جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

Architecture des ordinateurs

Chapitre 2:

Circuits Combinatoires et séquentiels

Pr. S. Amri

GI_ILSI / 3ème Année

AU : 2024/2025

Chapitre 2 : Logique combinatoire et séquentielle

L'analyse des circuits logiques nécessite l'utilisation d'une algèbre spécifique, dite « booléenne », fruit des travaux du mathématicien anglais George Boole du 19ème siècle. Ces travaux ont défini un ensemble d'opérateurs de base, ainsi que leurs propriétés, qui constituent une algèbre permettant de concevoir tout type de circuit. La construction de fonctions logiques répondant à des contraintes précises et leur représentation sous forme de circuits électroniques est réalisée par des opérateurs élémentaires. [1]

Alors qu'en algèbre classique, les variables et les fonctions peuvent prendre n'importe quelle valeur, ici elles sont limitées aux valeurs 0 et 1. Une fonction de n variables booléennes sera définie à partir de $\{0.\ 1\}^n$ à $\{0,\ 1\}$. Il est également complètement défini par les valeurs qu'il suppose dans les 2^n combinaisons possibles de ses variables d'entrée, comme chaque valeur de fonction ne peut être que 0 ou 1, il existe 2^{2^n} fonctions différentes de n variables. On peut donc décrire une fonction donnée (avec n variables) en expliquant ses 2^n valeurs, par exemple, à partir de sa table de vérité. Il s'agit d'un tableau de 2^n lignes, qui répertorie pour chaque combinaison possible de variables d'entrée (une ligne) la valeur assumée par la fonction.

1. *Circuits logiques*

Un circuit logique est un Circuit dans lequel seules 2 valeurs logiques sont possibles : 0 ou 1. Il est réalisé par un circuit électrique (transistors) :

- Une faible tension représente la valeur binaire 0.
- Une tension élevée représente 1.

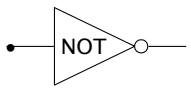
Les composants de base d'un circuit logique sont : les **portes logiques**.

► Porte logique :

- Permet de combiner les signaux binaires.
- Reçoit en entrée une ou plusieurs valeurs binaires (souvent 2).
- Renvoie une unique valeur binaire en sortie.

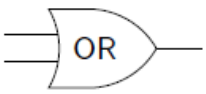
a. Fonctions logiques élémentaires

i. Fonction NON (NOT)



- Si la valeur d'entré est 1, alors la sortie vaut 0.
- Si la valeur d'entrée est 0, alors la sortie vaut 1.

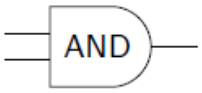
ii. Fonction OU (OR)



a	b	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

$$S = f(a,b) = a + b$$

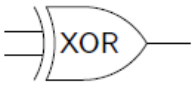
iii. Fonction ET (AND)



a	b	S
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

$$S = f(a,b) = a \times b = ab$$

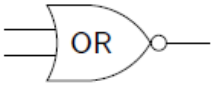
iv. Fonction OU-exclusif (XOR)



a	b	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

$$\begin{aligned} S = f(a,b) &= a \oplus b \\ &= (a+b)\overline{(ab)} \\ &= (a+b)(\bar{a}+\bar{b}) \\ &= a\bar{a} + a\bar{b} + b\bar{a} + b\bar{b} \\ &= a\bar{b} + b\bar{a} \\ &= \overline{\overline{a\bar{b}}} + \overline{\overline{b\bar{a}}} \\ &= \overline{\overline{a\bar{b}}\;\overline{b\bar{a}}} \end{aligned}$$

v. Fonction NON-OU (NOR)



a	b	S
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

$$S = f(a,b) = \overline{a+b}$$

vi. Fonction NON-ET (NAND)

a	b	S
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

$$S = f(a,b) = \overline{a.b} = \overline{ab}$$

► Exemple des fonctions booléennes de 2 variables :

$f(a,b)$	00	01	10	11
0	0	0	0	0
ab	0	0	0	1
$a\bar{b}$	0	0	1	0
a	0	0	1	1
$\bar{a}b$	0	1	0	0
b	0	1	0	1
$a \oplus b$	0	1	1	0
$a+b$	0	1	1	1

$f(a,b)$	00	01	10	11
$\overline{a+b}$	1	0	0	0
$\overline{a \oplus b}$	1	0	0	1
$\bar{b}$	1	0	1	0
$a+\bar{b}$	1	0	1	1
$\bar{a}$	1	1	0	0
$\bar{a}+b$	1	1	0	1
$\overline{ab}$	1	1	1	0
1	1	1	1	1

b. *Règles de calcul*

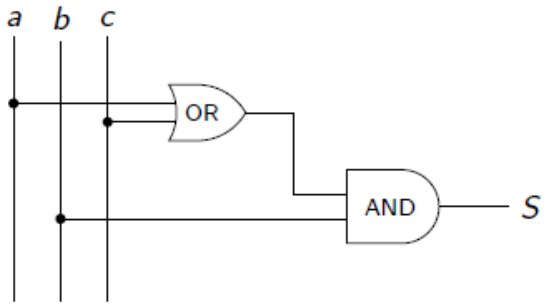
Commutativité	Associativité	Distributivité
$a+b=b+a$ $ab=ba$ $a \oplus b=b \oplus a$	$a+(b+c)=(a+b)+c=a+b+c$ $a(bc)=(ab)c=abc$ $a \oplus (b \oplus c)=(a \oplus b) \oplus c=a \oplus b \oplus c$	$a+(bc)=(a+b)(a+c)$ $a(b+c)=(ab)+(ac)=ab+ac$ $a(b \oplus c)=(ab) \oplus (ac)=ab \oplus ac$
Elément neutre	Elément absorbant	Idempotence
$a+0=a$ $1.a=a$ $a \oplus 0=a$	$a+1=1$ $0.a=0$	$a+a=a$ $aa=a$
Complémentaire	Lois de Morgan	Divers
$a+\bar{a}=1$ $a\bar{a}=0$ $\overline{a\bar{a}}=\bar{a}$ $\overline{a+a}=\bar{a}$ $\bar{\bar{a}}=a$ $a \oplus \bar{a}=1$ $a \oplus 1=\bar{a}$	$\overline{ab}=\bar{a}+\bar{b}$ $\overline{a+b}=\bar{a}\bar{b}$	$a+ab=a(a+b)=a$ $a+(\bar{a}b)=a+b$ $a(\bar{a}+b)=ab$ $a \oplus a=0$ $a \oplus \bar{b}=\bar{a} \oplus b=\overline{a \oplus b}$ $\bar{a} \oplus \bar{b}=a \oplus b$ $a \oplus b=a\bar{b}+\bar{a}b$ $\overline{a \oplus b}=ab+\bar{a}\bar{b}$

Table 1 : Tableau récapitulatif des règles de calcul logique [1]

c. *Construction et optimisation*

i. Du circuit logique à la table de vérité :

Soit le schéma d'un circuit logique :



a	b	c	$a+c$	$S=b(a+c)$
0	0	0	0	**0**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	1	1	**1**
1	0	0	1	**0**
1	0	1	1	**0**
1	1	0	1	**1**
1	1	1	1	**1**

⇒ La table de vérité correspondante.

ii. De la table de vérité au circuit logique :

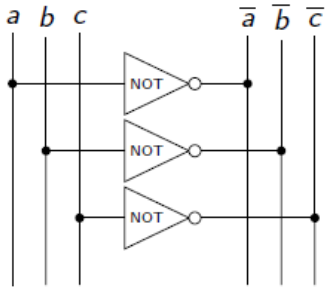
1) Ecrire l'équation de la fonction à partir de sa table de vérité

a	b	c	S
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

- Pour chaque cas où la sortie vaut 1 ajouter un « *minterme* » à la fonction.
- Un *minterme* est un AND de toutes les variables d'entrée de la fonction (éventuellement complémentées) :
  - Si la variable d'entrée vaut 1, elle est écrite directement dans le *minterme*.
  - Si la variable d'entrée vaut 0, elle est complémentée dans le *minterme*.

⇒ $S = f(a,b,c) = \bar{a}b\bar{c} + a\bar{b}c + abc$

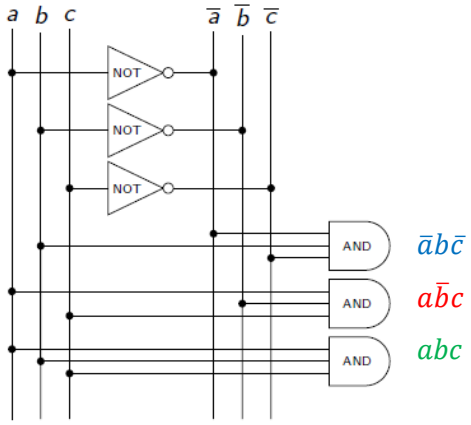
2) Réaliser la négation de toutes les variables d'entrée



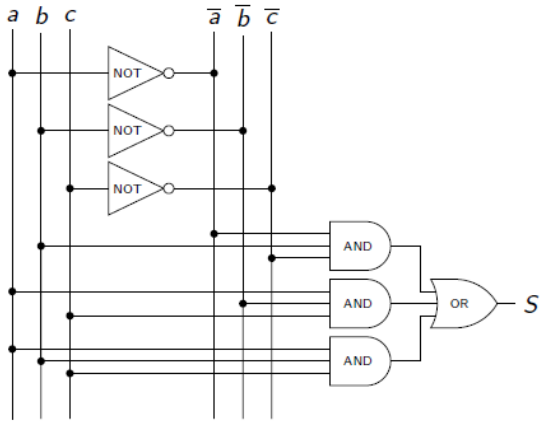
3) Construire une porte AND pour chacun des *mintermes*.

4) Etablir le câblage des portes OR avec les entrées appropriées.

$S = f(a,b,c) = \bar{a}b\bar{c} + a\bar{b}c + abc$ ⇒



5) Réunir l'ensemble des sorties des portes AND vers une porte OR, dont la sortie et le résultat de la fonction.



► Remarque : ce circuit n'est pas optimal pour la table de vérité initiale. Il faut **simplifier** la fonction pour minimiser le nombre portes logiques.

iii.Simplification

- Diminuer le nombre d'opérateurs.
- Diminuer le nombre de portes logiques (et donc le coût).

Il y a deux approches de simplification :

1) Méthode algébrique (algèbre de Boole).

► Exemple : la fonction majoritaire ⇒ donne 1 en sortie si la majorité des bits en entrée sont des 1 sinon 0.

$$\begin{aligned} f\ (a,b,c) &= \bar{a}bc + a\bar{b}c + abc + abc \\ &= (\bar{a}b + a\bar{b})c + ab(c + \bar{c}) \\ &= (a + b)(\bar{a} + \bar{b})c + ab \\ &= (ac + bc)\overline{ab} + ab \\ &= (ab + ac + bc)(\overline{ab} + ab) \\ &= ab + ac + bc \end{aligned}$$

a	*b*	*c*	*S*
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**

2) Méthode des tableaux de Karnaugh

- Permet de visualiser une fonction et d'en tirer naturellement une écriture simplifiée.
- Représentation de toutes les combinaisons d'états possibles pour un nombre de variables donné.
- Outil graphique qui permet de simplifier de manière méthodique des expressions booléennes.
- Exploite le codage de l'information et la notion d'adjacence.

✓ Principe :

- Mettre en évidence sur un graphique les *mintermes* ou *maxtermes* adjacents.
- Transformer les adjacences logiques en adjacences géométriques.

Table de vérité	vs.	**Tableau de Karnaugh**
1 ligne	⇒	1 case
n variables	⇒	2^n cases

✓ La méthode passe par trois phases :

1) Transcription de la fonction dans un tableau codé.
2) Recherche des adjacences pour simplification.
3) Mise en équations des groupements effectués.

► Exemple : fonction majoritaire

1) Ecrire la table de vérité sous la forme d'un code de Gray (ou binaire réfléchi) : les valeurs des entrées ne diffèrent que d'un seul bit entre chaque ligne.

a	*b*	*c*	*S*
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**

Table Initiale

⇒

a	*b*	*c*	*S*
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**

Code de Gray

2) Compacter la table : représenter la sortie en fonctions des entrées sous forme d'un tableau à 2 dimensions.

a	*b*	*c*	*S*
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**

⇒

a \ *bc*	**00**	**01**	**11**	**10**
0	0	0	1	0
1	0	1	1	1

3) Regrouper tous les bits 1 de telle sorte que :

- Les bits 1 d'un groupe sont adjacents ou des bords du tableau.
- La taille des groupes est une puissance de 2.
- Un groupe contient le plus de 1 possible.

a \ *bc*	**00**	**01**	**11**	**10**
0	0	0	1	0
1	0	1	1	1

4) En déduire la formule et le circuit :

- La formule est la somme (OR) des *mintermes*.
- Un *minterme* est le produit (AND) des variables du même groupe de bits :
  - Des variables qui valent toujours 1 dans ce groupe.
  - Des négations de celles qui valent toujours 0.
  - Les autres variables n'apparaissent pas dans le produit.

a \ bc	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$\Rightarrow \quad S = f(a,b,c) = bc + ac + ab$

2. *Circuits combinatoires*

a. Définitions

Les circuits combinatoires sont des circuits logiques combinés pour obtenir un nouveau circuit avec une fonction logique plus complexe. Ils sont nés du besoin de circuits logiques à plusieurs entrées et plusieurs sorties, et parfois des sorties qui dépendent d'entrées supplémentaires dites de sélection.

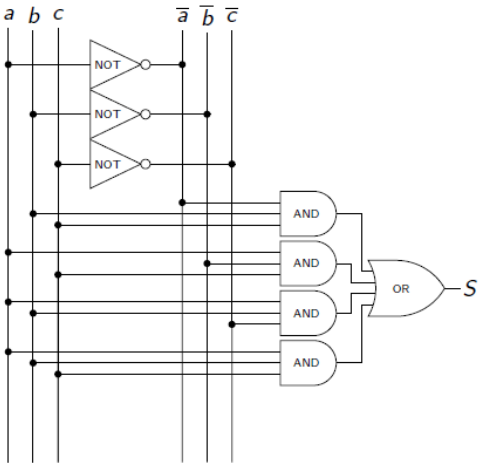
b. Caractéristiques

Un circuit combinatoire est caractérisé par :

- **Les entrées** :
  - Les données : ne sont pas des entrées de la table de vérité.
  - Les paramètres : bits de réglage.
  - Les variables d'entrée.
- **La sortie** : pas forcément unique !
  - Fonction logique : une seule valeur en sortie.
  - Circuit : plusieurs fonctions possibles pour obtenir le comportement voulu.
- **Le rôle** de différents éléments :
  - A quoi sert le circuit ?
  - Qu'obtient-on en sortie ?
  - Quel rôle jouent les entrées ?
- **La table de vérité** (une table par fonction).

► Exemple : la fonction majoritaire

a	b	c	S
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**



► Problème : sur un nombre pair d'entrées, une seule sortie ne suffit pas :
- Soit les 0 sont majoritaires (sortie 00)
- Soit les 1 sont majoritaires (sortie 01)
- Soit il n'y a pas de majoritaire (sortie 10)

► Solution : circuit combinatoire à 4 entrées et 2 sorties S_0 et S_1.

a	b	c	d	S_0	S_1
0	0	0	0	**0**	**0**
0	0	0	1	**0**	**0**
0	0	1	0	**0**	**0**
0	0	1	1	**1**	**0**
0	1	0	0	**0**	**0**
0	1	0	1	**1**	**0**
0	1	1	0	**1**	**0**
0	1	1	1	**0**	**1**
1	0	0	0	**0**	**0**
1	0	0	1	**1**	**0**
1	0	1	0	**1**	**0**
1	0	1	1	**0**	**1**
1	1	0	0	**1**	**0**
1	1	0	1	**0**	**1**
1	1	1	0	**0**	**1**
1	1	1	1	**0**	**1**

c. Circuits combinatoires de base

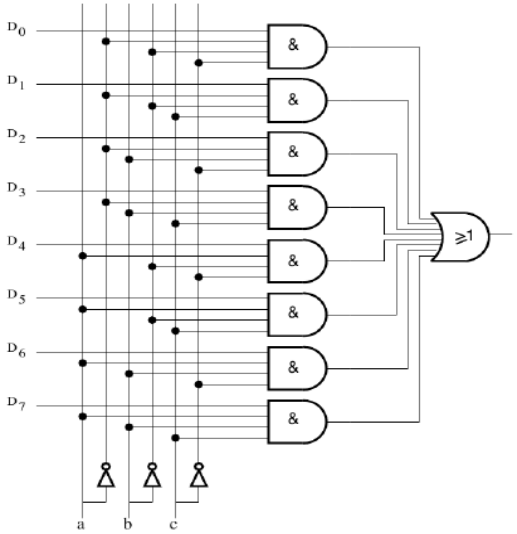
Les circuits combinatoires les plus communs :
- Le multiplexeur
- Le démultiplexeur
- Le décodeur
- Le comparateur
-

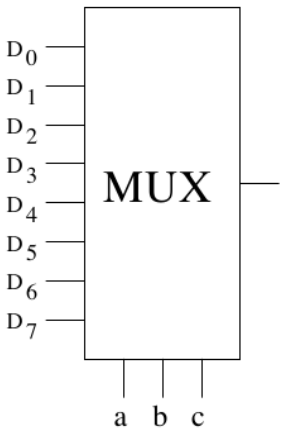
i. Le multiplexeur $2^n \times n$

- **Entrées** :
  - 2^n lignes d'entrée (données) : $D_0, \dots, D_{2^n-1}$
  - n lignes de sélection : $a, b, c, \dots$
- **Sortie** : une seule sortie S.
- **Rôle** : aiguiller la valeur de l'une des 2^n lignes d'entrée vers la sortie S.
  La ligne d'entrée choisie est désignée grâce aux bits de sélection.

a	b	c	S
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

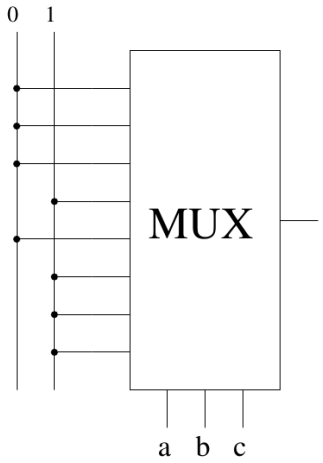
► Câblage du multiplexeur 8×3





► Exemple d'utilisation du multiplexeur : La fonction majoritaire avec un multiplexeur

a	b	c	S
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**

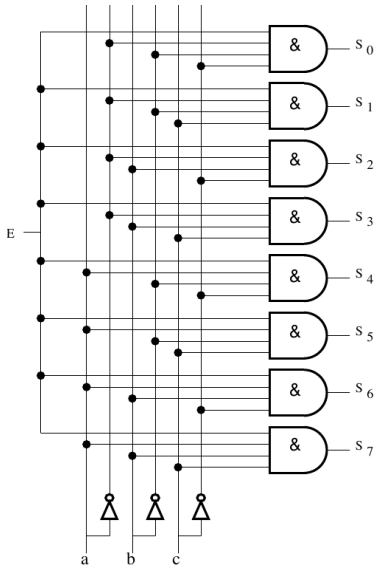


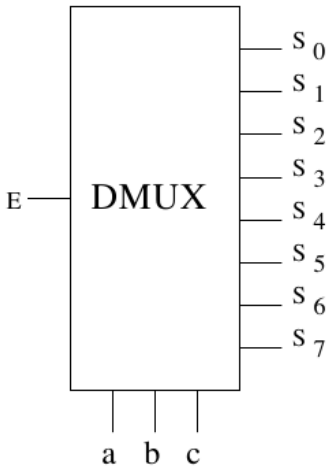
ii. Le démultiplexeur $2^n \times n$

- **Entrées** :
  - une lignes d'entrée (donnée) : E
  - n lignes de sélection : $a, b, c, \ldots$
- **Sortie** : 2^n lignes de sortie $S_0, \ldots, S_{2^n-1}$
- **Rôle** : aiguiller l'entrée E vers l'une des 2^n lignes de sortie.
  La ligne de sortie est désignée grâce aux bits de sélection.

a	b	c	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	E	0	0	0	0	0	0	0
0	0	1	0	E	0	0	0	0	0	0
0	1	0	0	0	E	0	0	0	0	0
0	1	1	0	0	0	E	0	0	0	0
1	0	0	0	0	0	0	E	0	0	0
1	0	1	0	0	0	0	0	E	0	0
1	1	0	0	0	0	0	0	0	E	0
1	1	1	0	0	0	0	0	0	0	E

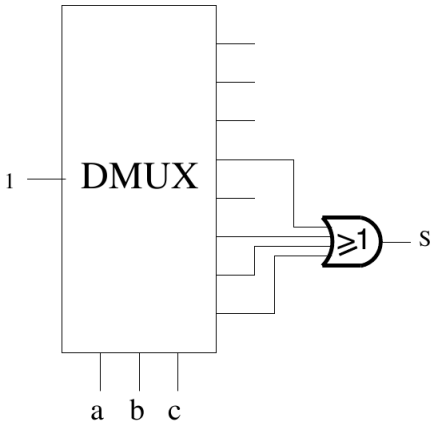
► Câblage du démultiplexeur 8×3





► Exemple d'utilisation du démultiplexeur : La fonction majoritaire avec un démultiplexeur

a	b	c	S
0	0	0	**0**
0	0	1	**0**
0	1	1	**1**
0	1	0	**0**
1	1	0	**1**
1	1	1	**1**
1	0	1	**1**
1	0	0	**0**

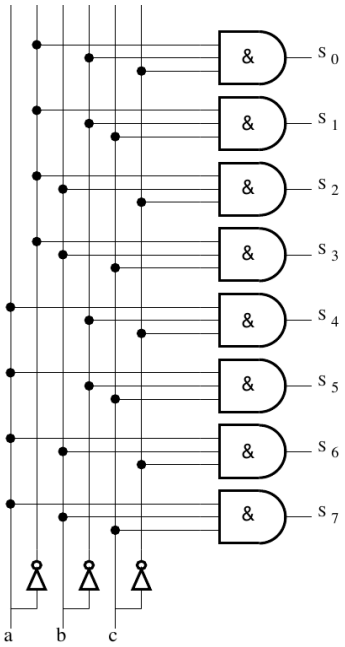


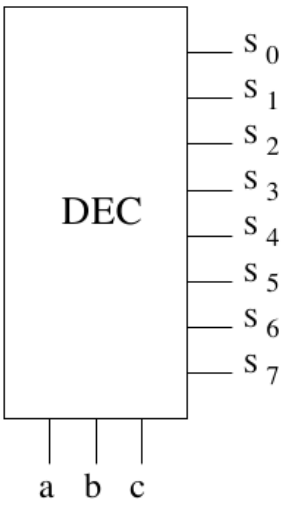
iii. Le décodeur $2^n \times n$

- **Entrées** : n lignes de sélection : $a, b, c, \ldots$
- **Sortie** : 2^n lignes de sortie $S_0, \ldots, S_{2^n-1}$
- **Rôle** : sélectionner (mettre à 1) l'une des 2^n lignes de sortie.
  La ligne de sortie est désignée grâce aux bits de sélection.

a	b	c	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

► Câblage du décodeur 8×3



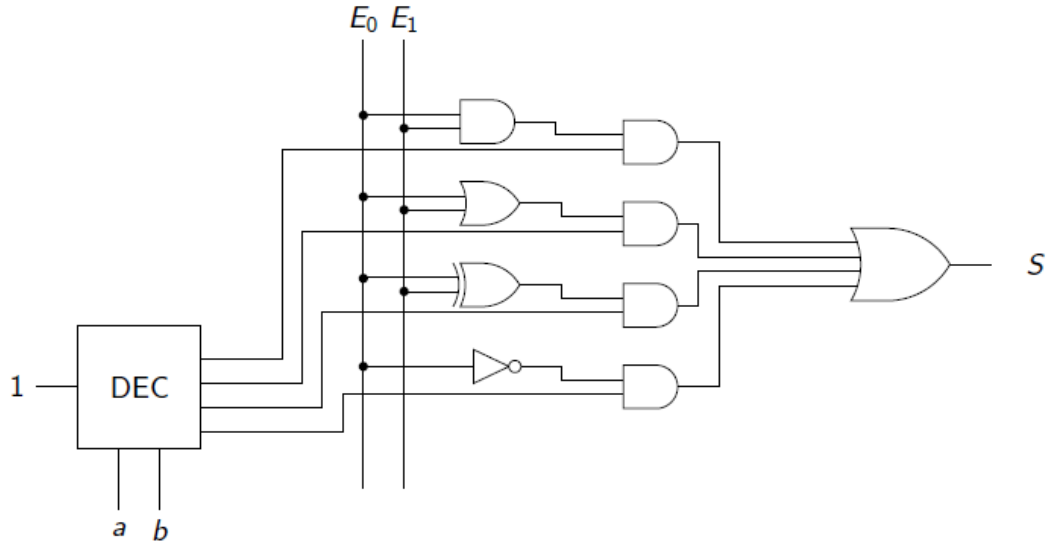


► Exemple d'utilisation d'un décodeur : activation de fonction

Ce circuit fait, au choix, l'une des 4 fonctions logiques (AND, OR, XOR, NOT) sur les données E_0 et E_1. Le choix de la fonction est déterminé par les valeurs de a et b selon la table de vérité suivante :

a	b	S
0	0	$E_0 \times E_1$
0	1	$E_0 + E_1$
1	0	$E_0 \oplus E_1$
1	1	$\overline{E_0}$

Réaliser le circuit logique correspondant en utilisant un décodeur



3. *Circuits arithmétiques*

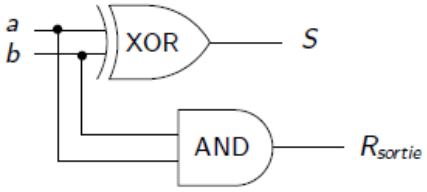
Les circuits arithmétiques sont des circuits combinatoires permettant d'effectuer des opérations arithmétiques sur les nombres en entrée. Les circuits arithmétiques de base sont :

- L'additionneur / Le soustracteur
- L'incrémenteur / Le décrémenteur
- Le décaleur
- L'Unité Arithmétique et Logique (UAL)

a. Demi-additionneur

- **Entrées** : les 2 bits à additionner a et b.
- **Sorties** :
  - La somme $S = a + b$.
  - La retenue de sortie R_{sortie}.
- **Rôle** : Additionner a et b en conservant la retenue.

a	b	S	R_{sortie}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

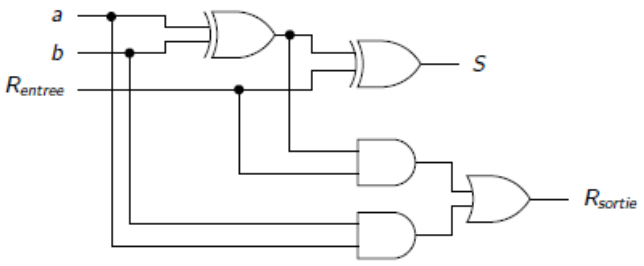


► Problème : si plusieurs additions successives, comment reporter la retenue ?

b. *Additionneur complet*

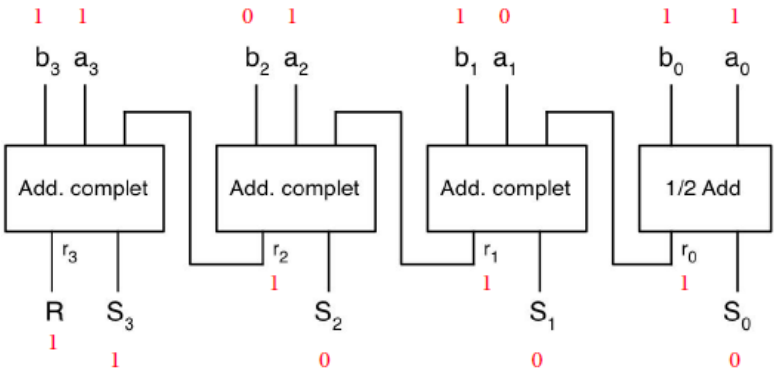
- **Entrées** :
  - Les 2 bits à additionner a et b.
  - La retenue d'entrée $R_{entrée}$
- **Sorties** :
  - La somme $S = a + b + R_{entrée}$
  - La retenue de sortie R_{sortie}
- **Rôle** : Additionner a et b en prenant en compte la retenue d'entrée $R_{entrée}$ et en conservant la retenue de sortie R_{sortie}

a	b	$R_{entrée}$	S	R_{sortie}
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



► Exemple : Additionneur 4 bits

Soit 2 nombres à additionner : $A = a_3\ a_2\ a_1\ a_0 = 1\ 1\ 0\ 1_{(2)}$ $B = b_3\ b_2\ b_1\ b_0 = 1\ 0\ 1\ 1_{(2)}$



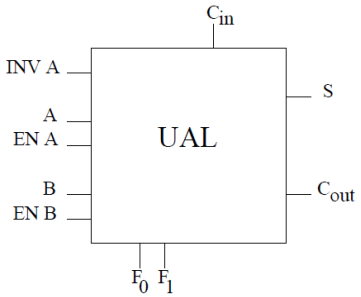
► $S = A + B = S_3\ S_2\ S_1\ S_0 = 1\ 0\ 0\ 0_{(2)}$ avec retenue $R = 1$.

c. *Unité Arithmétique et Logique*

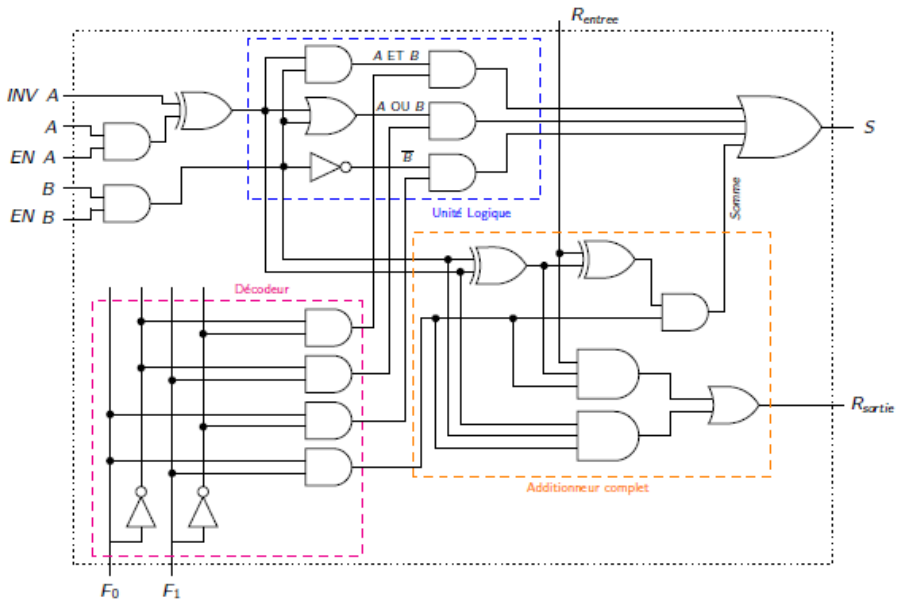
L'unité arithmétique et logique est l'organe responsable des calculs arithmétiques effectué par le processeur.

Elle est composée d'un ensemble de circuits combinatoires.

- **Entrées** :
  - A et B : les variables (données)
  - F_0 et F_1 : bits de choix du signal d'activation
  - $R_{entrée}$ (C_{in}) : la retenue d'entrée
  - EN A et EN B : les bits inhibiteurs de A et B (optionnel)
  - INV A : pour obtenir A (optionnel)



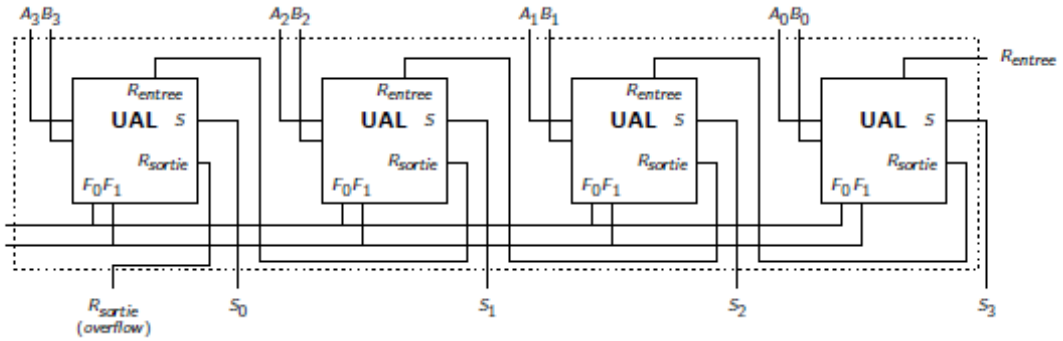
- **Sorties** :
  - S : le résultat de l'opération
  - R_{sortie} (C_{out}) : la retenue de sortie
- **Rôle** : Faire l'une des 4 opérations (en fonction des bits d'activation choisis) :
  - A ET B
  - A OU B
  - NON B
  - A + B + $R_{entrée}$



Pour 2 bits d'entrée, l'UAL est un circuit qui a peu d'intérêt . . .

► UAL à n bits :

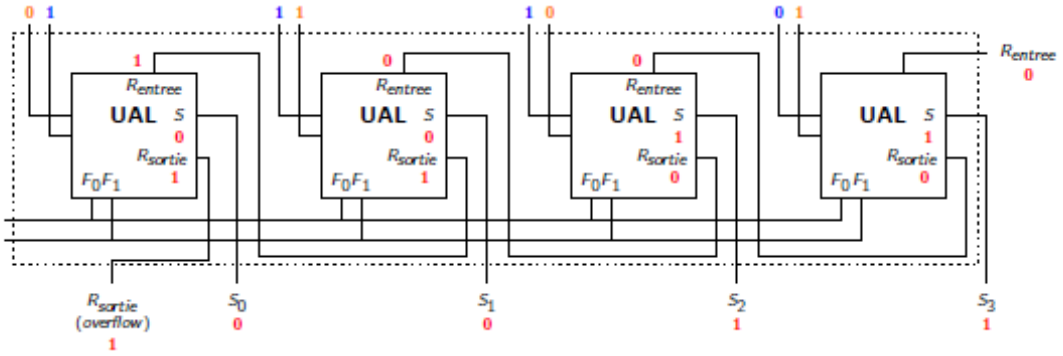
- En connectant les retenues de n UALs, on obtient une UAL n bits telle que :
  - Les opérations logiques sont des opérations bit à bit.
  - Les opérations arithmétiques sont effectuées sur des entiers en complément à 2 sur n bits.



► Exemple UAL à 4 bits :

On souhaite faire l'addition entre A et B (données) telle que :

$A = 14_{(10)} = a_3\ a_2\ a_1\ a_0 = 1\ 1\ 1\ 0_{(2)}$ $\quad$ $B = 5_{(10)} = b_3\ b_2\ b_1\ b_0 = 0\ 1\ 0\ 1_{(2)}$



$S = A + B = 14_{(10)} + 5_{(10)} = 19_{(10)} = 1\ 1\ 1\ 0_{(2)} + \ 0\ 1\ 0\ 1_{(2)} = 1\ 0011_{(2)}$

► UAL - Résumé des fonctions :

F_0	F_1	ENA	ENB	$INVA$	$R_{entrée}$	*Fonction*
0	0	1	1	0	0	$A\ ET\ B$
0	1	1	1	0	0	$A\ OU\ B$
0	1	0	0	0	0	0
0	1	0	1	0	0	B
0	1	1	0	0	0	A
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	$A + B$
1	1	0	0	0	1	1
1	1	0	0	1	0	-1
1	1	0	1	0	1	$B + 1$
1	1	0	1	1	0	$B - 1$
1	1	1	0	0	1	$A + 1$
1	1	1	0	1	1	$-A$
1	1	1	1	0	1	$A + B + 1$
1	1	1	1	1	1	$B - A$

4. *Circuits séquentiels*

a. Définitions

Dans un circuit combinatoire :

- Les valeurs de sorties, à un instant donné, sont imposées par celles des entrées.
- Le traitement des données est uniquement accessible immédiatement.
- La valeur de la sortie ne dépend que de l'entrée et pas de ce qui s'est passé auparavant.
- Applicable uniquement aux problèmes sans besoin de mémorisation.
- On sait traiter et manipuler l'information, comment la mémoriser ?

⇒ **Circuits séquentiels** (= circuits logiques à mémoire)

Un circuit séquentiel est circuit logique capable de mémoriser des informations. Sa sortie dépend de variables internes en plus des variables d'entrée. L'ensemble des informations mémorisées représente l'état du circuit. La modification des informations mémorisées implique la modification de l'état du circuit.

Les circuits séquentiels sont utilisés dans certains types de mémoires, tels que :

- Les bascules
- Les bascules latch
- Les bascules flip-flop
- Les registres

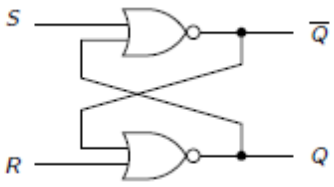
b. Les bascules

Une bascule :

- permet de mémoriser un bit.
- « se souvient » de la valeur que le circuit a enregistrée.
- est construite avec une ou deux portes logiques NON-OU (ou NON-ET).
- comporte une ou plusieurs entrées.
- comporte une ou deux sorties.

La sortie maintient son état même après disparition du signal de commande
⇒ **Logique séquentielle**

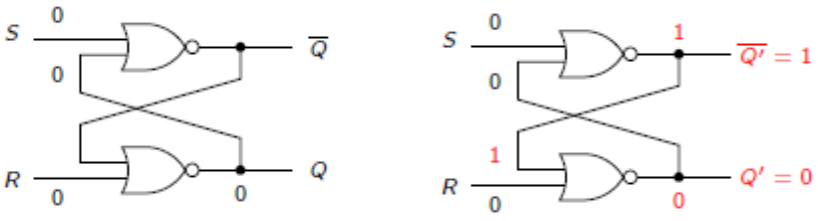
i. Bascule RS



- **Entrée** : deux variables d'entrée :
  - S (Set) pour la mise à l'état 1 de la bascule
  - R (Reset) pour la mise à l'état 0 de la bascule
- **Sortie** : deux variables de sortie : Q et $\bar{Q}$
- La valeur de sortie Q_n à l'instant $t = n$ dépend :
  - des variables d'entrées
  - de la valeur antérieure de la sortie (Q_{n-1})

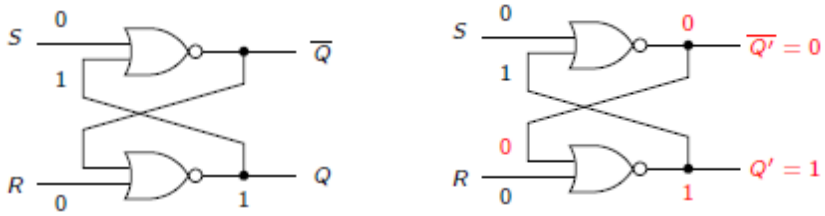
► Bascule RS : états stables

- Cas 1 : On suppose que $S = R = Q = 0 \Rightarrow \bar{Q}' = 1$ et $Q' = 0$



Bascule RS à l'état 0

- Cas 2 : On suppose que $S = R = 0$ et $Q = 1 \Rightarrow \bar{Q}' = 0$ et $Q' = 1$

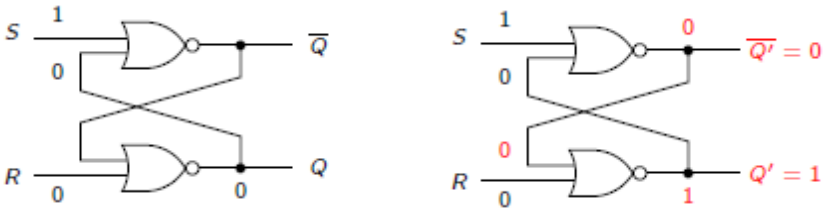


Bascule RS à l'état 1

- Les deux sorties Q' et $\bar{Q}'$ ne peuvent pas être simultanément à 0.
- Les deux sorties Q' et $\bar{Q}'$ ne peuvent pas être simultanément à 1.
- Pour $S = R = 0$, la bascule offre deux états stables qui dépendent de Q.

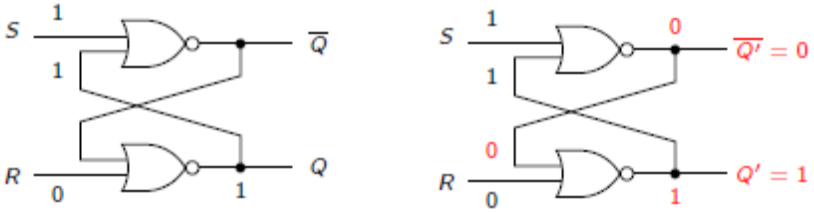
► Bascule RS : activation

- Cas 3.1 : On suppose que $S = 1$ et $R = Q = 0 \Rightarrow \bar{Q}' = 0$ et $Q' = 1$.



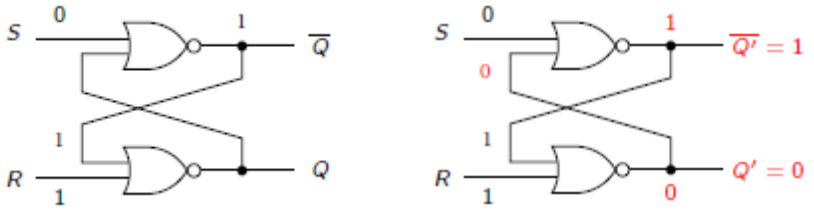
Bascule RS à l'état 1

- Cas 3.2 : On suppose que $S = Q = 1$ et $R = 0 \Rightarrow \bar{Q}' = 0$ et $Q' = 1$.



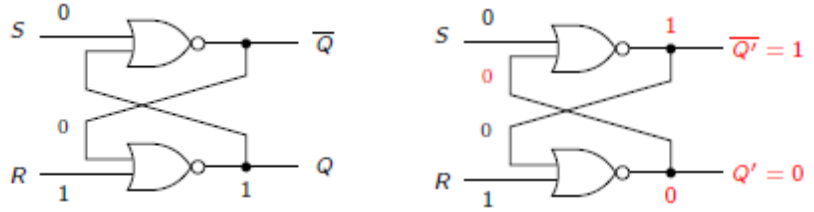
Bascule RS à l'état 1

- Cas 4.1 : On suppose que $S = Q = 0$ et $R = 1 \Rightarrow \bar{Q}' = 1$ et $Q' = 0$.



Bascule RS à l'état 0

- Cas 4.2 : On suppose que $S = 0$ et $R = Q = 1 \Rightarrow \bar{Q}' = 1$ et $Q' = 0$.



Bascule RS à l'état 0

- Si $S = 1$, la bascule RS passe (ou se maintient) à la valeur $Q' = 1$.
- Si $R = 1$, la bascule RS passe (ou se maintient) à la valeur $Q' = 0$.
- Une bascule RS « se souvient » de l'action antérieure de R ou S.

► Bascule RS : Table de vérité

S	R	Q	$\bar{Q}$	Q'	$\bar{Q}'$
0	0	0	0	x	x
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	x	x
0	1	0	0	x	x
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	x	x
1	0	0	0	x	x
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	x	x
1	1	0	0	x	x
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	x	x

S	R	Q'	$\overline{Q'}$	Etat de la bascule
0	0	Q	$\bar{Q}$	Sorties inchangées
0	1	0	1	RESET : remise à 0
1	0	1	0	SET : mise à1
1	1	0	0	Non utilisé (état instable)

La bascule RS mémorise la valeur des entrées : sa sortie dépend de la dernière entrée mise à 1 (R ou S).

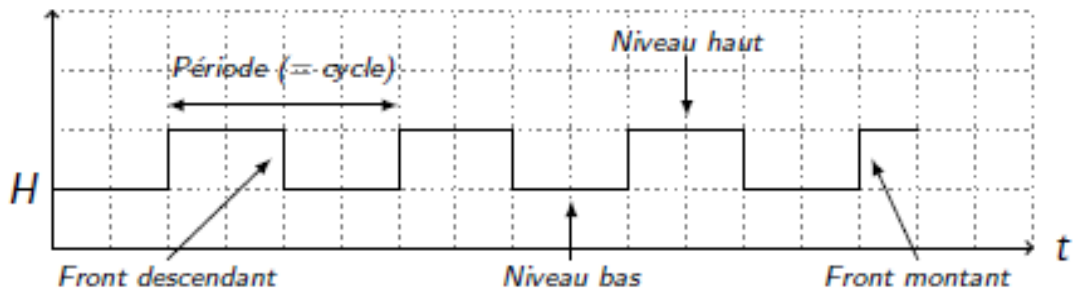
ii. Bascule RSH

L'ordre d'apparition des variables revêt une importance souvent cruciale. La conception des systèmes logiques dépend si une variable arrive avant l'autre ou bien si elles arrivent en même temps.

⇒ Besoin de respecter des relations de séquentialité contraignantes.
⇒ Utilisation d'**horloge** (base de temps ou système de cadencement).

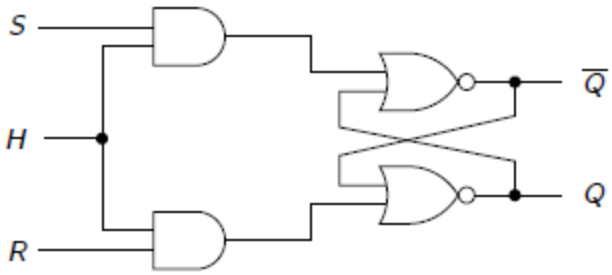
► Horloge :

- Système logique qui émet régulièrement une suite d'impulsions calibrées.
- Intervalle de temps entre deux impulsions = temps de cycle ou période de l'horloge.
- Fréquence des impulsions comprise entre 1 et 100 MHz.
- Temps de cycle compris entre 10 ns à 10 µs.



► Bascule RS + Horloge

- Permet de faire changer d'état à la bascule à un instant t précis.
- S_n et R_n : états des entrées à l'instant $t = n$.
- Q_{n+1} : sortie au prochain cycle d'horloge (instant $t = n + 1$)

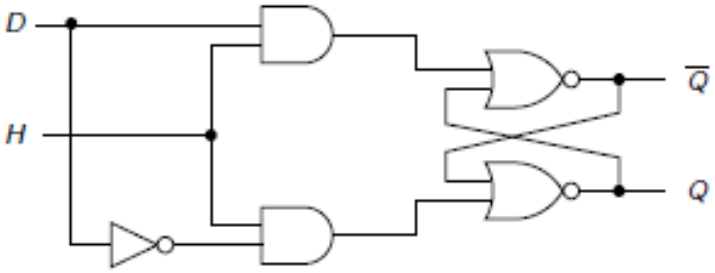


S_n	R_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	?

$$Q_{n+1} = S + \bar{R}Q_n$$

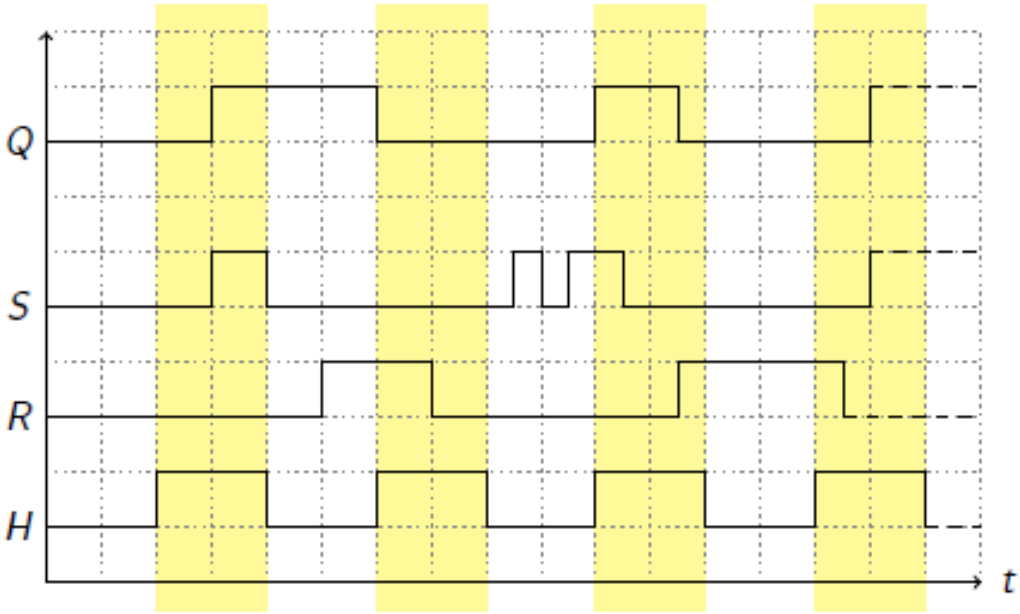
iii. Bascule D

- Pour résoudre l'ambiguïté propre à la bascule RS (quand $S = R = 1$)
- Fait en sorte que l'état correspondant à $S = R = 1$ ne soit jamais en entrée.
- Une seule entrée externe D.

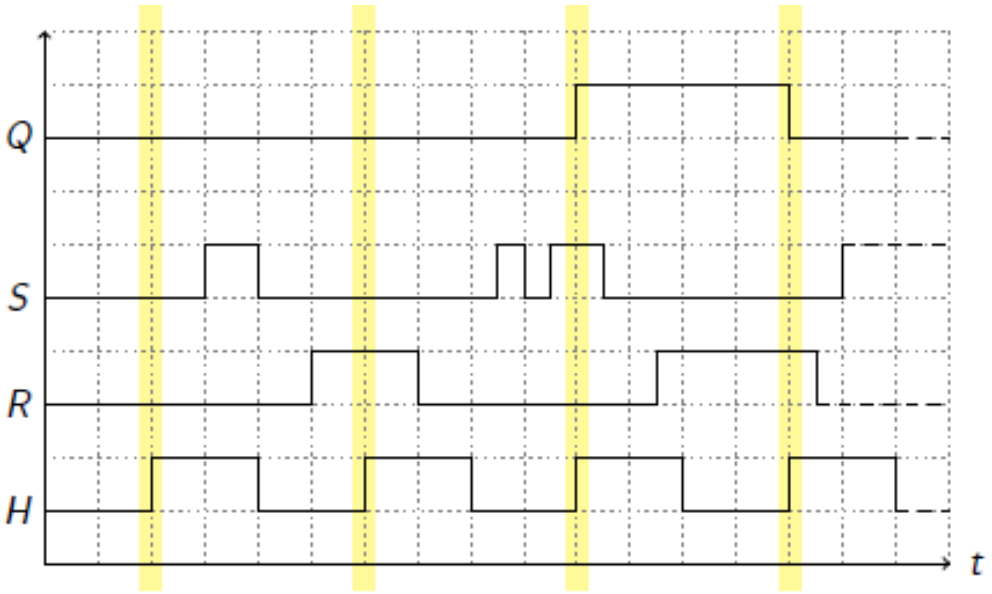


► Bascule latch :

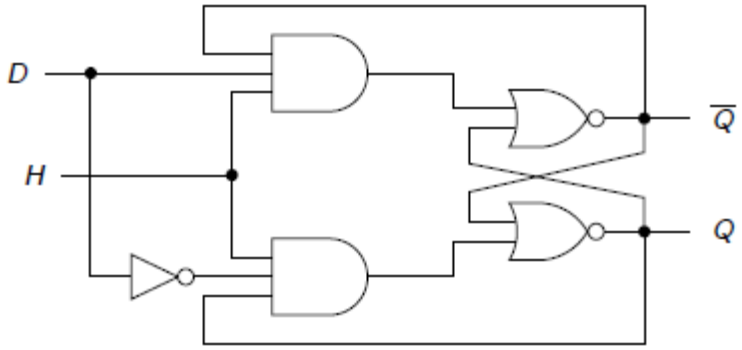
- Bascule asynchrone
- Change d'état lorsque l'horloge est au niveau 1 (= niveau haut)



- Bascule flip-flop :
  - Bascule synchrone
  - Change d'état lorsque l'horloge est en front montant

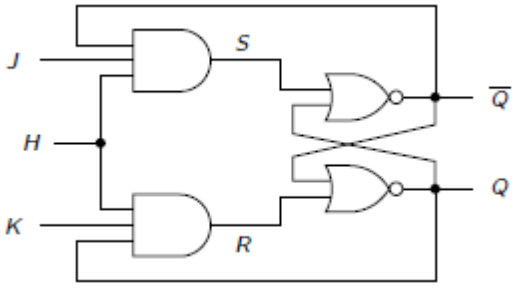


- Bascule D (flip-flop)



$$Q_{n+1} = D_n$$

iv. Bascule JK (flip-flop)



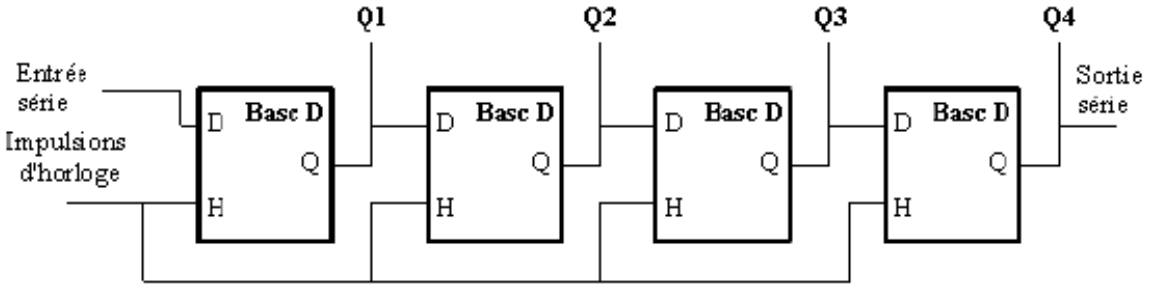
J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

J_n	K_n	Q_n	$\overline{Q_n}$	S	R	Q_{n+1}
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	1	0	1	1	0	1
1	1	1	0	0	1	0

$$Q_{n+1} = J_n\overline{Q_n} + \overline{K_n}Q_n$$

c. *Les registres*

- Une bascule est l'élément de base de la logique séquentielle.
- Une bascule permet de mémoriser un seul bit.
- Un registre est un ensemble ordonné de n bascules.
- Un registre permet de mémoriser une information sur n bits.



i. Types de registres

- Registres à chargement parallèle
- Registres à entrée/sortie série
- Registres à entrée série et sortie parallèle
- Registres à entrée parallèle et sortie série
- Registres à décalage circulaire

ii. Entrées asynchrones

Toutes les bascules précédentes peuvent se voir dotées d'entrées asynchrones permettant de forcer la valeur de la sortie :

- ***Preset*** : pour mettre la sortie Q à 1.
- ***Clear*** : pour mettre la sortie Q à 0.
- ***Enable*** : pour activer/désactiver le signal d'horloge.
- etc...