logigramme



# Bascule RS synchrone

## Chronogramme correspondant (front montant)



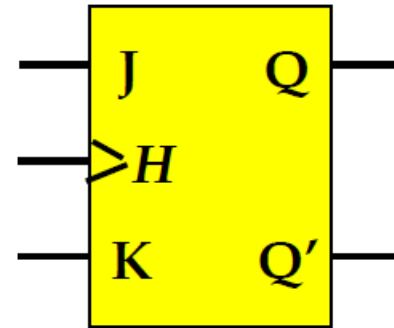
# Bascule JK

- La bascule JK est une bascule RS, mais dans le cas où la valeur de deux entrées est = 1, la sortie est portée à la négation de l'état précédente.
  - Telle que  $S = J$  et  $R = K$
- La bascule JK est la plus évoluée et son rôle est essentiel au comptage. Elle permet d'effectuer du comptage et de prépositionner, par ses entrées J et K, le départ du comptage et son arrêt.
  - Quand  $J = K = 1$ , la sortie bascule
  - J = Jump (mise à 1)
  - K = Kill (mise à 0)

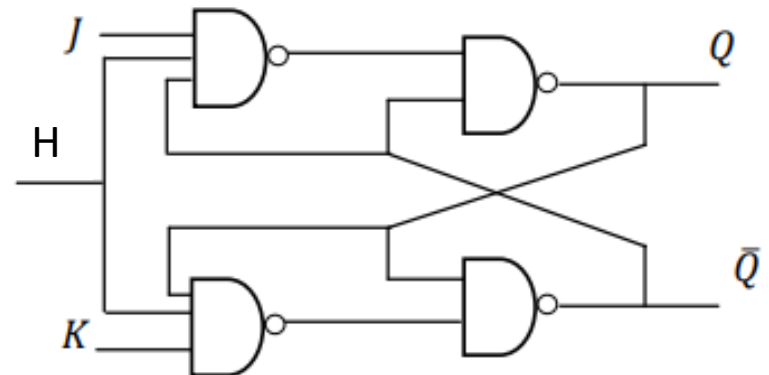
# Bascule JK

**Table de vérité**

Entrées			Sorties	
H	J	K	$Q_{t+1}$	$\overline{Q_{t+1}}$
0	x	x	$Q_t$ (état mémoire)	$\overline{Q_t}$ (état mémoire)
1	0	0	$Q_t$ (état mémoire)	$\overline{Q_t}$ (état mémoire)
1	0	1	0 Kill (mise à 0)	1
1	1	0	1 Jump (mise à 1)	0
1	1	1	$\overline{Q_t}$ (état mémoire)	$Q_t$ (état mémoire)

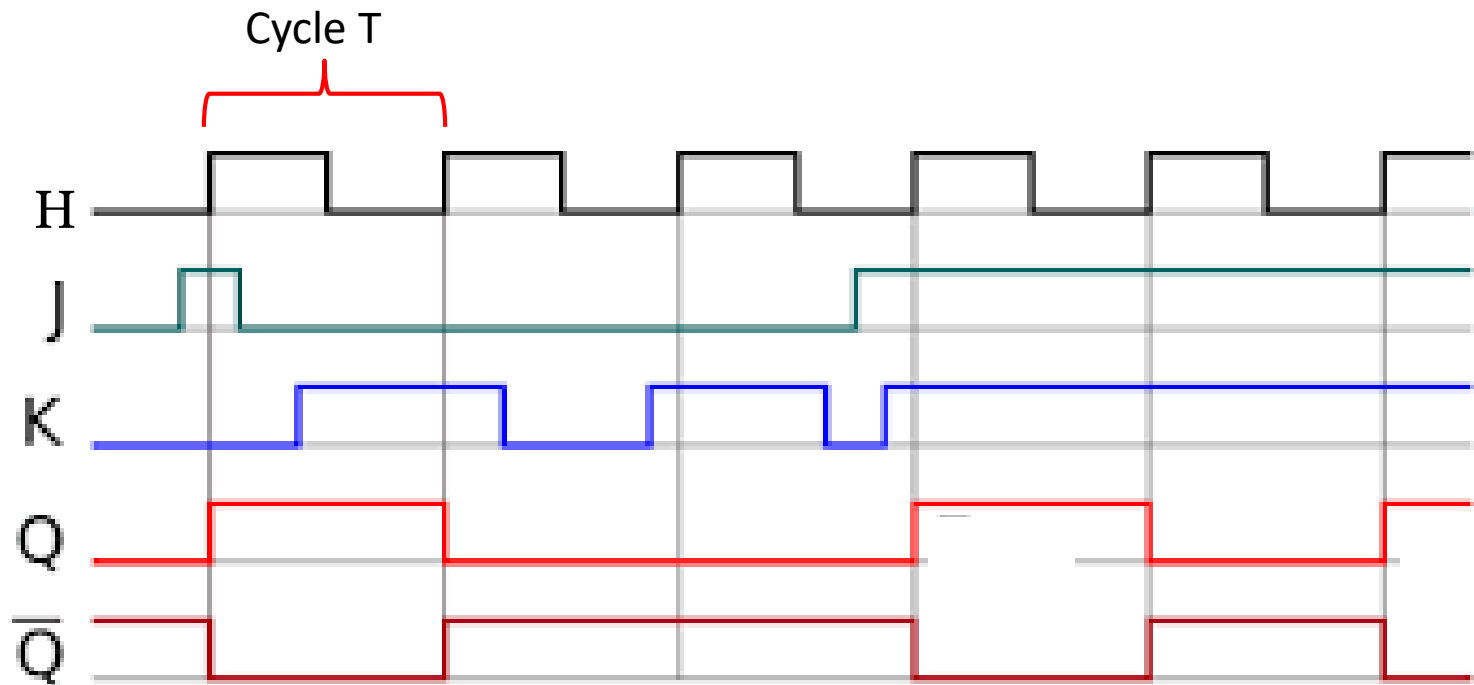


**Logigramme**



# Bascule JK

## Chronogramme correspondant



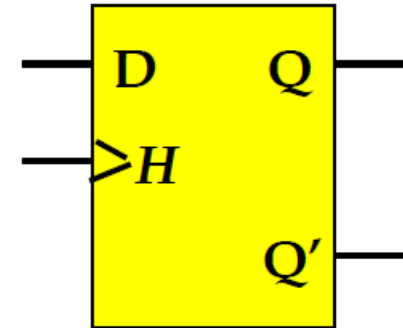
# Bascule D

- La bascule D (*Data*) est une bascule comportant uniquement une entrée de données : **D**. La valeur de l'entrée est recopiée sur la sortie à chaque **front d'horloge**.
- La bascule D est utilisée pour éliminer l'état indéterminé (**interdit**). Elle résume les deux entrées R et S par une seule entrée D telle que :  $S=D$  et  $R= D$ .

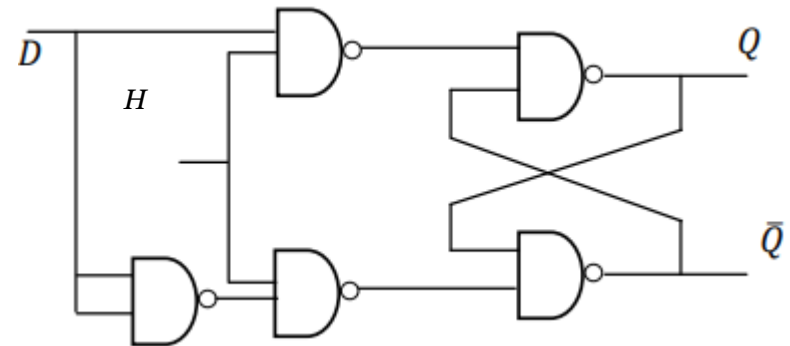
# Bascule D

## Table de vérité

Entrées		Sorties	
H	D	$Q_{t+1}$	$\overline{Q_{t+1}}$
0	0	$Q_t$ (état mémoire)	$\overline{Q_t}$ (état mémoire)
0	1	$Q_t$ (état mémoire)	$\overline{Q_t}$ (état mémoire)
1	0	0	1
1	1	1	0



## Logigramme



# Bascule D

## Chronogramme correspondant (niveau Haut)

