République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Tissemsilt
Ahmed ben yahia el-wancharissi

Faculté de Science et de la Technologie
Département de maths et informatique

Polycopié de cours

Structure Machine 2

Niveau : 1ère Année Licence (Mathématique et Informatique)

Enseignant : Dr. KIDAR Ali

Année Universitaire : ***2025/2026***

Chapitre I : Introduction Générale

Ce manuel de cours de structure machine 2 explique d'une façon simple et facile la structure et le fonctionnement de l'ordinateur en commençant par des notions de base. Ce polycopié s'adresse d'abord aux étudiants LMD (1ère année licence) socle commun Mathématique et Informatique, et, aux personnes qui s'intéressent à une connaissance de base aux circuits numériques et l'architecture des ordinateurs.

A l'issue de ces cours, l'étudiant(e) sera capable de :

- Comprendre la notion de circuits numériques.
- Apprendre l'algèbre de Boole et la simplification des fonctions logiques.
- Connaître les différentes fonctions intégrées de la logique combinatoire (Codage, décodage, transcodage, circuits arithmétiques et circuits d'aiguillage)
- Savoir les circuits logiques élémentaires (portes logiques, bascules).
- Comprendre le fonctionnement des bascules de base : D, T, JK et RS.
- Connaître les méthodes de synthèse de systèmes logiques combinatoires et séquentiels.
- Différencier entre les circuits logiques combinatoires et séquentiels usuels.
- Etablir le chronogramme d'un système séquentiel.
- Réaliser des circuits logiques combinatoires et séquentiels relatifs à son domaine de spécialité.
- apprenant acquerra les différents outils utilisés pour concevoir et réaliser des circuits et applications logiques combinatoires.
- apprenant acquerra les différents outils utilisés pour concevoir et réaliser des circuits et applications logiques séquentielles.
- Comprendre la notion de circuits intégrés.

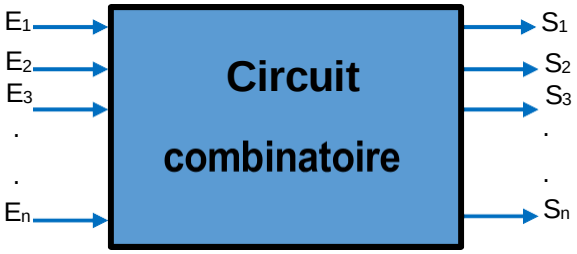
Dans ces cours, nous aborderons quelques concepts de base qui mènent à la conception des circuits numérique (combinatoires, séquentiels et intégrés). Nous traiterons, dans la première partie de ce polycopié les circuits logiques combinatoire, et nous ferons la synthèse de quelques uns de ces circuits comme circuits arithmétiques (l'additionneur, le soustracteur et le comparateur), codeur, décodeur, transcodage et circuits d'aiguillage (multiplexeur et démultiplexeur). Puis, nous présenterons certains circuits séquentiels très importants comme la mémoire, le registre ou le compteur dans lesquels la notion de chronologie des événements joue un rôle central. Puis, nous terminerons les circuits logiques séquentiel par présenter une méthode de synthèse en l'illustrant par la synthèse d'un compteur et décompter. La dernière partie porte sur la conception d'un circuit intégré, nous étudierons de façon générale les deux grands familles TTL et CMOS en donnant leurs principales caractéristiques. Dans ce qui suit, on détaille le programme de la matière.

Chapitre II : Les Circuits Combinatoires

2.1 Définition

Un circuit combinatoire est défini par une ou plusieurs fonctions logiques. Un circuit combinatoire est un circuit numérique dont les sorties dépendent uniquement des entrées.

- $S_i=F(E_i)$
- $S_i=F(E_1,E_2,\ldots,E_n)$



C'est possible d'utiliser des circuits combinatoires pour réaliser d'autres circuits plus complexes. Exemple de Circuits combinatoires : Demi Additionneur, Additionneur complet, Comparateur, Multiplexeur, Démultiplexeur, Encodeur et Décodeur…etc.

2.2. Synthèse d'une fonction combinatoire

Pour faire l'étude et la réalisation d'un circuit combinatoire il faut suivre les étapes suivantes :

- 1- Il faut bien comprendre le fonctionnement du système.
- 2- Il faut définir les variables d'entrée.
- 3- Il faut définir les variables de sortie.
- 4- Etablir la table de vérité.
- 5- Ecrire les équations algébriques des sorties (à partir de la table de vérité).
- 6- Effectuer des simplifications (algébrique ou par Karnaugh).
- 7- Faire le schéma avec un minimum de portes logiques.

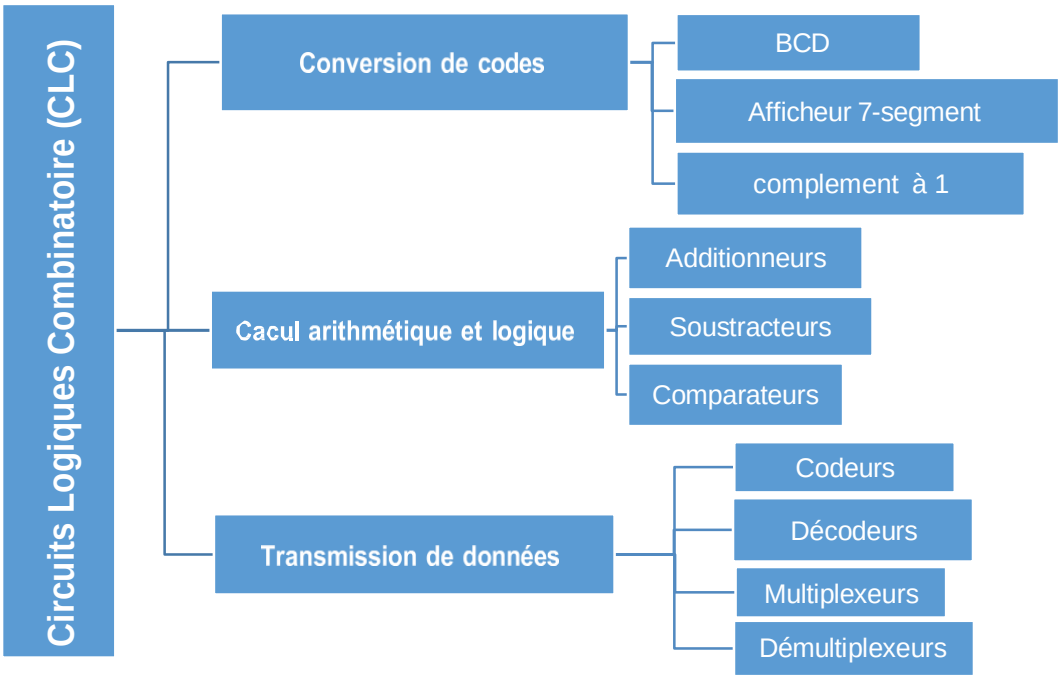
La synthèse d'un circuit combinatoire signifier la détermination d'un logigramme simplifier à partir de la définition d'une fonction logique. D'une façon générale, la démarche est la suivante :

- a- Schéma symbolique : construire son schéma symbolique en identifier les entrées et les sorties (In/Out) de la fonction.
- b- Table de vérité : construire sa table de vérité.
- c- Equations de sorties : extraire les équations de sorties à partir de la table de vérité, simplifier les fonctions de sortie via les théorèmes de l'algèbre de Boole ou les tables de Karnaugh
- d- Schéma logique : Dessiner le schéma du circuit logique à l'aide d'opérateurs (NOT, AND, OR, NAND, NOR)

2.3. Classification

On distingue au moins 3 classes de circuits logiques combinatoires :

- Les circuits de calcul arithmétiques et logiques ;
- Les circuits d'aiguillage et de transmission de données.



➲ Les circuits de calcul arithmétiques et logiques : Ce sont généralement des circuits logiques combinatoires permettant d'effectuer des calculs arithmétiques (addition, soustraction, multiplication) sur des entiers ou des nombres en virgule flottantes et des opérations logiques comme des négations, des ET, des OU ou des OU-Exclusifs. On les trouve le plus souvent dans les unités de calculs des ordinateurs communément appelées UAL ou unité arithmétique et logique.

➲ Les circuits de transmission de données : C'est un groupe de circuits permettant d'aiguiller les informations (données) binaires à travers des lignes électriques (souvent appelé BUS) d'une source (une petite mémoire appelée registre ou des capteurs, interrupteurs ou boutons poussoirs) vers une destination (registre ou un afficheur par exemple). Le décodeur, le multiplexeur en sont des exemples.

➲ Les convertisseurs de code Les nombres sont habituellement codés sous une forme ou une autre afin de les représenter ou de les utiliser au besoin. Par exemple, un nombre 'sept' est codé en décimal à l'aide du symbole $(7)_{10}$. Ce nombre est affiché sur votre calculatrice en se servant du codage 7 segments, mais au sein de l'unité de calcul de votre calculatrice, ce même nombre est codé en général en complément à 2. Bien que les ordinateurs numériques traitent tous des nombres binaires, il y a des situations où la représentation binaire naturelle des nombres n'est pas pratiques ou est inefficace ce qui nécessite des codes plus appropriés. Cette situation fait cohabiter, dans une même machine, divers codes pour représenter une même information. Des circuits de conversion d'un code vers un autre sont donc utiliser.

2.4. Circuits combinatoires usuelles

2.4.1. Additionneur

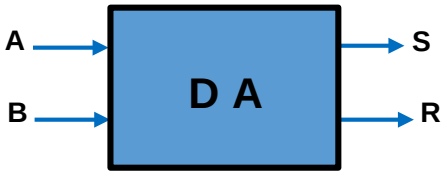
L'addition est une opération très courante dans un microprocesseur. Outre dans l'unité arithmétique, elle sert pour incrémenter le compteur de programme et pour les calculs d'adresses. il existe de multiples façons de construire des additionneurs efficaces en temps et en nombre de portes logiques utilisées [1].

2.4.1.1. Demi-Additionneur

• Le demi additionneur est un circuit combinatoire qui permet de réaliser la somme arithmétique de deux nombres A et B chacun sur un bit.

• A la sotie on va avoir la somme S et la retenue R (Carry).

a- Schéma symbolique :



En binaire l'addition sur un seul bit se fait de la manière suivante :

$$\begin{cases} 0+0=0 & \text{retenue}=0 \\ 0+1=1 & \text{retenue}=0 \\ 1+0=1 & \text{retenue}=0 \\ 1+1=0 & \text{retenue}=1 \end{cases}$$

b- Table de vérité :

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

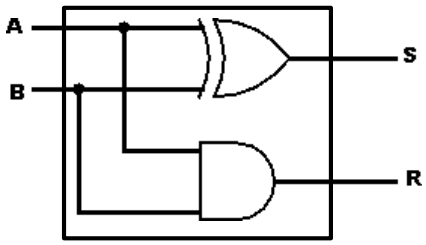
c- Equations de sortie :

De la table de vérité on trouve :

$$S = \overline{A}B + A\overline{B} = A \oplus B$$

$$R = AB$$

d- Schéma logique



Demi-additionneur à 1 bit

2.4.1.2. Additionneur complet

- En binaire lorsque on fait une addition il faut tenir en compte de la retenue entrante.

$$\begin{array}{ccccccc} R_4 & R_3 & R_2 & R_1 & R_0 = 0 & \qquad & R_{i-1} \\ & A_4 & A_3 & A_2 & A_1 & & A_i \\ + & B_4 & B_3 & B_2 & B_1 & + & B_i \\ \hline R_4 & S_4 & S_3 & S_2 & S_1 & & R_i \quad S_i \end{array}$$

Exemple d'un additionneur complet 1 bit

- L'additionneur complet un bit possède 3 entrées :

  A_i : le premier nombre sur un bit.

  B_i : le deuxième nombre sur un bit.

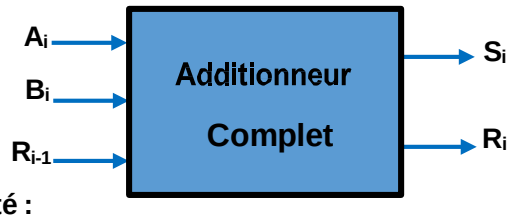
  R_{i-1} : la retenue entrante sur un bit.

- Il possède deux sorties :

  S_i : la somme

  R_i : la retenue sortante

a- Schéma symbolique



b- Table de vérité :

A_i	B_i	R_{i-1}	S_i	R_i
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

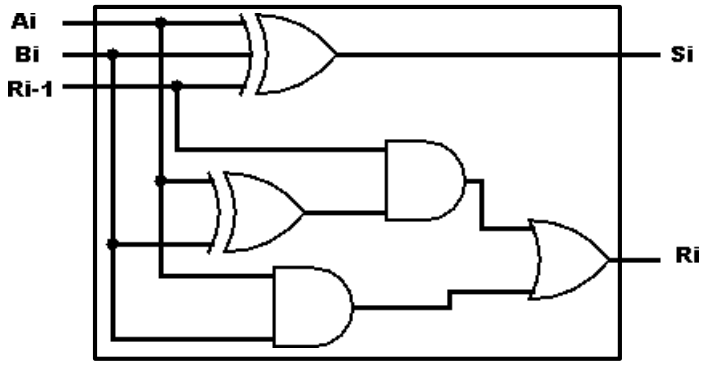
c- Equations de sortie :

De la table de vérité on trouve :

$$\begin{aligned} S_i &= \cancel{A_i}\cancel{B_i}R_{i-1} + \cancel{A_i}B_i\cancel{R_{i-1}} + A_i\cancel{B_i}\cancel{R_{i-1}} + A_iB_iR_{i-1} \\ &= \cancel{A_i}\left(\cancel{B_i}R_{i-1} + B_i\cancel{R_{i-1}}\right) + A_i\left(\cancel{B_i}\cancel{R_{i-1}} + B_iR_{i-1}\right) \\ &= \cancel{A_i}\left(B_i \oplus \cancel{R_{i-1}}\right) + A_i\left(\cancel{B_i} \oplus R_{i-1}\right) \\ &= A_i \oplus B_i \oplus R_{i-1} \end{aligned}$$

$$R_i = \cancel{A_i}B_iR_{i-1} + A_i\cancel{B_i}R_{i-1} + A_iB_i\cancel{R_{i-1}} + A_iB_iR_{i-1}$$
$$= R_{i-1}\left(\cancel{A_i}B_i + A_i\cancel{B_i}\right) + A_iB_i\left(\cancel{R_{i-1}} + R_{i-1}\right)$$
$$= R_{i-1}\left(A_i \oplus B_i\right) + A_iB_i$$

d- Schéma logique :



Additionneur complet à 1 bit

2.4.1.3. Additionneur à 4 bits

• Un additionneur à 4 bits est un circuit qui permet de faire l'addition de deux nombres A et B de 4 bits chacun : A(A4A3A2A1) et B(B4B3B2B1) En plus il tient en compte de la retenu entrante.

• En sortie on va avoir le résultat sur 4 bits ainsi que la retenue (5 bits en sortie)

• Donc au total le circuit possède 9 entrées et 5 sorties.

• Avec 9 entrées on a 2^9 =512 combinaisons

• Il faut trouver une solution plus facile et plus efficace pour concevoir ce circuit ?

• Lorsque on fait l'addition en binaire, on additionne bit par bit en commençant à partir du poids fiable et à chaque fois on propage la retenue sortante au bit du rang supérieur. L'addition sur un bit peut se faire par un additionneur complet sur 1 bit.

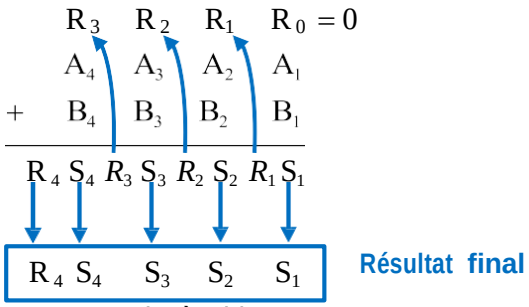
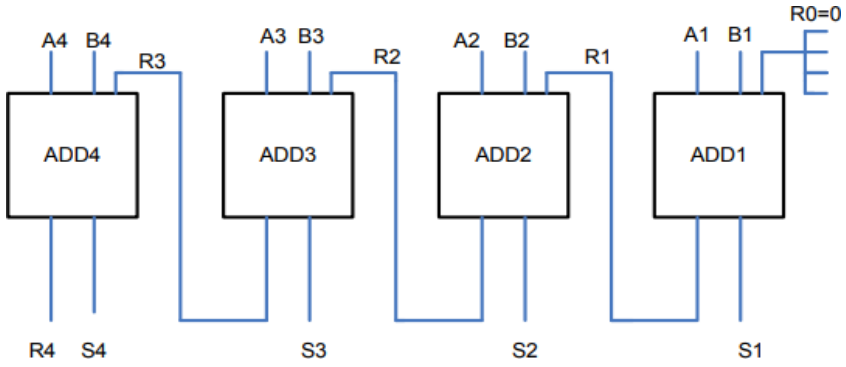


Schéma logique d'un additionneur complet à 4 bits



Additionneur n-bits. Pour effectuer l'addition de deux nombres de n bits, il suffit de chaîner entre eux n additionneurs 1-bit complets. La retenue est ainsi propagée d'un additionneur à l'autre. Un tel additionneur est appelé un additionneur série. Bien que tous les chiffres des deux nombres de n-bits X et Y soient disponibles simultanément au début du calcul, à t=0, le temps de calcul est déterminé par la propagation de la retenue à travers les n additionneurs 1-bit [2].

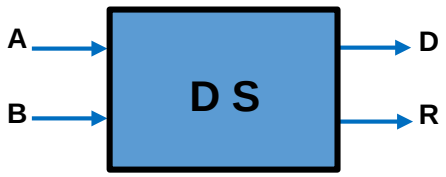
2.4.2. Soustracteur

Pour une soustraction de A et B, on peut adopter la même approche que pour l'addition. On commence par la définition de l'opérateur binaire de base et on l'utilise pour réaliser des soustractions de nombres binaires. En pratique, se pose le problème de la représentation des nombres signés dans le cas où B > A. Pour résoudre ce problème, on convient d'une représentation des nombres négatifs, la soustraction est alors ramenée à une addition. La représentation généralement utilisée est celle du complément vrai ou complément à 2.

2.4.2.1. Demi-soustracteur

• Le demi soustracteur est un circuit combinatoire qui permet de réaliser la différence arithmétique de deux nombres A et B chacun sur un bit.
• A la sotie on va avoir la différence D et la retenue R.

a- Schéma symbolique :



En binaire l'addition sur un seul bit se fait de la manière suivante :

$$\begin{cases} 0-0=0 & \text{retenue}=0 \\ 0-1=1 & \text{retenue}=1 \\ 1-0=1 & \text{retenue}=0 \\ 1-1=0 & \text{retenue}=0 \end{cases}$$

b- Table de vérité :

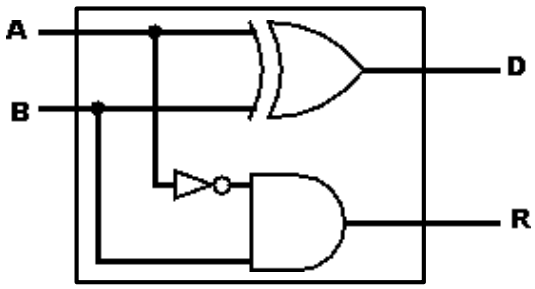
A	B	D	R
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

c- Equations de sortie :

De la table de vérité on trouve :

$$\begin{cases} D = \bar{A}B + A\bar{B} = A \oplus B \\ R = \bar{A}B \end{cases}$$

d-schéma logique :



Demi-soustracteur à 1 bit

2.4.2.2. Soustracteur complet

Pour obtenir un soustracteur binaire complet il faut prendre en compte l'éventuelle retenue précédente Ri-1.

• En binaire lorsque on fait une différence il faut tenir en compte de la retenue entrante.

R_4	R_3	R_2	R_1	$R_0 = 0$	
	A_4	A_3	A_2	A_1	
–	B_4	B_3	B_2	B_1	
	D_4	D_3	D_2	D_1	

	R_{i-1}	
	A_i	
–	B_i	
R_i	D_i	

Exemple d'un soustracteur complet 1 bit

• Le soustracteur complet un bit possède 3 entrées :

A_i : le premier nombre sur un bit.

B_i : le deuxième nombre sur un bit.

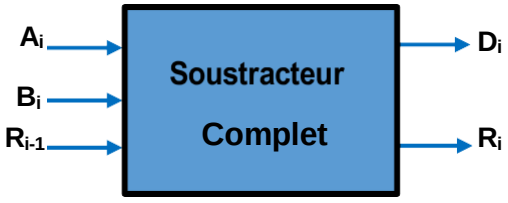
R_{i-1} : la retenue entrante sur un bit.

• Il possède deux sorties :

D_i : la différence

R_i : la retenue sortante

a- Schéma symbolique



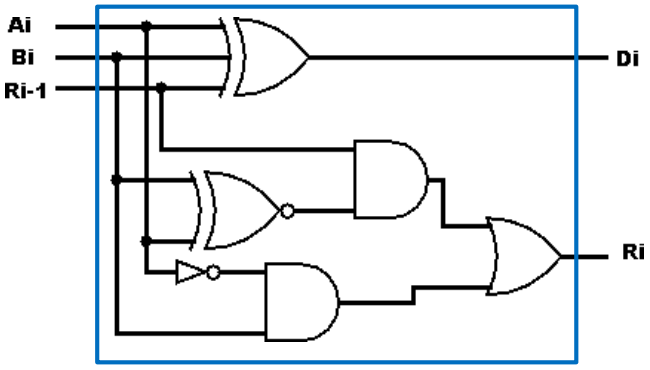
b- Table de vérité :

A_i	B_i	R_{i-1}			D_i	R_i
0	0	0			0	0
0	0	1			1	1
0	1	0			1	1
0	1	1			0	1
1	0	0			1	0
1	0	1			0	0
1	1	0			0	0
1	1	1			1	1

c- Equations de sortie :

$$D_i = \overline{A_i}\,\overline{B_i}\,R_{i-1} + \overline{A_i}\,B_i\,\overline{R_{i-1}} + A_i\,\overline{B_i}\,\overline{R_{i-1}} + A_i\,B_i\,R_{i-1}$$
$$= \overline{A_i}\left(\overline{B_i}\,R_{i-1} + B_i\,\overline{R_{i-1}}\right) + A_i\left(\overline{B_i}\,\overline{R_{i-1}} + B_i\,R_{i-1}\right)$$
$$= \overline{A_i}\left(B_i \oplus R_{i-1}\right) + A_i\left(\overline{B_i \oplus R_{i-1}}\right)$$
$$= A_i \oplus B_i \oplus R_{i-1}$$
$$R_i = \overline{A_i}\,\overline{B_i}\,R_{i-1} + \overline{A_i}\,B_i\,\overline{R_{i-1}} + \overline{A_i}\,B_i\,R_{i-1} + A_i\,B_i\,R_{i-1}$$
$$= R_{i-1}\left(\overline{A_i}\,\overline{B_i} + A_i\,B_i\right) + \overline{A_i}\,B_i\left(\overline{R_{i-1}} + R_{i-1}\right)$$
$$= R_{i-1}\left(\overline{A_i \oplus B_i}\right) + \overline{A_i}\,B_i$$

d- Schéma logique :



Soustracteur complet à 1 bit

2.4.2.3. Soustracteur à 4 bits

- Un soustracteur à 4 bits est un circuit qui permet de faire la différence de deux nombres A et B de 4 bits chacun : A($A_4A_3A_2A_1$) et B($B_4B_3B_2B_1$) En plus il tient en compte de la retenu entrante
- En sortie on va avoir le résultat sur 4 bits ainsi que la retenue (5 bits en sortie)
- Donc au total le circuit possède 9 entrées et 5 sorties.
- Avec 9 entrées on a 2^9 =512 combinaisons
- Il faut trouver une solution plus facile et plus efficace pour concevoir ce circuit ?

•Lorsque on fait la soustraction en binaire, on soustrait bit par bit en commençant à partir du poids fiable et à chaque fois on propage la retenue sortante au bit du rang supérieur. La soustraction sur un bit peut se faire par un soustracteur complet sur 1 bit.

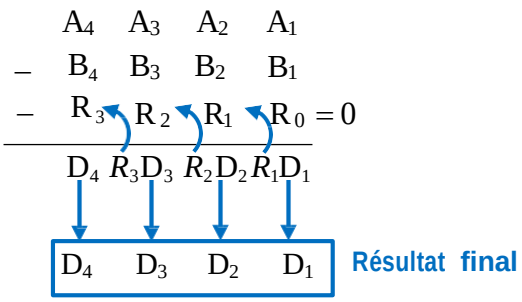
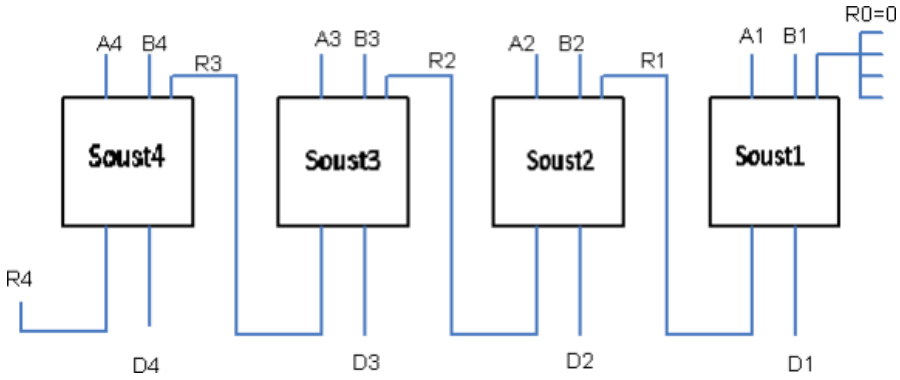


Schéma logique d'un soustracteur complet à 4 bits



2.4.3. Additionneur/Soustracteur

Le circuit de 5 bits ci-dessous effectue une somme ou une différence suivant la valeur de la commande C_{md}, Figure 3. Si Cmd vaut 0, le circuit calcule la somme A+B. Si, au contraire, C_{md} vaut 1, le circuit calcule la différence A-B. En effet, chacune des portes Xor effectue la négation ou non d'une entrée Bi suivant la valeur de C_{md}.

2.4.3. Additionneur/Soustracteur

Le circuit de 5 bits ci-dessous effectue une somme ou une différence suivant la valeur de la commande Cmd, Figure 3. Si Cmd vaut 0, le circuit calcule la somme A+B. Si, au contraire, Cmd vaut 1, le circuit calcule la différence A-B. En effet, chacune des portes Xor effectue la négation ou non d'une entrée Bi suivant la valeur d'entrée de sélection.

2.4.4. Comparateur

2.4.4.1. Comparateur à 1 bit

• C'est un circuit combinatoire qui permet de comparer entre deux nombres binaire A et B. Exemple d'un comparateur à un bit.

• Il possède 2 entrées :

A : sur un bit

B : sur un bit

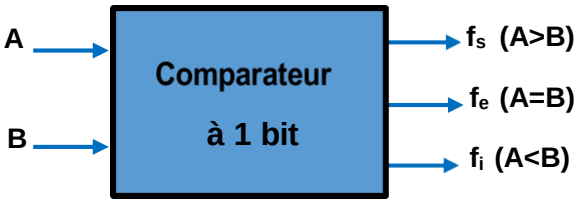
• Il possède 3 sorties

fe : égalité (A=B)

fi : inférieur (A < B)

fs : supérieur (A > B)

a- Schéma symbolique :



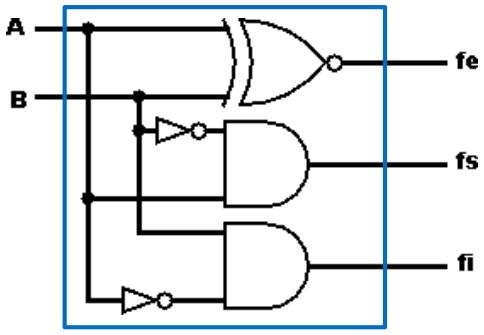
b- Table de vérité :

A	B			fs	fe	fi
0	0			0	1	0
0	1			0	0	1
1	0			1	0	0
1	1			0	1	0

c- Equations de sortie :

$$f_s = A\overline{B}$$
$$f_i = \overline{A}B$$
$$f_e = \overline{A}\,\overline{B} + AB = \overline{A \oplus B}$$

d- Schéma logique :



Comparateur à 1 bit

2.4.4.2. Comparateur à 2 bits

• Il permet de faire la comparaison entre deux nombres A(A_2A_1) et B(B_2B_1) chacun sur deux bits.

a- Schéma symbolique :

b- Table de vérité :

A_2	A_1	B_2	B_1	f_s	f_e	f_i
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

c- Equations de sortie :

$$f_s = A_2\overline{A_1}\overline{B_2}\overline{B_1} + A_2A_1\overline{B_2}\overline{B_1} + A_2\overline{A_1}B_2\overline{B_1} + A_2A_1\overline{B_2}B_1 + A_2\overline{A_1}\overline{B_2}B_1 + A_2A_1B_2\overline{B_1}$$

$$= A\overline{B} + \left(\overline{A \oplus B}\right)A\overline{B}$$

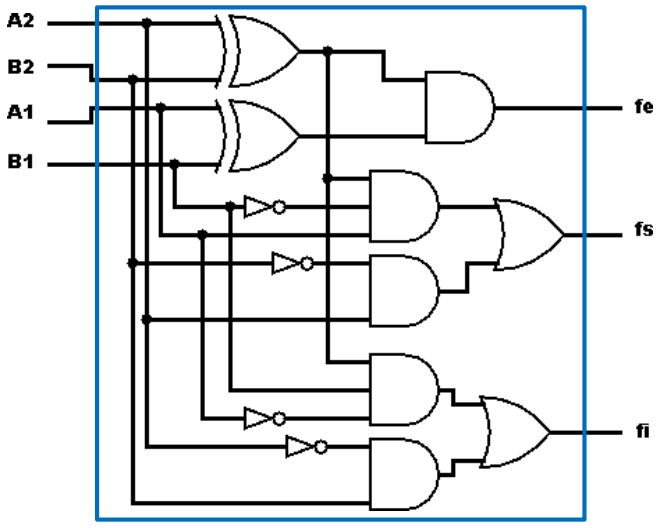
$$f_i = \overline{A_2}\overline{A_1}\overline{B_2}B_1 + \overline{A_2}\overline{A_1}B_2\overline{B_1} + \overline{A_2}A_1B_2B_1 + \overline{A_2}A_1B_2\overline{B_1} + \overline{A_2}A_1\overline{B_2}B_1 + A_2\overline{A_1}B_2B_1$$

$$= \overline{A}B + \left(\overline{A \oplus B}\right)\overline{A}B$$

$$f_e = A_2\overline{A_1}\overline{B_2}\overline{B_1} + \overline{A_2}A_1\overline{B_2}B_1 + A_2\overline{A_1}B_2\overline{B_1} + A_2A_1B_2B_1$$

$$= \left(\overline{A \oplus B}\right)\left(\overline{A \oplus B}\right)$$

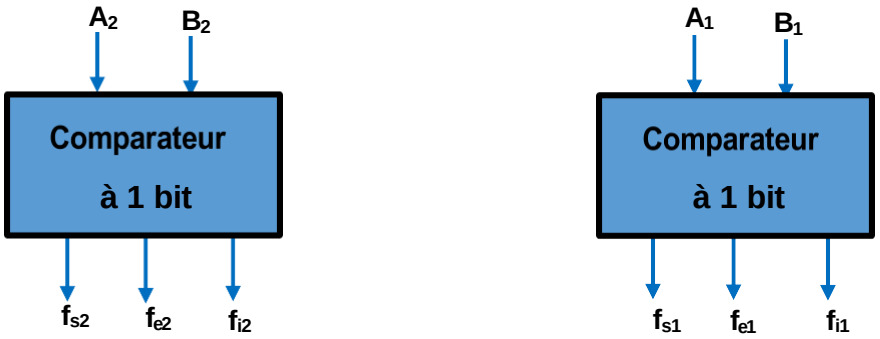
d- Schéma logique :



Comparateur à 2 bit

2.4.4.3. Comparateur 2 bits avec des comparateurs 1 bit

C'est possible de réaliser un comparateur 2 bits en utilisant des comparateurs 1 bit et des portes logiques. Il faut utiliser un comparateur pour comparer les bits du poids faible et un autre pour comparer les bits du poids fort. Il faut combiner entre les sorties des deux comparateurs utilisés pour réaliser les sorties du comparateur final.

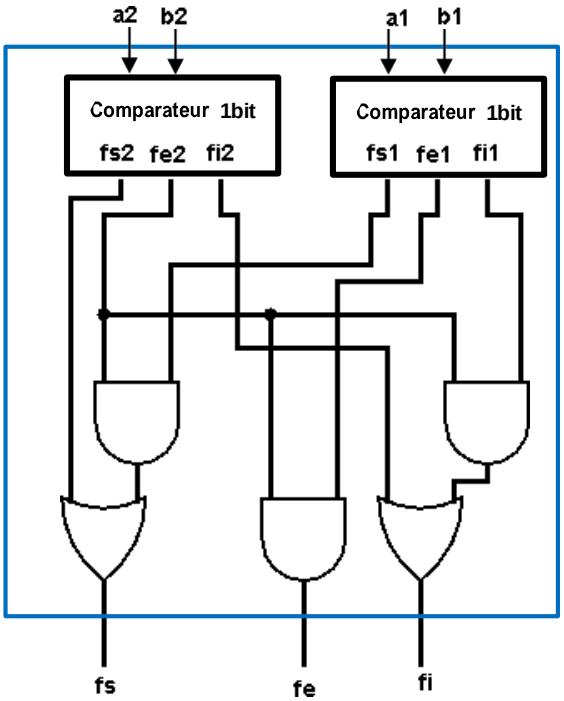


$$A = B \text{ si } A_2 = B_2 \text{ et } A_1 = B_1 \longrightarrow f_e = \left(\overline{A_2 \oplus B_2}\right)\left(\overline{A_1 \oplus B_1}\right) = f_{e2} f_{e1}$$

$$A > B \text{ si } A_2 > B_2 \text{ ou } \left(A_2 = B_2 \text{ et } A_1 > B_1\right) \longrightarrow f_s = A_2 \overline{B_2} + \left(\overline{A_2 \oplus B_2}\right) A_1 \overline{B_1} = f_{s2} + f_{e2} f_{s1}$$

$$A < B \text{ si } A_2 < B_2 \text{ ou } \left(A_2 = B_2 \text{ et } A_1 < B_1\right) \longrightarrow f_i = \overline{A_2} B_2 + \left(\overline{A_2 \oplus B_2}\right) \overline{A_1} B_1 = f_{i2} + f_{e2} f_{i1}$$

Schéma logique d'un comparateur à deux entrées à deux bits



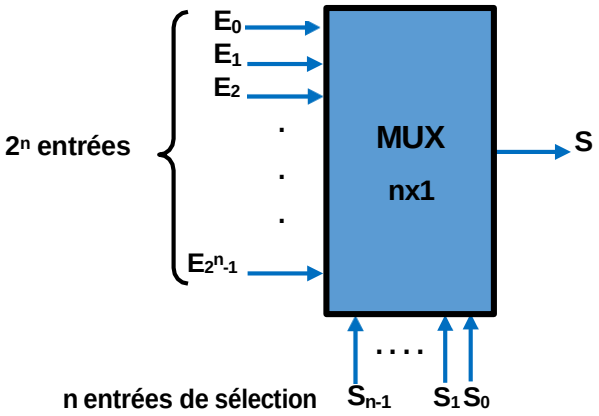
2.4.5. Multiplexeur

Un multiplexeur possède plusieurs entrées et une seule sortie. Il agit comme un sélecteur de données en orientant vers sa sortie la donnée présente sur l'une de ses entrées [3]. Un multiplexeur

est un circuit combinatoire qui permet de sélectionner une information (1 bit) parmi 2^n valeurs en entrée.

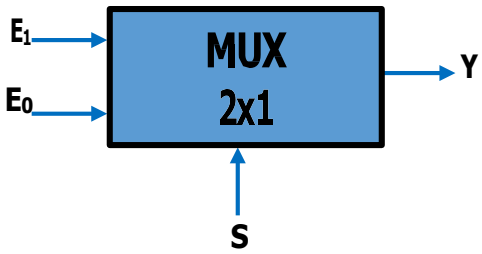
Il possède :

- 2^N entrées d'information
- Une seule sortie
- N entrées de sélection (commandes)



2.4.5.1. Multiplexeur 2x1

a-Schéma symbolique:



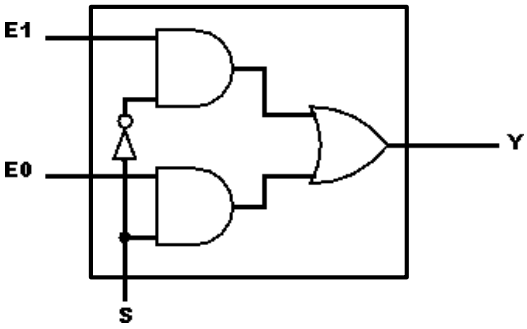
b- Table de vérité :

S		Y
0		E_0
1		E_1

c- Equations de sortie :

$$Y = \overline{S}E_0 + S\overline{E_1}$$

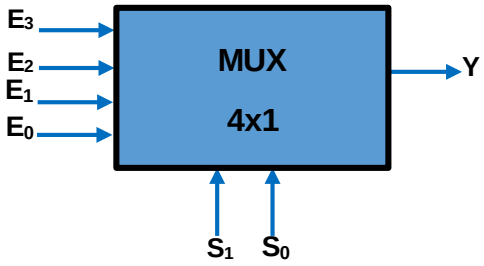
d- Schéma logique :



Multiplexeur 2x1

2.4.5.2. Multiplexeur 4X1

a- Schéma symbolique :



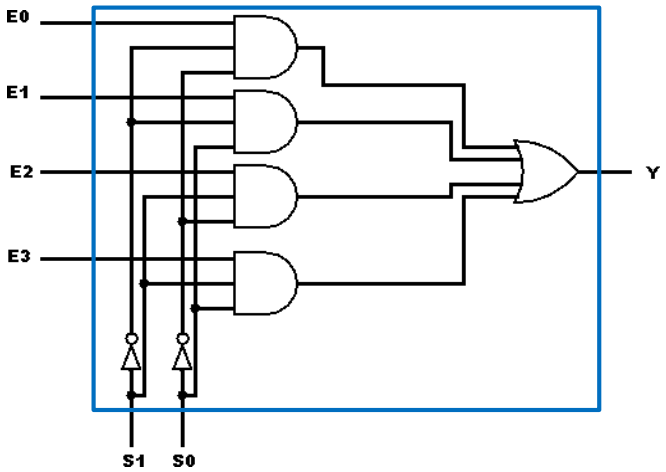
b- Table de vérité :

S_1	S_0	Y
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

c- Equations de sortie :

$$Y = \overline{S}_1\overline{S}_0E_0 + \overline{S}_1S_0E_1 + S_1\overline{S}_0E_2 + S_1S_0E_3$$

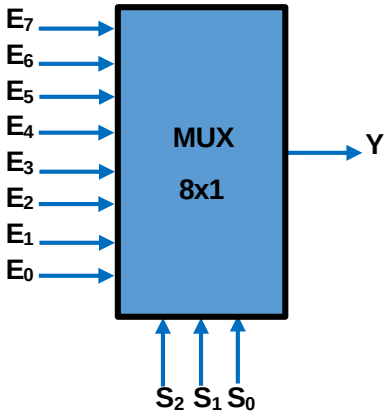
d- Schéma logique :



Multiplexeur 4x1

2.4.5.3. Multiplexeur 8X1

a-schéma symbolique :



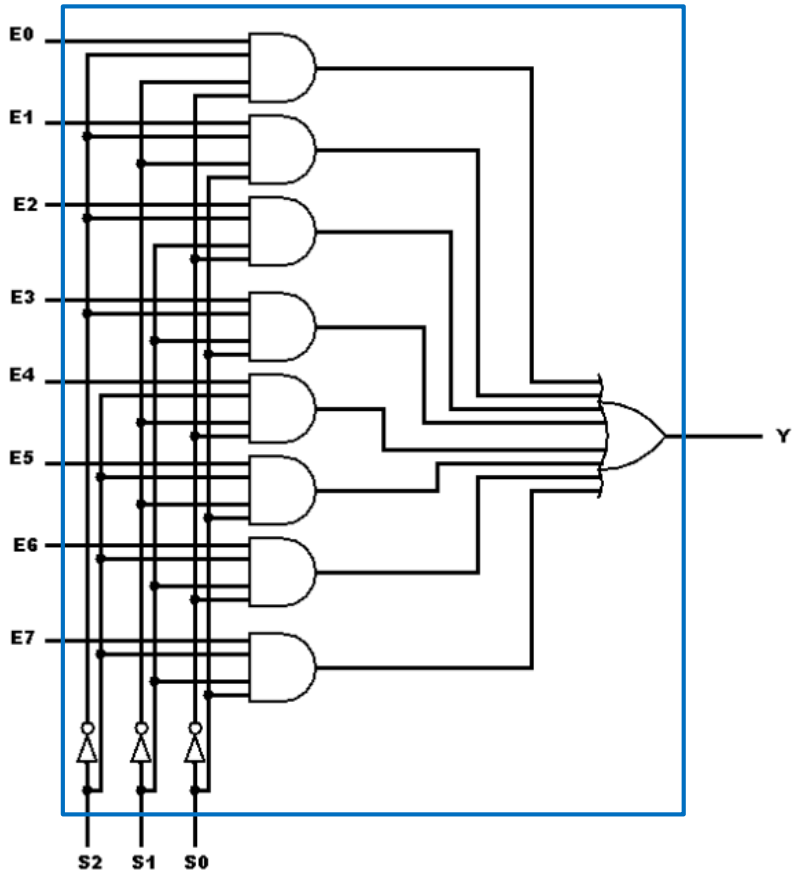
b-Table de vérité :

S_2	S_1	S_0			Y
0	0	0			E_0
0	0	1			E_1
0	1	0			E_2
0	1	1			E_3
1	0	0			E_4
1	0	1			E_5
1	1	0			E_6
1	1	1			E_7

c- équations de sortie :

$$Y = \bar{S_2}\bar{S_1}\bar{S_0}E_0 + \bar{S_2}\bar{S_1}S_0E_1 + \bar{S_2}S_1\bar{S_0}E_2 + \bar{S_2}S_1S_0E_3 + S_2\bar{S_1}\bar{S_0}E_4 + S_2\bar{S_1}S_0E_5 + S_2S_1\bar{S_0}E_6 + S_2S_1S_0E_0$$

d-schéma logique :



Multiplexeur 8X1

Pour réaliser une fonction logique par un Mux on effectue les opérations suivantes [4] :

- On écrit l'équation de la fonction logique sous la première forme canonique (on détermine le nombre d'entrée).

- On écrit l'équation du Mux caractérisé par le nombre d'entrée d'adresse (nombre d'entrée d'adresse = nombre d'entrées de la fonction).

- Identification de deux équations.

Exemple : **Additionneur complet avec des multiplexeurs 8X1**

•Nous avons besoin d'utiliser deux multiplexeurs : Le premier pour réaliser la fonction de la somme et l'autre pour donner la retenue.

b- Table de vérité :

A_i	B_i	R_{i-1}	S_i	R_i
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

c- Equations de sortie :

-La fonction de la somme

$$S_i = \overline{A_i}\,\overline{B_i}\,\overline{R_{i-1}}(0) + A_i\overline{B_i}\,\overline{R_{i-1}}(1) + \overline{A_i}B_i\overline{R_{i-1}}(1) + A_iB_i\overline{R_{i-1}}(0) + A_i\overline{B_i}R_{i-1}(1) + \overline{A_i}B_iR_{i-1}(0) + A_iB_iR_{i-1}(0)$$

$$S_i = \overline{S_2}\,\overline{S_1}\,\overline{S_0}(E_0) + \overline{S_2}\,\overline{S_1}S_0(E_1) + \overline{S_2}S_1\overline{S_0}(E_2) + \overline{S_2}S_1S_0(E_3) + S_2\overline{S_1}\,\overline{S_0}(E_4) + S_2\overline{S_1}S_0(E_5) + S_2S_1\overline{S_0}(E_6) + S_2S_1S_0(E_7)$$

$$S_i = \overline{S_2}\,\overline{S_1}\,\overline{S_0}(0) + \overline{S_2}\,\overline{S_1}S_0(1) + \overline{S_2}S_1\overline{S_0}(1) + \overline{S_2}S_1S_0(0) + S_2\overline{S_1}\,\overline{S_0}(1) + S_2\overline{S_1}S_0(0) + S_2S_1\overline{S_0}(0) + S_2S_1S_0(1)$$

Tel que, on pose

$S_2 = A_i$

$S_1 = B_i$

$S_0 = R_{i-1}$

$E_0 = 0,\ E_1 = 1,\ E_2 = 1,\ E_3 = 0,\ E_4 = 1,\ E_5 = 0,\ E_6 = 0,\ E_7 = 1$

-La fonction de la retenue

$$R_i = \overline{A_i}\,\overline{B_i}\,\overline{R_{i-1}}(0) + \overline{A_i}\,\overline{B_i}R_{i-1}(0) + \overline{A_i}B_i\overline{R_{i-1}}(0) + A_i\overline{B_i}R_{i-1}(1) + A_iB_i\overline{R_{i-1}}(0) + A_i\overline{B_i}\,\overline{R_{i-1}}(1) + A_iB_iR_{i-1}(1)$$

$$R_i = \overline{S_2}\,\overline{S_1}\,\overline{S_0}(E_0) + \overline{S_2}\,\overline{S_1}S_0(E_1) + \overline{S_2}S_1\overline{S_0}(E_2) + \overline{S_2}S_1S_0(E_3) + S_2\overline{S_1}\,\overline{S_0}(E_4) + S_2\overline{S_1}S_0(E_5) + S_2S_1\overline{S_0}(E_6) + S_2S_1S_0(E_7)$$

$$R_i = \overline{S_2}\,\overline{S_1}\,\overline{S_0}(0) + \overline{S_2}\,\overline{S_1}S_0(0) + \overline{S_2}S_1\overline{S_0}(0) + \overline{S_2}S_1S_0(1) + S_2\overline{S_1}\,\overline{S_0}(0) + S_2\overline{S_1}S_0(1) + S_2S_1\overline{S_0}(1) + S_2S_1S_0(1)$$

Tel que, on pose

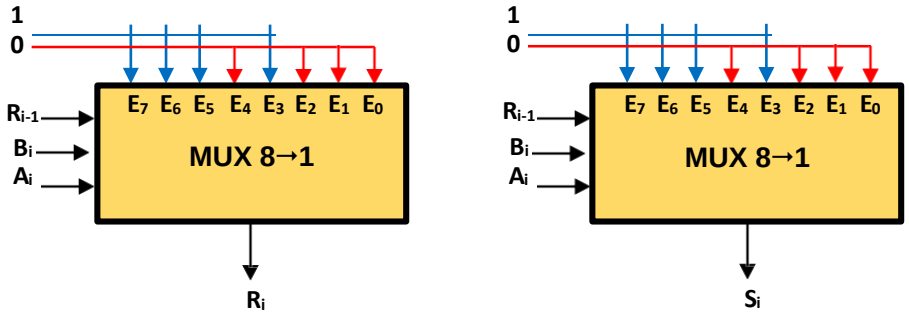
$S_2 = A_i$

$S_1 = B_i$

$S_0 = R_{i-1}$

$E_0 = 0,\ E_1 = 1,\ E_2 = 1,\ E_3 = 0,\ E_4 = 1,\ E_5 = 0,\ E_6 = 0,\ E_7 = 1$

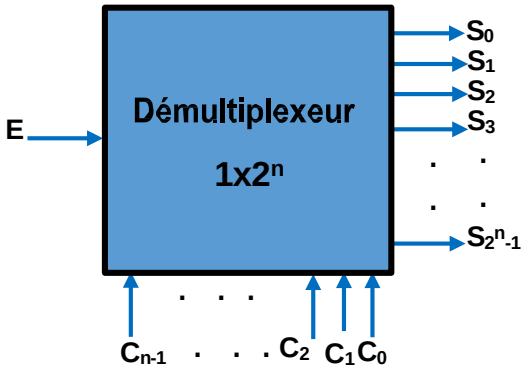
d- Schéma logique :



2.4.6. Démultiplexeurs

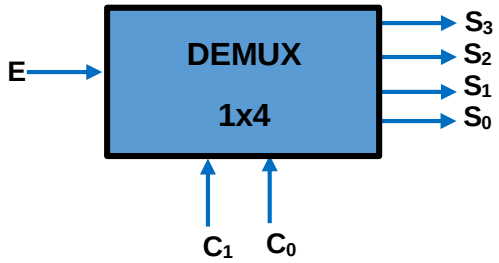
• Il joue le rôle inverse d'un multiplexeur, il permet de faire passer une information dans l'une des sorties selon les valeurs des entrées de commandes. Il possède :

- Une seule entrée
- 2^N sorties
- N entrées de sélection (commandes



2.4.6.1. Démultiplexeurs 1X4

a-schéma symbolique :



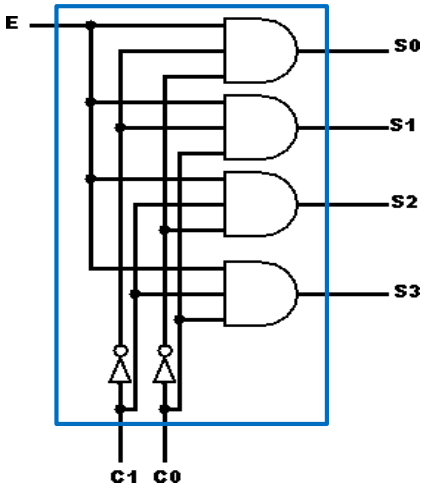
b- Table de vérité :

C_1	C_0		S_3	S_2	S_1	S_0
0	0		0	0	0	E
0	1		0	0	E	0
1	0		0	E	0	0
1	1		E	0	0	0

c- équations de sortie :

$$S_0 = \overline{C_1}\,\overline{C_0}\,E$$
$$S_1 = \overline{C_1} C_0\,E$$
$$S_2 = C_1\overline{C_0}\,E$$
$$S_3 = C_1 C_0 E$$

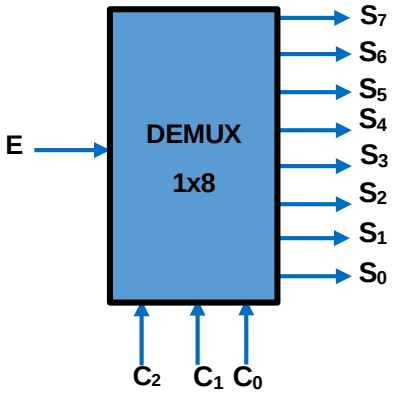
d- Schéma logique :



Démultiplexeurs 1X4

2.4.6.2. Démultiplexeurs 1X8

a- Schéma symbolique :



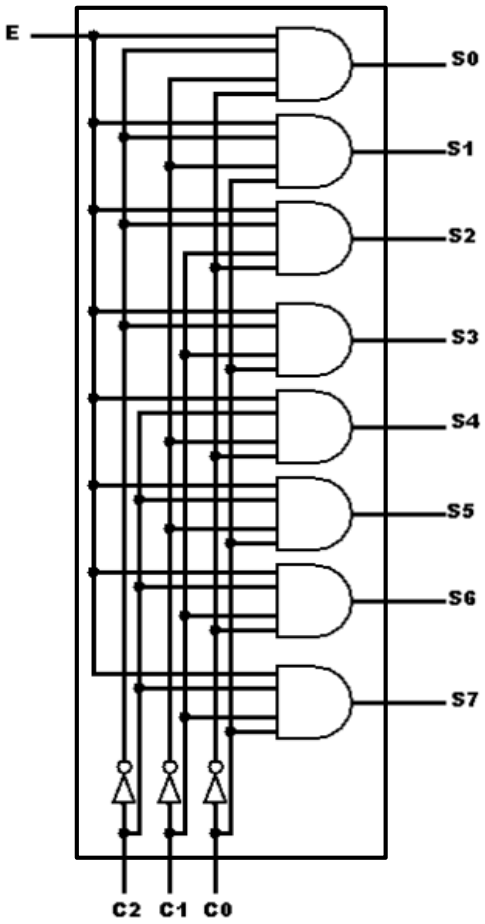
b- Table de vérité :

C_2	C_1	C_0	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	E
0	0	1	0	0	0	0	0	0	E	0
0	1	0	0	0	0	0	0	E	0	0
0	1	1	0	0	0	0	E	0	0	0
1	0	0	0	0	0	E	0	0	0	0
1	0	1	0	0	E	0	0	0	0	0
1	1	0	0	E	0	0	0	0	0	0
1	1	1	E	0	0	0	0	0	0	0

c- Equations de sortie :

$$S_0 = \overline{C_2}\,\overline{C_1}\,\overline{C_0}\,E$$
$$S_1 = \overline{C_2}\,\overline{C_1}\,C_0\,E$$
$$S_2 = \overline{C_2}\,C_1\,\overline{C_0}\,E$$
$$S_3 = \overline{C_2}\,C_1\,C_0\,E$$
$$S_4 = C_2\,\overline{C_1}\,\overline{C_0}\,E$$
$$S_5 = C_2\,\overline{C_1}\,C_0\,E$$
$$S_6 = C_2\,C_1\,\overline{C_0}\,E$$
$$S_7 = C_2 C_1 C_0 E$$

d- Schéma logique :

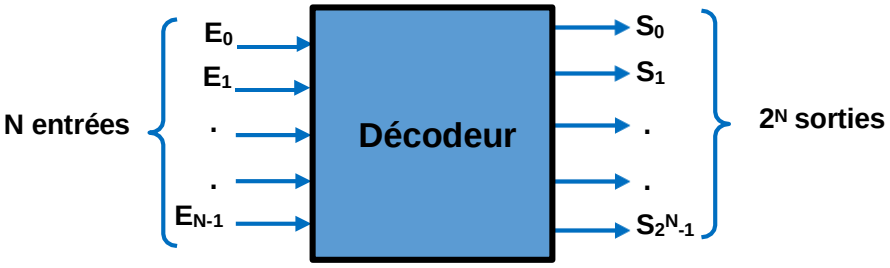


Démultiplexeurs 1X8

2.4.7. Décodeur binaire

C'est un circuit combinatoire qui est constitué de :

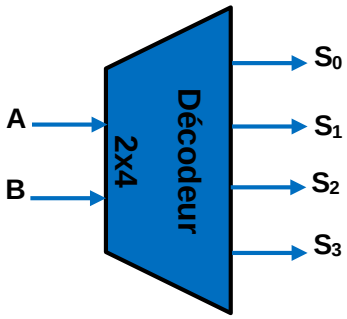
- N entrées de données
- 2^N sorties



• Pour chaque combinaison en entrée une seule sortie est active à la fois

2.4.7.1. Décodeur binaire 2x4

a-schéma symbolique :



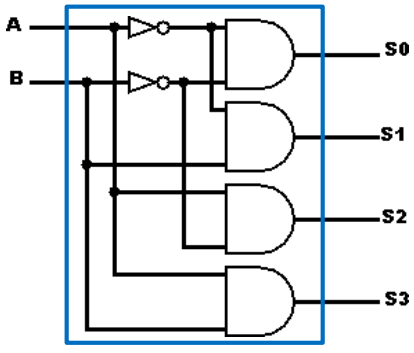
b-Table de vérité :

A	B		S0	S1	S2	S3
0	0		1	0	0	0
0	1		0	1	0	0
1	0		0	0	1	0
1	1		0	0	0	1

c- équations de sortie :

$$S_0 = \overline{A}\,\overline{B}$$
$$S_1 = \overline{A}B$$
$$S_2 = A\overline{B}$$
$$S_3 = AB$$

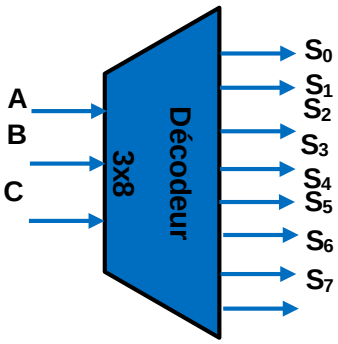
d-schéma logique :



Décodeur binaire 2x4

2.4.7.2. Décodeur binaire 3X8

a- Schéma symbolique :



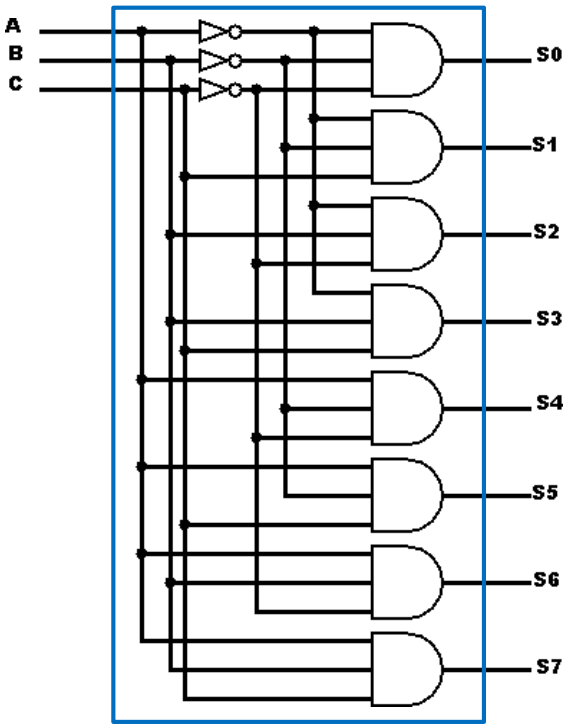
b- Table de vérité :

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

c- Equations de sortie :

$$S_0 = \bar{A}\bar{B}\bar{C}$$
$$S_1 = \bar{A}\bar{B}C$$
$$S_2 = \bar{A}B\bar{C}$$
$$S_3 = \bar{A}BC$$
$$S_4 = A\bar{B}\bar{C}$$
$$S_5 = A\bar{B}C$$
$$S_6 = AB\bar{C}$$
$$S_7 = ABC$$

d- Schéma logique :



Décodeur binaire 3X8

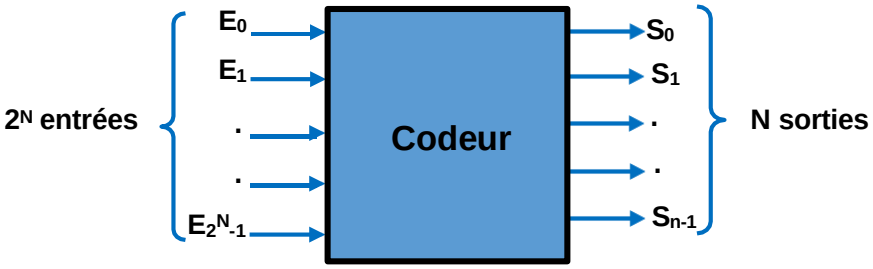
2.4.8. Encodeur binaire (codeur)

• Il joue le rôle inverse d'un décodeur. Il possède :

• 2^N entrées

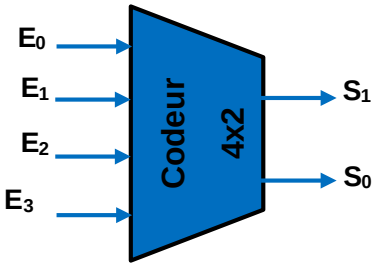
• N sorties

• Pour chaque combinaison en entrée on va avoir son numéro (en binaire) à la sortie.



2.4.8.1. Encodeur binaire 4x2

a- Schéma symbolique :



b- Table de vérité :

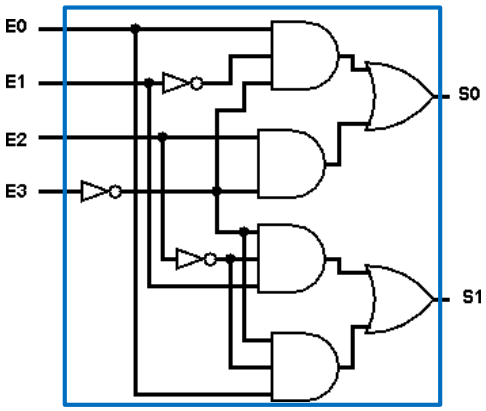
E_3	E_2	E_1	E_0		S_1	S_0
1	X	X	X		**0**	**0**
0	**1**	X	X		**0**	**1**
0	**0**	**1**	X		**1**	**0**
0	**0**	**0**	**1**		**1**	**1**

c- équations de sortie :

$$S_0 = \bar{E}_3 E_2 + \bar{E}_3 \bar{E}_2 \bar{E}_1 E_0 = \bar{E}_3 (E_2 + \bar{E}_2 \bar{E}_1 E_0) = \bar{E}_3 (E_2 + \bar{E}_1 E_0) = \bar{E}_3 E_2 + \bar{E}_3 \bar{E}_1 E_0$$

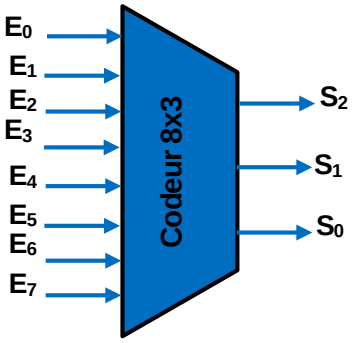
$$S_1 = \bar{E}_3 \bar{E}_2 E_1 + \bar{E}_3 \bar{E}_2 \bar{E}_1 E_0 = \bar{E}_3 \bar{E}_2 (E_1 + \bar{E}_1 E_0) = \bar{E}_3 \bar{E}_2 (E_1 + E_0) = \bar{E}_3 \bar{E}_2 E_1 + \bar{E}_3 \bar{E}_2 E_0$$

d- Schéma logique :



2.4.8.2. Encodeur binaire 8x3

a- Schéma symbolique :



b- Table de vérité :

E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0			S_2	S_1	S_0
1	**X**	X	X	X	X	X	X			**0**	**0**	**0**
0	**1**	X	X	X	X	X	X			**0**	**0**	**1**
0	**0**	**1**	X	X	X	X	X			**0**	**1**	
0	**0**	**0**	**1**	X	X	X	X			**0**	**1**	
0	**0**	**0**	**0**	**1**	X	X	X			**1**	**0**	
0	**0**	**0**	**0**	**0**	**1**	X	X			**1**	**0**	**1**
0	**0**	**0**	**0**	**0**	**0**	**1**	X			**1**	**1**	
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**			**1**	**1**	

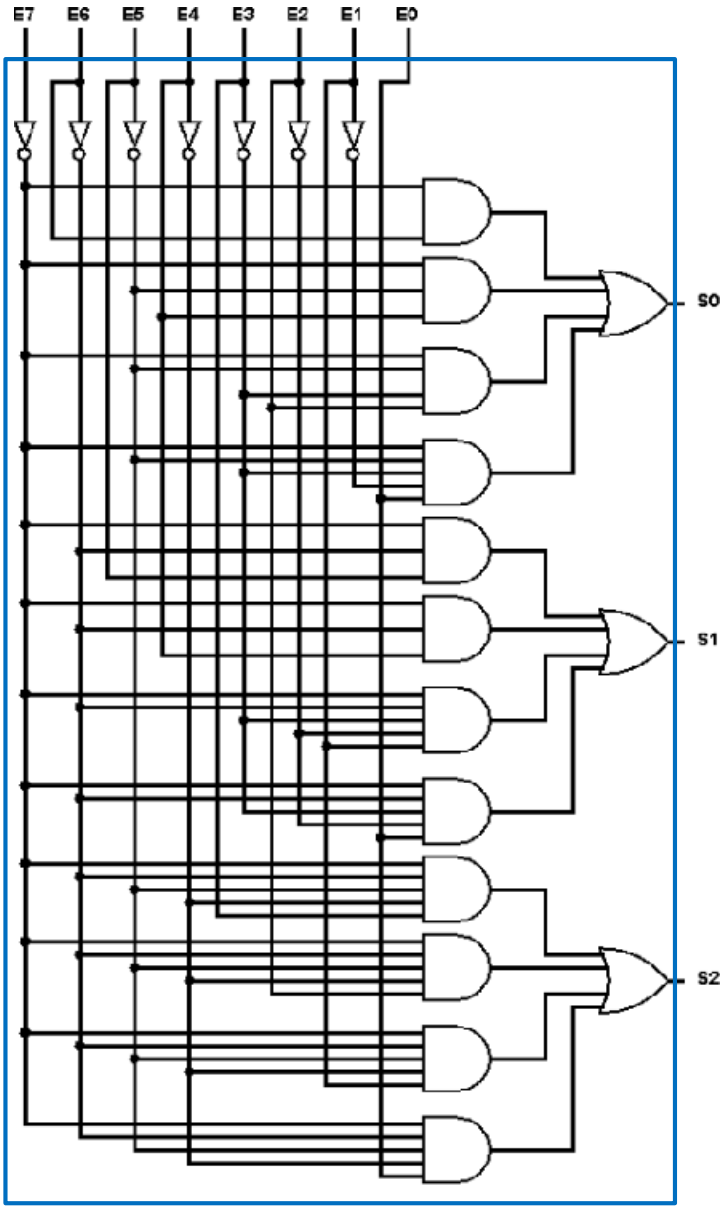
c- Equations de sortie :

$$S_0 = \overline{E_7}E_6 + \overline{E_7}\,\overline{E_6}E_5E_4 + \overline{E_7}E_6\overline{E_5}\,\overline{E_4}E_3E_2 + \overline{E_7}E_6E_5\overline{E_4}\,\overline{E_3}\,\overline{E_2}E_1E_0$$
$$= \overline{E_7}E_6 + \overline{E_7}\,\overline{E_5}E_4 + \overline{E_7}\,\overline{E_5}\,\overline{E_3}E_2 + \overline{E_7}\,\overline{E_5}\,\overline{E_3}\,\overline{E_1}E_0$$

$$S_1 = \overline{E_7}\,\overline{E_6}E_5 + \overline{E_7}\,\overline{E_6}E_5E_4 + \overline{E_7}E_6\overline{E_5}\,\overline{E_4}\,\overline{E_3}E_2E_1 + \overline{E_7}E_6E_5\overline{E_4}\,\overline{E_3}\,\overline{E_2}\,\overline{E_1}E_0$$
$$= \overline{E_7}\,\overline{E_6}E_5 + \overline{E_7}\,\overline{E_6}E_4 + \overline{E_7}\,\overline{E_6}\,\overline{E_3}\,\overline{E_2}E_1 + \overline{E_7}\,\overline{E_6}\,\overline{E_3}\,\overline{E_2}E_0$$

$$S_2 = \overline{E_7}\,\overline{E_6}\,\overline{E_5}\,\overline{E_4}E_3 + \overline{E_7}\,\overline{E_6}\,\overline{E_5}E_4\overline{E_3}E_2 + \overline{E_7}\,\overline{E_6}E_5\overline{E_4}\,\overline{E_3}\,\overline{E_2}E_1 + \overline{E_7}E_6\overline{E_5}\,\overline{E_4}\,\overline{E_3}\,\overline{E_2}\,\overline{E_1}E_0$$
$$= \overline{E_7}\,\overline{E_6}\,\overline{E_5}\,\overline{E_4}E_3 + \overline{E_7}\,\overline{E_6}\,\overline{E_5}\,\overline{E_4}E_2 + \overline{E_7}\,\overline{E_6}\,\overline{E_5}\,\overline{E_4}E_1 + \overline{E_7}\,\overline{E_6}\,\overline{E_5}\,\overline{E_4}E_0$$

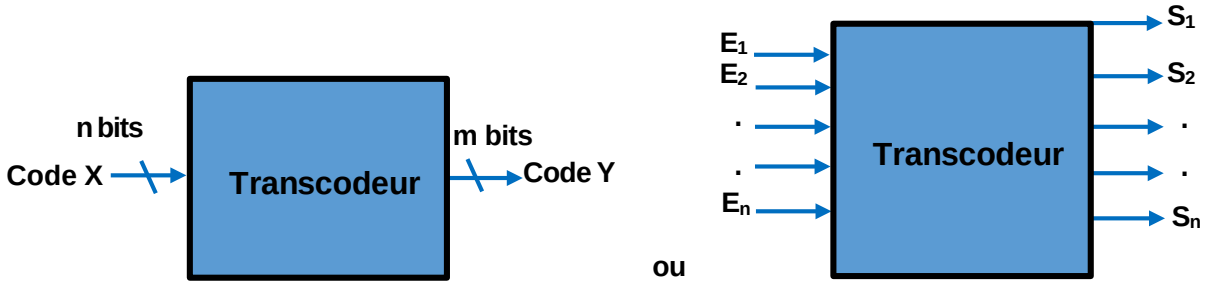
d- Schéma logique :



2.4.9. Transcodeur

• C'est un circuit combinatoire qui permet de transformer un code X (sur n bits) en entrée en un code Y (sur m bits) en sortie.

• Passage d'un code (Code X) à un code (Code Y)



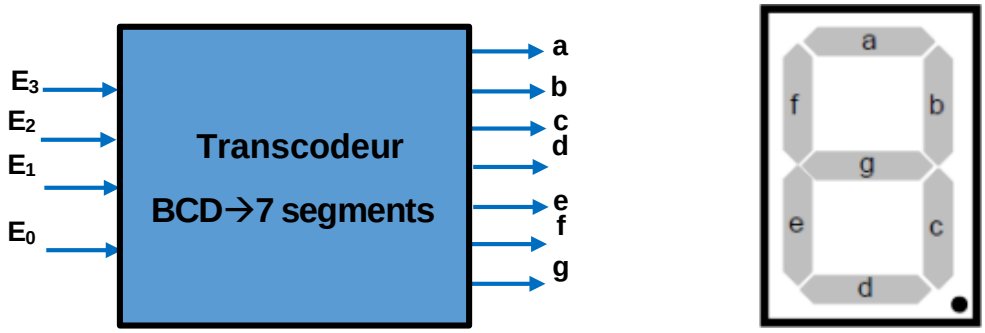
• Exemples de code : Binaire, binaire réfléchi, 7-segments, BCD, …

2.4.9.1. Transcodeur 7-segments

BCD⇨ code affichage chiffre (afficheur 7-segments

Le transcodeur 7 segments accepte en entrée les 4 bits DCB (a0, a1, a2, a3) et rend actives les sorties qui vont permettre de faire passer un courant dans les segments d'un afficheur numérique pour former les chiffres décimaux (de 0 à 9).

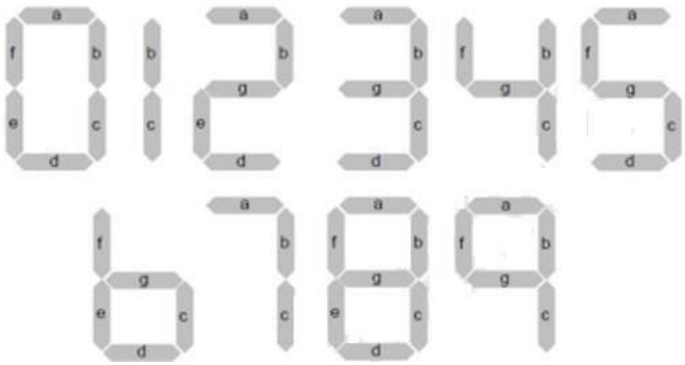
~ a-schéma symbolique



Code binaire 0 à 9 Configuration alimentation des diodes (ou LCD)

~ b-Table de vérité

Il y'a 6 combinaisons intitulés 10, 11, 12, 13, 14, 15. Les autres chiffres sont affichés comme suit :



Affichage	Table de vérité												
	Entrées					Sorties							
	E_3	E_2	E_1	E_0		a	b	c	d	e	f	g
0	0	0	0	0		1	1	1	1	1	1	0
1	0	0	0	1		0	1	1	0	0	0	0
2	0	0	1	0		1	1	0	1	1	0	1
3	0	0	1	1		1	1	1	1	0	0	1
4	0	1	0	0		0	1	1	0	0	1	1
5	0	1	0	1		1	0	1	1	0	1	1
6	0	1	1	0		0	0	1	1	1	1	1
7	0	1	1	1		1	1	1	0	0	0	0
8	1	0	0	0		1	1	1	1	1	1	1
9	1	0	0	1		1	1	1	0	0	1	1
10	1	0	1	0		X	X	X	X	X	X	X
11	1	0	1	1		X	X	X	X	X	X	X
12	1	1	0	0		X	X	X	X	X	X	X
13	1	1	0	1		X	X	X	X	X	X	X
14	1	1	1	0		X	X	X	X	X	X	X
15	1	1	1	1		X	X	X	X	X	X	X

~ c- Equations de sortie

Segment a

E_1E_0 \ E_3E_2	00	01	11	10
00	1	0	X	1
01	0	1	X	1
11	1	1	X	X
10	1	0	X	X

$$a\,(E_3,E_2,E_1,E_0) = E_3 + E_2E_0 + E_2\overline{E_1} + \overline{E_2}\,\overline{E_0}$$

Segment b

E_1E_0 \ E_3E_2	00	01	11	
00	1	1	X	1
01	1	0	X	
11	1	1	X	X
10	1	0	X	

$$b\,(E_3,E_2,E_1,E_0) = \overline{E_2} + \overline{E_1}\,\overline{E_0} + E_1E_0$$

Segment c

E_1E_0 \ E_3E_2	00	01	11	10
00	1	1	X	1
01	1	1	X	1
11	1	1	X	X
10	0	1	X	X

$$c(E_3,E_2,E_1,E_0) = E_2 + \overline{E_1} + E_0$$

Segment d

E_1E_0 \ E_3E_2	00	01	11	10
00	1	0	X	
01	0	1	X	0
11	1	0	X	X
10	1	1	X	

$$d\,(E_3,E_2,E_1,E_0) = E_1\overline{E_0} + \overline{E_2}E_1 + \overline{E_2}\,\overline{E_0} + E_2\overline{E_1}E_0$$

Segment e

E_3E_2 / E_1E_0	00	01	11	10
00	1	0	X	1
01	0	0	X	1
11	0	0	X	X
10	1	1	X	X

$$e(E_3,E_2,E_1,E_0) = E_1\overline{E_0} + E_3E_0 + \overline{E_2}\,\overline{E_0}$$

Segment f

E_3E_2 / E_1E_0	00	01	11	10
00	1	1	X	
01	0	1	X	
11	0	0	X	
10	0	1	X	

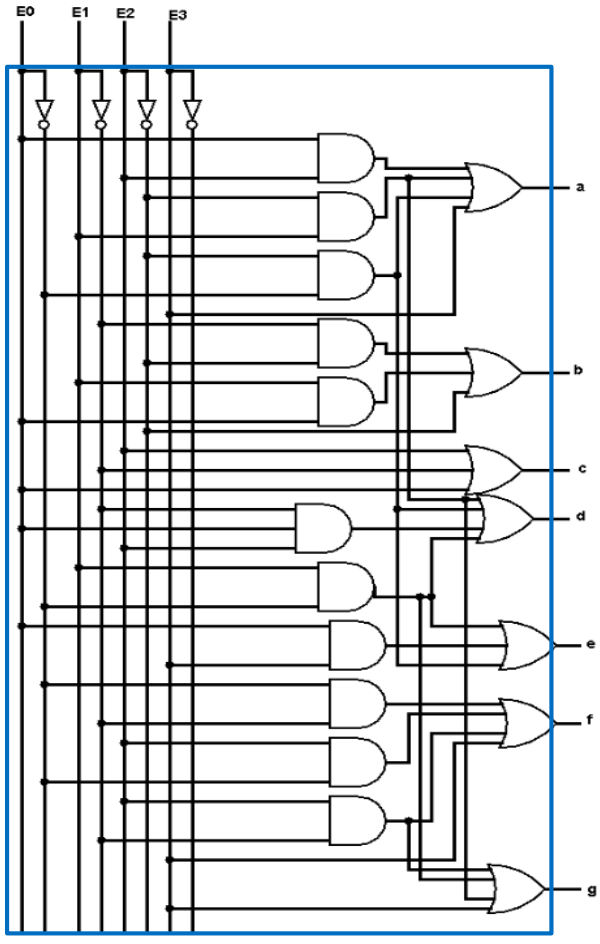
$$f(E_3,E_2,E_1,E_0) = E_3 + \overline{E_1}\,\overline{E_0} + E_2\overline{E_1} + E_2\overline{E_0}$$

Segment g

E_3E_2 / E_1E_0	00	01	11	10
00	0	1	X	1
01	0	1	X	1
11	1	0	X	X
10	1	1	X	X

$$g(E_3,E_2,E_1,E_0) = E_3 + E_2\overline{E_1} + E_1\overline{E_0} + \overline{E_2}E_1$$

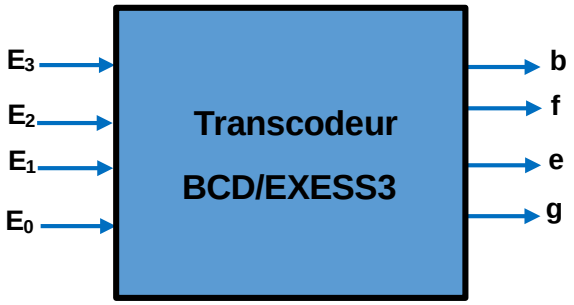
~ **d- Schéma logique**



Transcodeur BCD→7 segments

2.4.9.2. Transcodeur BCD/EXESS3

~ **a-schéma symbolique**



~ **b-Table de vérité :**

Entrées				Sorties			
E_3	E_2	E_1	E_0	S_3	S_2	S_1	S_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

~ **c- Equations de sortie :**

E_1E_0 \ E_3E_2	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	0	1	X	X
10	0	1	X	X

$$S_3(E_3,E_2,E_1,E_0) = E_3 + E_2E_1 + E_2E_0$$

E_1E_0 \ E_3E_2	00	01	11	
00	0	1	X	
01	1	0	X	
11	1	0	X	
10	1	0	X	X

$$S_2(E_3,E_2,E_1,E_0) = E_2\overline{E_1}\,\overline{E_0} + \overline{E_2}E_0 + \overline{E_2}E_1$$

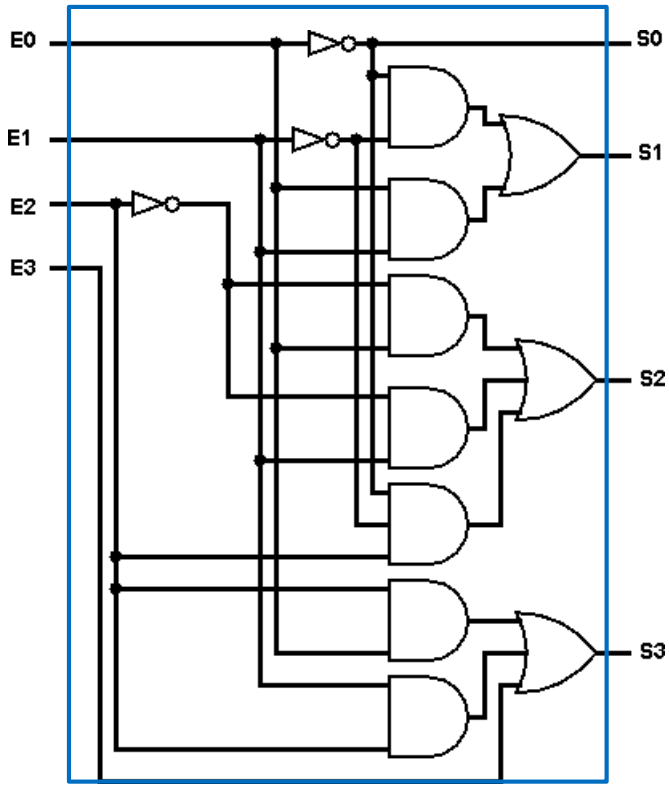
E_1E_0 \ E_3E_2	00	01	11	10
00	1	1	X	1
01	0	0	X	0
11	1	1	X	X
10	0	0	X	X

$$S_1(E_3,E_2,E_1,E_0) = \overline{E_1}\overline{E_0} + E_1E_0$$

E_1E_0 \ E_3E_2	00	01	11	10
00	1	1	X	
01	0	0	X	
11	0	0	X	
10	1	1	X	

$$S_0(E_3,E_2,E_1,E_0) = \overline{E_0}$$

~ **d- Schéma logique**



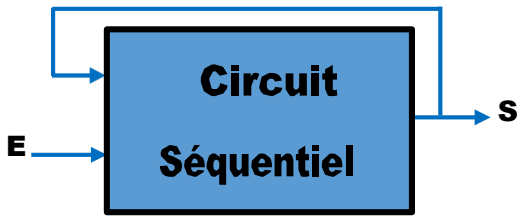
Transcodeur BCD/EXESS3

Chapitre III : Les Circuits Séquentiels

3.1. Introduction

Dans la logique combinatoire nous les signaux de sortie ne dépendaient que des états des variables d'entrée. Pour les circuits de logique séquentielle nous devons tenir compte de l'état du système. Ainsi les sorties dépendent des entrées mais également de l'état du système. Celui-ci dépend aussi des entrées. Les systèmes séquentiels sont des systèmes dont le fonctionnement dépend d'une part de la valeur des entrées et d'autre part par l'état du système La logique séquentielle a pour élément de base « la bascule » contrairement à la logique combinatoire qui avait pour élément de base la porte logique. Les circuits séquentiels présentent une caractéristique de mémoire. La différence entre la logique combinatoire et la logique séquentielle est que :

- Logique combinatoire : les états de sortie dépendent uniquement de la combinaison des variables d'entrées.
- Logique séquentielle : l'état de la sortie dépend à la fois de la combinaison des variables d'entrée et de l'état antérieur de la sortie (temporelle).



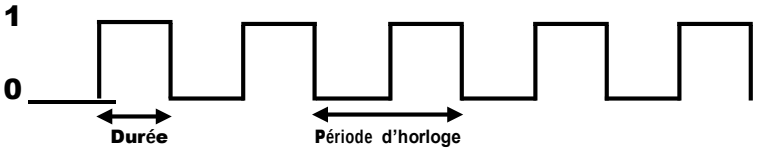
$$S_{t+1} = f(E, S_t) \text{ ou } S^+ = f(E, S)$$

3.2. Système séquentiels synchrone et asynchrones

On classe les systèmes séquentiels en deux grandes catégories :

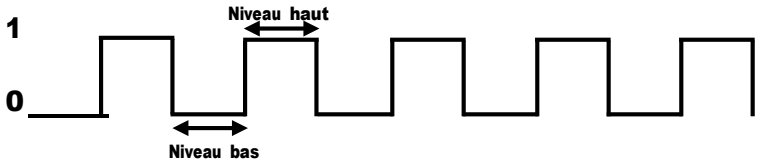
a- Systèmes séquentiels asynchrones : dit ainsi s'il peut évoluer seul sans ordre extérieur. Les sorties sont définies par les entrées et par le fonctionnement de circuit.

b- Systèmes séquentiels synchrones : dit ainsi s'il ne peut évoluer que sur un ordre extérieur à lui-même (en absence d'ordre le système reste figé dans l'état ou il se trouve) on appelle cette entrée de commande horloge. Sa fréquence est l'inverse de sa période (ou temps de cycle) [5]. Le signal horloge : est un signal de synchronisation périodique logique qui passe de l'état 1 à l'état 0 et de 0 à 1 d'une façon périodique dans le temps.



Forme des signaux de commandes :

a. Signal à niveau :



➲ Synchronisation sur niveau haut

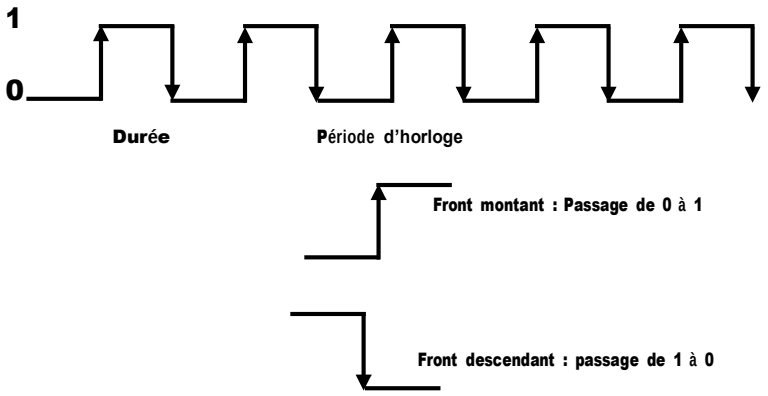
- Si H=0 : la sortie maintient son état, quelles que soient les valeurs des entrées.
- Si H=1 : la bascule fonctionne en mode normale, les sorties obéissent aux entrées.

Donc la bascule ne fonctionne normalement que si H=1(Niveau haut).

➲ Synchronisation sur niveau bas

- Si H=1 : la sortie maintient son état, quelles que soient les valeurs des entrées.
- Si H=0 : la bascule fonctionne en mode normale.

b. Signal impulsionnel



➲ Synchronisation sur front

Les variables logiques ont deux niveaux : le niveau logique bas « 0 » et le niveau logique haut « 1 ».

- Le passage du niveau bas vers le niveau haut s'appel front montant.
- Le passage du niveau haut vers le niveau bas s'appel front descendent.

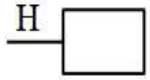
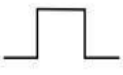
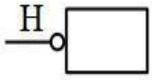
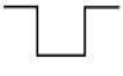
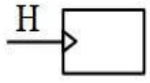
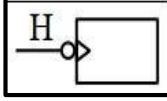
Symbole	Fonctionnement	Forme du signal
	Horloge fonctionnant sur niveau haut (H)	
	Horloge fonctionnant sur niveau bas (B)	
	Horloge fonctionnant sur passage du niveau bas au niveau haut	
	Horloge fonctionnant sur passage du niveau haut au niveau bas	

Table 2.1. Types d'horloges et leurs symboles.

Les entrées ne sont validées que au moment où les impulsions d'horloge sont produites, certain sont sensible à des fronts montants ou descendants.

3.3. Les bascule

3.3.1. Définition d'une bascule

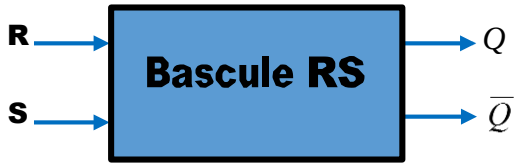
Une bascule (flip-flop) a pour rôle de mémoriser une information élémentaire. C'est une mémoire à 1 bit. La mémorisation fait appel à un verrou (latch) ou système de blocage. Les bistables (Flip Flop) et les Bascules (MASTER SLAVE Flip Flop ou LATCH) représentent la base de la logique séquentielle dont la fonction essentielle est la fonction de mémorisation. La bascule est un circuit qui comporte une ou plusieurs entrées et deux sorties complémentaires Q et $\overline{Q}$.

3.3.2. Les types des bascules

Les bistables les plus utilisées sont : Bistable RS, Bistable JK ; bistable D et bistable T.

3.3.2.1. Bascule RS :(Reset_set)

Est constituée par deux entrées (S mise à 1 (Set)) et (R mise à 0 (Rset)) et de deux sorties Q et $\overline{Q}$ [6].



Le nouvel état de la bascule Q+ dépend de son état antérieur Q et de l'état des entrées R et S. $Q^+=f(Q,R,S)$

Sorties inchangées : ➲**Set** : remise à 1 ➲**Reset** : remise à 0 à proscrire

Fonctionnement statique :

R : Reset : entrée de mise à 0 de la bascule (si R=1 ; la sortie Q=0).

S : set : entrée de mise à « 1 » de la bascule (si S=1 ; la sortie Q=1).

La condition R=S=1 est interdite. Elle donne un état indéterminé de la sortie.

Table de vérité :

R	S	Q	Q^+
0	0	0	0 mémorisation de l'information
0	0	1	1 mémorisation de l'information
0	1	0	1 mise à 1 de la sortie
0	1	1	1 mise à 1 de la sortie
1	0	0	0 mise à 0 de la sortie
1	0	1	0 mise à 0 de la sortie
1	1	0	X état indéterminé
1	1	1	X état indéterminé

La table de vérité condensée

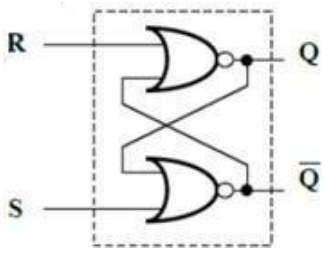
R	S	Q^+
0	0	Q conservation de l'état interne
0	1	1 mise à1 $(\forall Q)$
1	0	0 mise à 0 $(\forall Q)$
1	1	X état indéterminé

Les équations d'état

$$
\begin{aligned}
Q^+ &= \overline{R}\overline{S}Q + \overline{R}SQ + \overline{R}S\overline{Q} \\
&= \overline{R}\overline{S}Q + \overline{R}S\left(Q + \overline{Q}\right) \\
&= \overline{R}\overline{S}Q + \overline{R}S \\
&= \overline{R}\left(\overline{S}Q + S\right) \\
&= \overline{R}\left(\left(\overline{S} + S\right)\left(Q + S\right)\right) \\
&= \overline{R}\left(Q + S\right)
\end{aligned}
$$

$$Q^+ = \overline{\overline{Q}}^+ = \overline{\overline{R}\left(Q + S\right)} = \overline{R + \left(Q + S\right)}$$

Représentation à l'aide de portes NOR



Représentation à l'aide de portes NAND

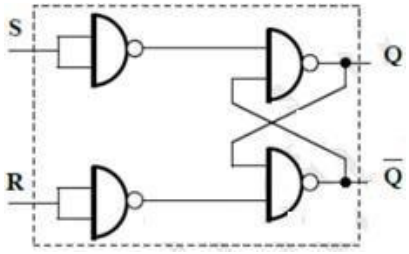
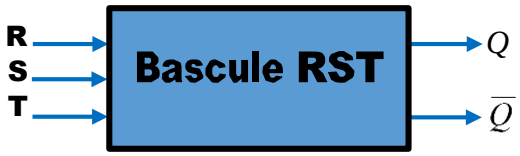


Table de transition d'une bascule RS

Q	Q^+	.R	S
0	0	X	0
0	1	0	1
1	0	1	0
1	1	0	X

3.3.2.2. Bascules RST



Fonctionnement :

Les entrées R et S sont prises en compte que si elles sont en coïncider avec un signal de commande T (horloge) :

- Si T=0 la bascule conserve son état.
- Si $T=1$ $\begin{cases} R=S=0 & \text{conservation d'état} \\ R=1, S=0 & \text{mise à 0} \\ R=0, S=1 & \text{mise à 1} \\ R=S=1 & \text{indéterminé} \end{cases}$

Table de vérité :

T	R	S	Q	Q^+
0	0	0	0	0 conservation d'état
0	0	0	1	1 conservation d'état
0	0	1	0	0 conservation d'état
0	0	1	1	1 conservation d'état
0	1	0	0	0 conservation d'état
0	1	0	1	1 conservation d'état
0	1	1	0	0 conservation d'état
0	1	1	1	1 conservation d'état
1	0	0	0	0 conservation d'état
1	0	0	1	1 conservation d'état
1	0	1	0	1 mise à 1 de la sortie
1	0	1	1	1 mise à 1 de la sortie
1	1	0	0	0 mise à 0 de la sortie
1	1	0	1	0 mise à 0 de la sortie
1	1	1	0	X état indéterminé
1	1	1	1	X état indéterminé

La table de vérité condensée

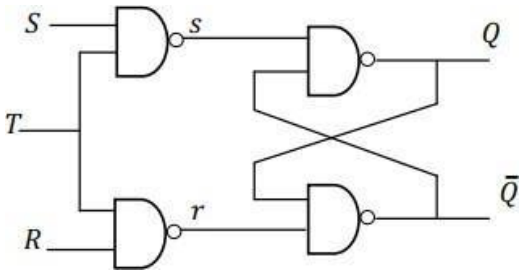
T	R	S	Q^+
0	X	X	Q conservation de l'état (mémorisation de l'information)
1	0	0	Q conservation de l'état (mémorisation de l'information)
1	0	1	1 mise à 1 $(\forall Q)$
1	1	0	0 mise à 0 $(\forall Q)$
1	1	1	X état indéterminé

Les équations d'état

$$Q^+ = S\overline{R}T + \overline{T}Q + \overline{S}\overline{R}TQ$$

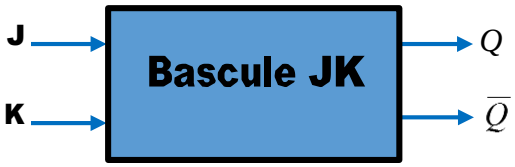
$$\begin{cases} s = \overline{TS} = \overline{T} + \overline{S} \\ r = \overline{TR} = \overline{T} + \overline{R} \end{cases}$$

Représentation par les portes NAND



3.3.2.3. Bascule JK

Les bascules JK, identiques aux bascules RS pour les entrées autres que 11 (l'entrée J correspond à S et l'entrée K à R). Pour les entrées 11 ces bascules fonctionnent en mode Toggle à savoir que leur sortie Q change de valeur à chaque impulsion d'horloge. Par rapport aux bascules RS, les bascules JK permettent d'utiliser toutes les combinaisons des entrées.



Fonctionnement statique :

Contrairement à la bascule RS, la condition J=K=1, ne donne pas lieu à une condition indéterminée, mais par contre la bascule passe à l'état opposé.

Table de vérité :

J	K	Q	Q^+
0	0	0	0 conservation d'état (mémorisation de l'information)
0	0	1	1 conservation d'état (mémorisation de l'information)
0	1	0	0 mise à 0 de la sortie
0	1	1	0 mise à 0 de la sortie
1	0	0	1 mise à 1 de la sortie
1	0	1	1 mise à 1 de la sortie
1	1	0	1 inverse d'état (basculement)
1	1	1	0 inverse d'état (basculement)

La table de vérité condensée

J	K	Q^+
0	0	Q conservation de l'état interne
0	1	0 mise à 0 $(\forall Q)$
1	0	1 mise à 1 $(\forall Q)$
1	1	$\overline{Q}$ inverse d'état (basculement)

Les équations d'état

$$\begin{aligned} Q^{+} &= \overline{J}\,\overline{K}Q + J\overline{K}\overline{Q} + J\overline{K}Q + JK\overline{Q} \\ &= \overline{K}Q\left(\overline{J} + J\right) + J\overline{Q}\left(K + \overline{K}\right) \\ &= \overline{K}Q + J\overline{Q} \end{aligned}$$

Table de transition d'une bascule JK

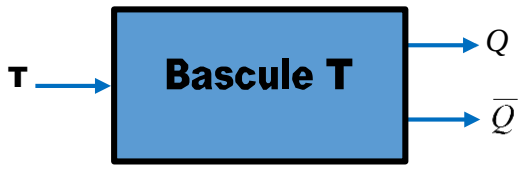
On connait les valeurs des sorties, comment déterminer les valeurs des entrées JK ? La table des transitions d'une bascule JK se remplit à l'aide de sa table de vérité.

- ligne $1 \Rightarrow$ Q passe de 0 à 0 $\Rightarrow$ { état mémoire $(J=0, K=0)$; mise à 0 $(J=0, K=1)$ } $\Rightarrow (J=0, K=X)$
- ligne $2 \Rightarrow$ Q passe de 0 à 1 $\Rightarrow$ { état inverseur $(J=1, K=1)$; mise à 1 $(J=1, K=0)$ } $\Rightarrow (J=1, K=X)$
- ligne $3 \Rightarrow$ Q passe de 1 à 0 $\Rightarrow$ { état inverseur $(J=1, K=1)$; mise à 0 $(J=0, K=1)$ } $\Rightarrow (J=X, K=1)$
- ligne $4 \Rightarrow$ Q passe de 1 à 1 $\Rightarrow$ { état mémoire $(J=0, K=0)$; mise à 1 $(J=1, K=0)$ } $\Rightarrow (J=X, K=0)$

Q	Q^{+}	J	K	
0	0	0	X	mise à 0 ou état mémoire
0	1	1	X	mise à 1 ou basculement
1	0	X	1	mise à 0 ou basculement
1	1	X	0	mise à 1 ou état mémoire

3.3.2.4. Bascule T (Trigger flip-flop)

Bascule à déclenchement



Fonctionnement :

- si T=1, inversion des états ;
- si T=0, Q ne change pas.

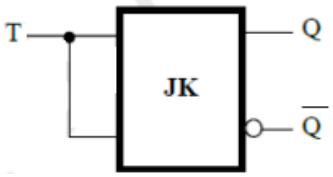
Table de vérité :

T	Q	Q^{+}	
0	0	0	conservation de l'état (état mémoire)
0	1	1	conservation de l'état (état mémoire)
1	0	1	inverse d'état (basculement)
1	1	0	inverse d'état (basculement)

Les équations d'état

$$Q^+ = \overline{T}Q + T\overline{Q}$$
$$= T \oplus Q$$

Réalisation



Remarque : En remplaçant J et K par T dans l'équation de la bascule JK on aura

$$Q^+ = TQ + \overline{T}\,\overline{Q}$$
$$= T \oplus \overline{Q}$$

3.3.2.5. Bascule D (Delay)

Recopie, sur sa sortie Q, le signal d'entrée D

Fonctionnement :

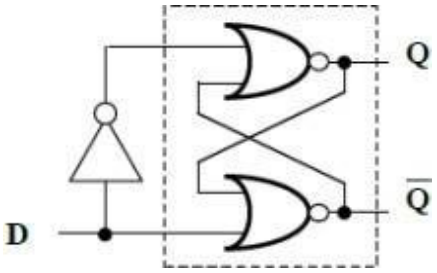
Un appui sur D →Mise à 1 de Q (si D=1, Q^+ = D).

Un relâchement de D →Mise à 0 de Q.

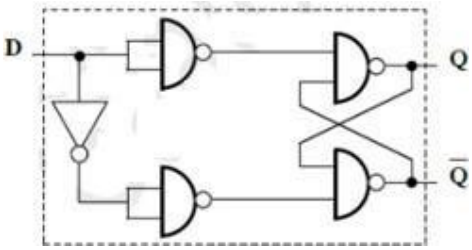
Table de vérité :

D	Q	Q^+	
0	0	0	mise à 0
0	1	0	mise à 0
1	0	1	mise à 1
1	1	1	mise à 1

Réalisation à l'aide de portes NOR



Réalisation à l'aide de portes NAND



Remarque : En mettant S=D et R= $\overline{D}$ dans l'équation de la bascule RS on aura

$$Q^+ = D\overline{D}Q + \overline{\overline{D}}D$$
$$= D$$

Ainsi on obtient une bascule D en rajoutant un inverseur entre S et R.

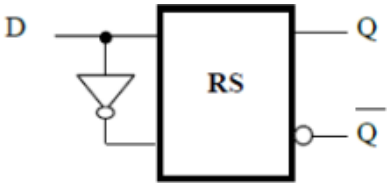


Table de transition d'une bascule D

Q	Q^+		D	
0	0		0	mise à 0
0	1		1	mise à 1
1	0		0	mise à 0
1	1		1	mise à 1

3.4. Utilisation des bascules

3.4.1. Utilisation des bascules pour réaliser un registre

3.4.1.1. Définition d'un registre

Un registre est d'abord un ensemble de cases ou cellules mémoires capables de stocker une information (un mot ou un nombre binaire). Un registre est constitué d'une collection de bascules de même type, actionnées par la même impulsion d'horloge. Le nombre de bascules constituant un registre détermine sa capacité [7]. Dans le système binaire, une case mémoire est définie à l'aide d'une bascule. Un registre est donc un ensemble ordonné de bascules.

- Une case mémoire est définie à l'aide d'une bascule
- 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 (sauvegarder) une information sur n bits.

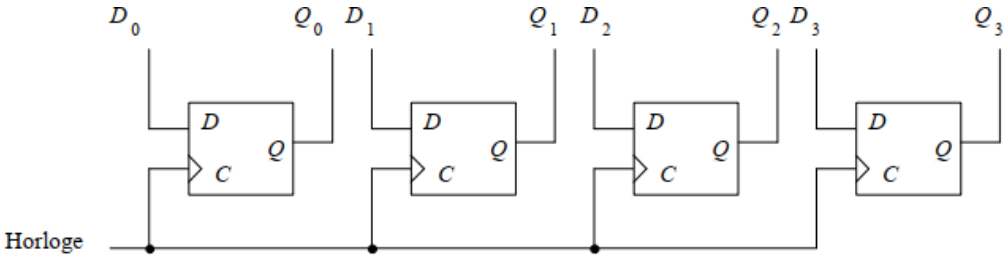


Figure 3. 1. Exemple d'un registre à 4 bits à base de bascules D.

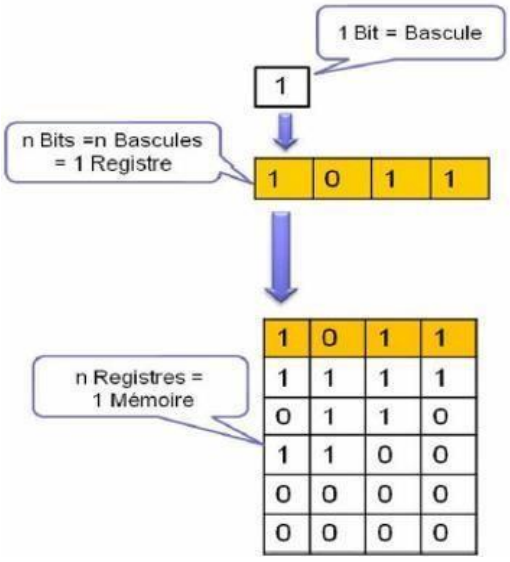


Figure 3. 2. Illustration de la relation bit, registre et mémoire.

3.4.1.2. Fonctionnement d'un registre

Un registre sert à mémoriser un mot ou un nombre binaire. Le schéma d'un tel système comporte autant de bascules type D que l'élément binaires à mémoriser. Toutes les bascules sont commandées par le même signal d'horloge.

Les principales fonctions d'un registre sont :

~ La mémorisation

Mémoriser l'information telle qu'elle reste sans aucun changement

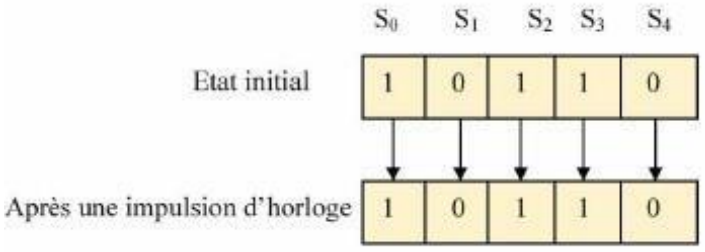


Figure 3. 3. Illustration de la mémorisation.

~ Le décalage à droite ou à gauche

Décaler l'information soit de la gauche vers la droite ou de la droite vers la gauche

- Décalage à droite : pour chaque impulsion d'horloge le contenu de la bascule de rang i est transmis à celle de rang i+1.

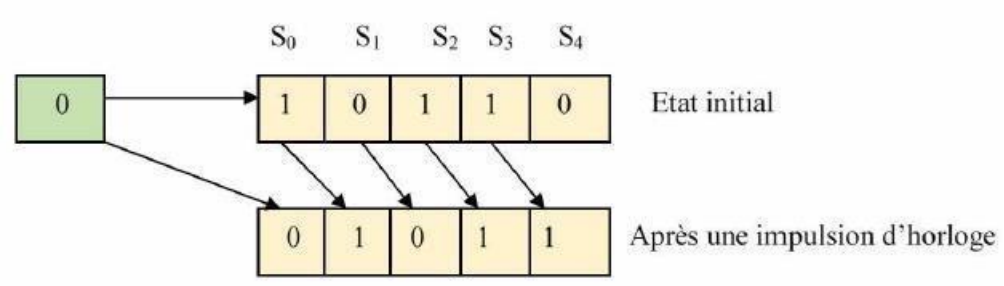


Figure 3. 4. Illustration du décalage à droite.

- Décalage à gauche : pour chaque impulsion d'horloge la bascule de rang i prend le contenu de la bascule de rang i-1.

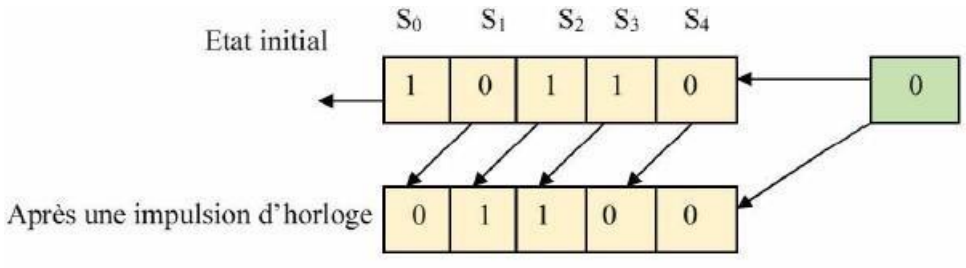
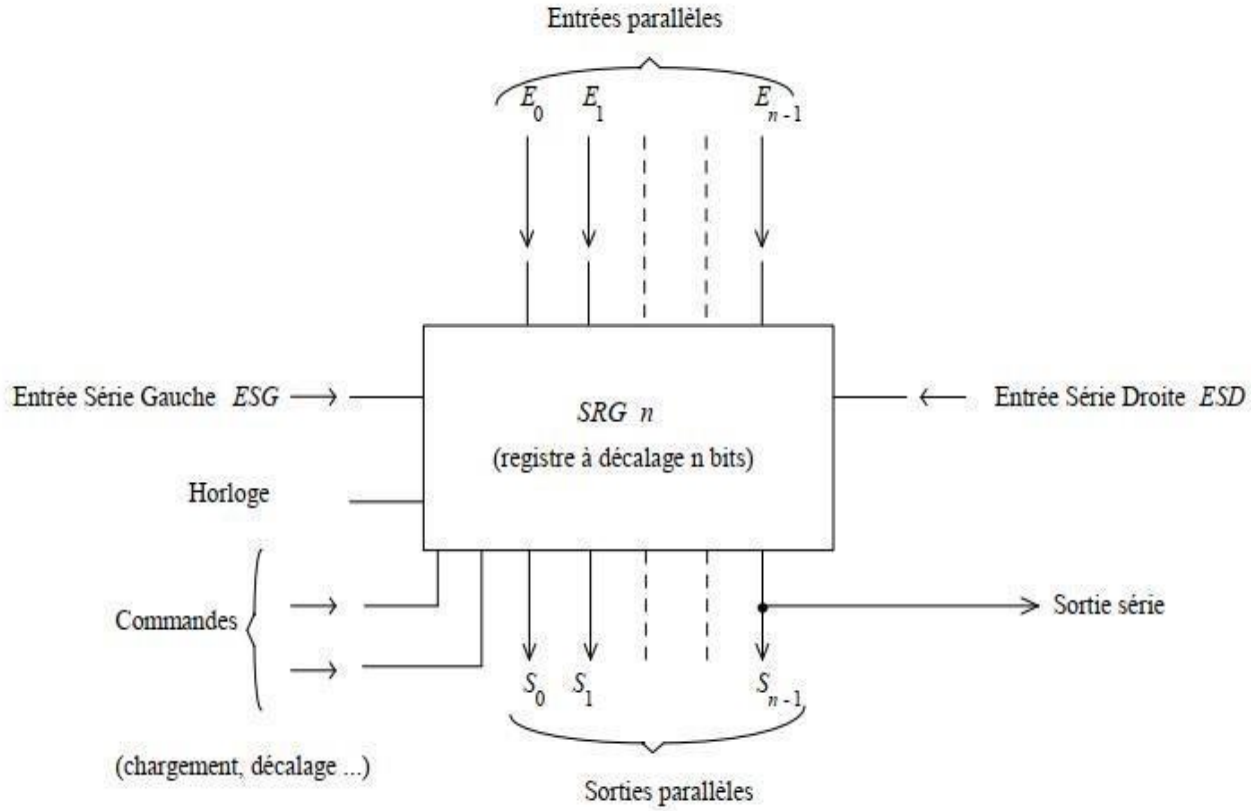


Figure 3. 5. Illustration du décalage à gauche.

3.4.1.3. Type des registres

Plus généralement un registre peut se représenter par le schéma suivant [8]:



Il existe plusieurs types de registres :

- Registre de mémorisation
- Registre à entrées parallèles et sorties parallèles (Registre à chargement parallèle).
- Registre à entrée série et sortie série
- Registre à entrée série et sortie parallèle.
- Registre à entrée parallèle et sortie série.
- Registre à décalage à droite.
- Registre à décalage à gauche.
- Registre à décalage circulaire.

3.4.1.3.1. Registre de mémorisation

Registre de mémorisation peut se réaliser sous la forme représentée par l'exemple au-dessous et qui consiste à interdire l'action de l'horloge en intercalant une porte ET (and) en série.

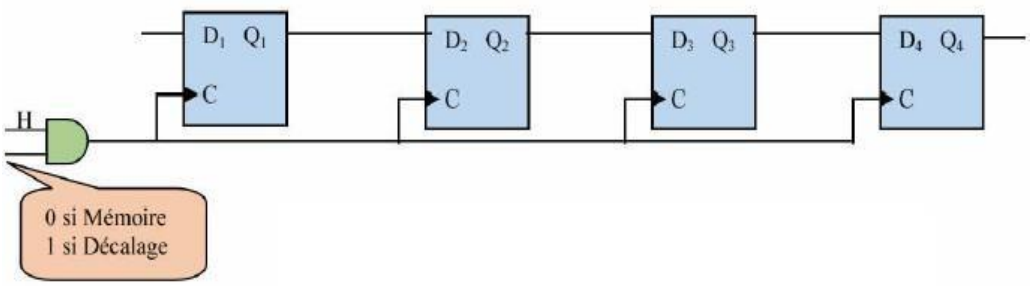


Figure 3. 6. Exemple de registre de mémorisation à base de bascules D.

3.4.1.3.2. Registre à entrées parallèles et sorties parallèles (Registre à chargement parallèle)

- Il peut charger une information sur N bits en même temps.
- Les n bascules chargement d'états en même temps.
- Chaque bascule de rang i prend la valeur de l'information i.
- Il possède une entrée de chargement chg (chg=0 état mémoire, chg=1 chargement)

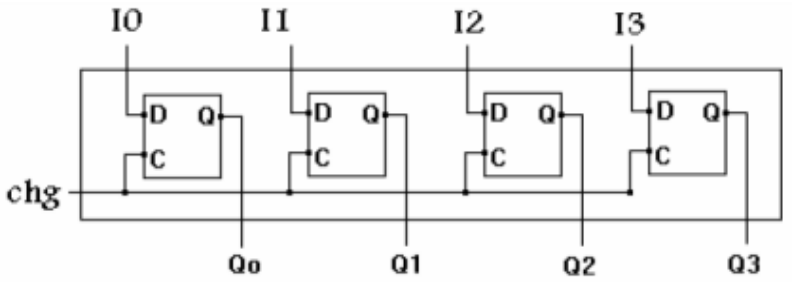


Figure 3. 7. Exemple de registre à chargement parallèle à base de bascules D.

3.4.1.3.3. Registre à entrée série et sortie série

- L'information est introduite bit par bit (en série).
- L'ensemble du registre est décalé d'une bascule de rang i vers la bascule de rang i+1 et la première bascule reçoit une nouvelle entrée E_S.
- Un tel registre est appelé soit :

~ Registre à entrée série à gauche et à sortie série à droite.

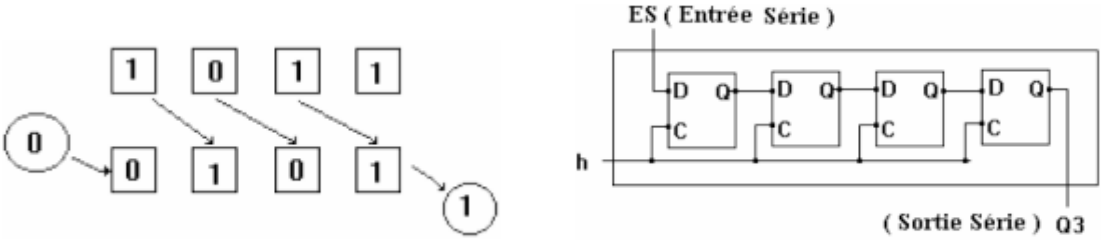


Figure 3. 8. Exemple de registre à entrée série à gauche et à sortie série à droite à base de bascules D.

~ Registre à entrée série à droite et à sortie série à gauche

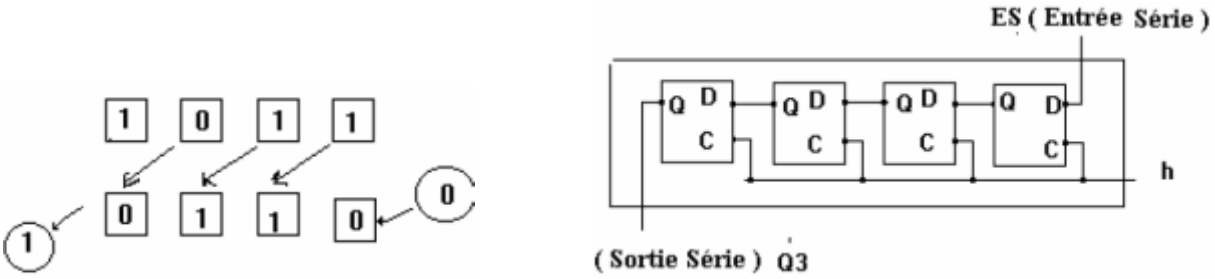


Figure 3. 9. Exemple de registre à entrée série à droite et à sortie série à gauche à base de bascules D.

3.4.1.3.4. Registre à entrée série et sortie parallèle

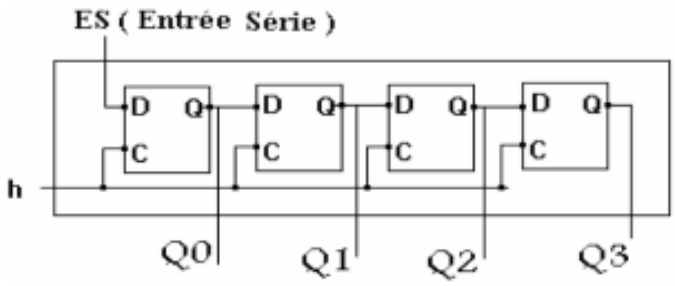


Figure 3. 10. Exemple de registre à entrée série et à sortie parallèle à base de bascules D.

3.4.1.3.5. Registre à entrée parallèle et sortie série

Un registre à décalage à entrée parallèle et sortie série transforme un codage spatial en codage temporel. Dans cet exemple Si X=1 l'entrée parallèle est inhibée et l'entrée série est validée. Si X = 0 l'entrée série est bloquée par contre le chargement par l'entrée parallèle est autorisé.

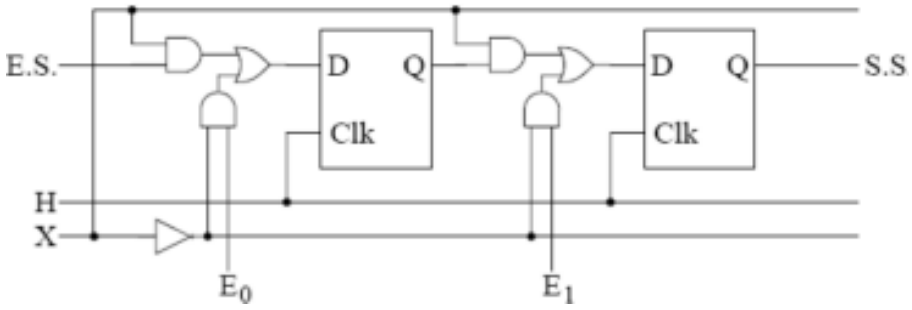


Figure 3. 11. Exemple de registre à entrée parallèle et à sortie série à base de bascules D.

3.4.1.3.6. Registre à décalage à droite

C'est un registre qui effectue un décalage vers la droite tel que l'entrée de la première bascule reçoit une nouvelle entrée E_S.

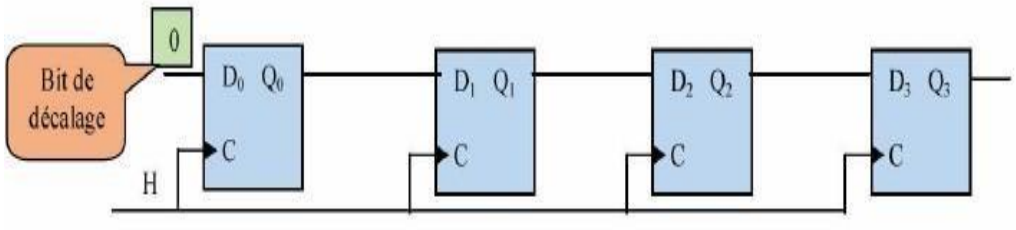


Figure 3. 12. Exemple de registre à décalage à droite à base de bascules D.

3.4.1.3.7. Registre à décalage à gauche

C'est un registre qui effectue un décalage vers la gauche tel que l'entrée de la première bascule reçoit une nouvelle entrée E_S.

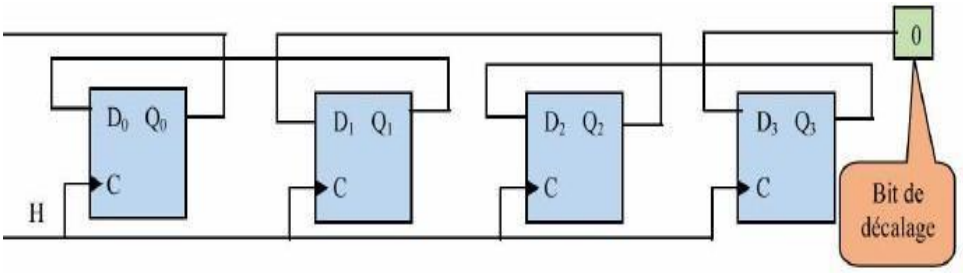


Figure 3. 13. Exemple de registre à décalage à gauche à base de bascules D.

3.4.1.3.6. Registre à décalage circulaire

• C'est un registre qui effectue un décalage vers la gauche en répercutant la sortie de la dernière bascule vers l'entrée de la dernière bascule.

• Le décalage peut être un décalage droite (circulaire droite) ou gauche (circulaire gauche)

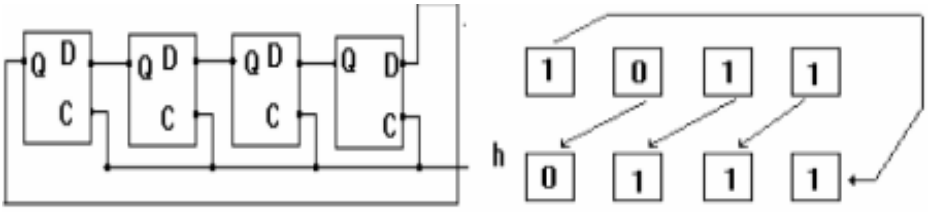


Figure 3. 14. Exemple de registre à décalage circulaire à base de bascules D.

3.4.2. Utilisation des bascules pour la mémoire centrale

3.4.2.1. Définition d'une mémoire

Une mémoire est un dispositif capable d'enregistrer, de conserver et de restituer des informations. Les éléments de mémoire d'un ordinateur se répartissent en plusieurs niveaux caractérisés par leur capacité et leur temps d'accès.

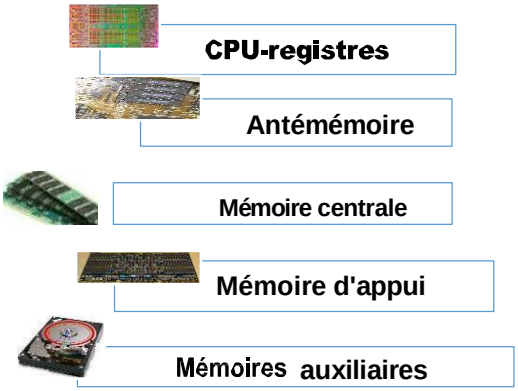


Figure 3. 15. Principaux niveaux de mémoire d'un ordinateur.

3.4.2.2. Différents types de la mémoire

Notons que plus on s'éloigne de la CPU vers les mémoires auxiliaires plus le temps d'accès et la capacité des mémoires augmentent, mais que le coût par bit diminue.

1. les registres : sont les éléments de mémoire situés dans l'unité centrale de traitement (CPU), élément de stockage des opérandes et des résultats intermédiaires.

2. L'antémémoire ou **mémoire cache** Introduit en milieu des années 60 par M. V. Wilkes sous le nom de mémoire esclave (Slave Memory), l'antémémoire ou mémoire cache, ou plus simplement cache, est une mémoire qui sert de tampon entre le processeur et la mémoire [9]. Est une mémoire rapide de faible capacité. Cette mémoire permet au CPU de faire moins d'accès à la mémoire centrale et ainsi de gagner de temps.

3. La mémoire centrale est l'organe principal de rangement des informations utilisées par la CPU. La mémoire centrale est un organe qui permet d'enregistrer, de stocker et de restituer les informations [10]. On distingue deux types :

3.1. Mémoires vives

- Les mémoires vives ou RAM (signifie en anglais : Random Access Memory) sont des mémoires à lecture et écriture qui permettent d'enregistrer des informations, de les conserver et de les restituer.

Figure 3. 16. RAM.

- La mémoire centrale est composée d'un ensemble ordonné de 2^m cellules (point mémoire), chaque cellule contenant un mot de n bits, c'est-à-dire que les n bits sont traités (écrits ou lus) simultanément. La cellule mémoire est la plus petite subdivision (entité atomique) de la mémoire dans laquelle il est possible de lire ou d'écrire une information [11].
- Une mémoire peut être représentée comme une armoire de rangement constituée de différents tiroirs. Chaque tiroir représente alors une case mémoire qui peut contenir un seul élément : des données. Le nombre de cases mémoires pouvant être très élevé, il est alors nécessaire de pouvoir les identifier par un numéro. Ce numéro est appelé adresse. Chaque donnée devient alors accessible grâce à son adresse [12].

Pour pouvoir identifier individuellement chaque mot on utilise m lignes d'adresse (signal adr). La taille d'un bloc mémoire est donc 2^m, le premier mot se situant à l'adresse 0 et le dernier à l'adresse 2^m-1.

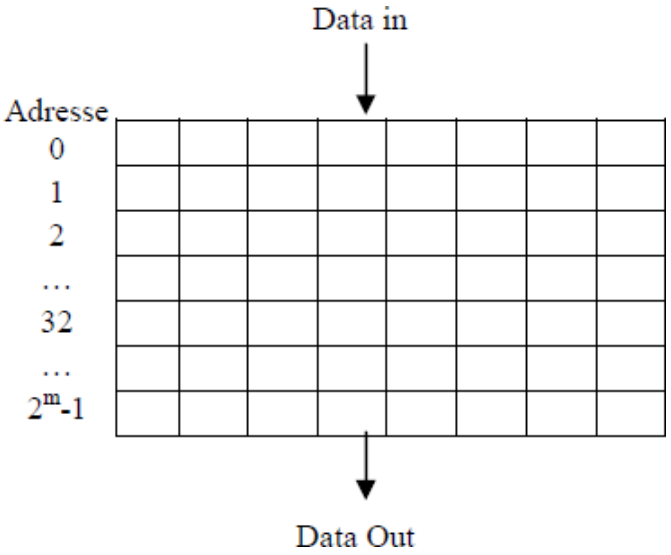


Figure 3. 17. Exemple d'une cellule d'un octet.

Parmi les caractéristiques d'une mémoire nous trouvons la capacité et le format. La capacité représente le nombre total de bits et le format correspond à la longueur des mots. Si m est le nombre de bits d'adresse et n est le nombre de bits par mot, la capacité de la mémoire est donnée par :

Capacité = 2^m mots = 2^m n bits

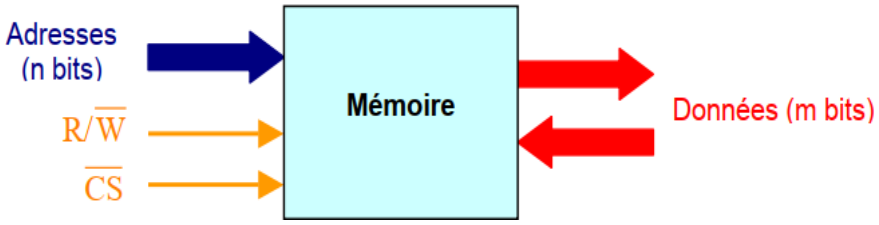


Figure 3. 18. Présentation d'un circuit mémoire.

Sur un circuit mémoire (Figure 3.18) on peut distinguer :

- Les entrées d'adresses
- Les entrées de données
- Les sorties de données
- Les entrées de commandes :
  - Une entrée de selection de lecture ou d'écriture $\left(R/\overline{W}\right)$.
  - Une entrée de selection du circuit $\overline{CS}$.

Une opération de lecture ou d'écriture de la mémoire suit toujours le mêmes cycle :

1. Sélection de l'adresse
2. Choi de l'opération à effectuer $\left(R/\overline{W}\right)$.
3. Sélection de la mémoire $\left(\overline{CS}=0\right)$
4. Lecture ou écriture de la donnée.

Il existe deux grandes familles de mémoires RAM : les RAM statiques (SRAM) ou les RAM dynamiques (DRAM).

- Dans le cas des RAM statique, le point mémoire élémentaire est une bascule.
- Dans le cas des mémoires dynamiques (DRAM), l'élément de mémorisation est un condensateur (capacité) commandée par un transistor. Ce type de mémoire est très utilisé car peu couteux [13].

3.2. Mémoires mortes

La mémoire morte permet de stocker des données nécessaires au démarrage de l'ordinateur:

- Appelée ROM (Read Only Memory, c'est donc une mémoire en lecture seule),
- Ne s'efface pas lors de la mise hors tension du système,
- Contient les éléments essentiels au démarrage de l'ordinateur.

Les mémoires mortes sont utilisées, entre autres, pour stocker :

- Les informations nécessaires au démarrage d'un ordinateur (BIOS, instructions de démarrage, microcode)
- Des tables de constantes ou des tables de facteurs de conversion

Figure 3. 19. ROM.

4. La mémoire d'appui est une mémoire intermédiaire entre la mémoire centrale et les mémoires auxiliaires.

5. Les mémoires auxiliaires (appelées aussi mémoires périphériques ou mémoires de masse ou alors secondaire) sont des mémoires de grande capacité et de coût relativement faible ; qui permettent de stocker les informations pour une plus longue période que ne le fait la mémoire principale de capacité plus limitée. Ce sont par exemple les disques et disquettes, ou les bandes magnétiques.

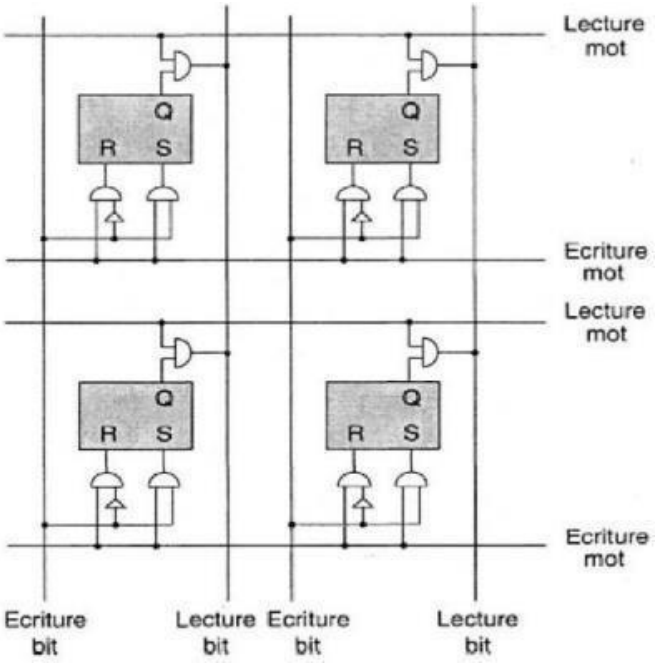


Figure 3. 20. Présentation d'une mémoire.

3.4.2.3. Caractéristiques d'une mémoire

~ **La capacité :** c'est le nombre total de bits que contient la mémoire. Elle s'exprime aussi souvent en octet.

~ **Le format des données :** c'est le nombre de bits que l'on peut mémoriser par case mémoire. On dit aussi que c'est la largeur du mot mémorisable.

~ **Le temps d'accès :** c'est le temps qui s'écoule entre l'instant où a été lancée une opération de lecture/écriture en mémoire et l'instant où la première information est disponible sur le bus de données.

~ **Le temps de cycle :** il représente l'intervalle minimum qui doit séparer deux demandes successives de lecture ou écriture.

~ **Le débit :** c'est le nombre maximum d'informations lues ou écrites par seconde.

~ **Volatilité :** elle caractérise la permanence des informations dans la mémoire. L'information stockée est volatile si elle risque d'être altérée par un défaut d'alimentation électrique et non volatile dans le cas contraire.

3.4.3. Utilisation des bascules pour réaliser des compteurs

3.4.3.1. C'est quoi un compteur ?

• Un compteur est un circuit séquentiel qui possède N états (E_0,E_1,…..,E_{n-1}).

• À chaque top d'horloge, il passe de l'état E_i à l'état E_{i+1}.

• Il revient toujours à l'état initiale E_0 : Un compteur possède un cycle (une séquence d'états).

• Un compteur est constitué de n bascules.

• Le nombre d'états d'un compteur est inférieur ou égale à 2^n.

Caractéristiques générales des compteurs [14]:

– commande d 'horloge (synchrone ou asynchrone)

– capacité de comptage

– code de comptage

– vitesse de comptage

– comptage/décomptage

– possibilités de présélection

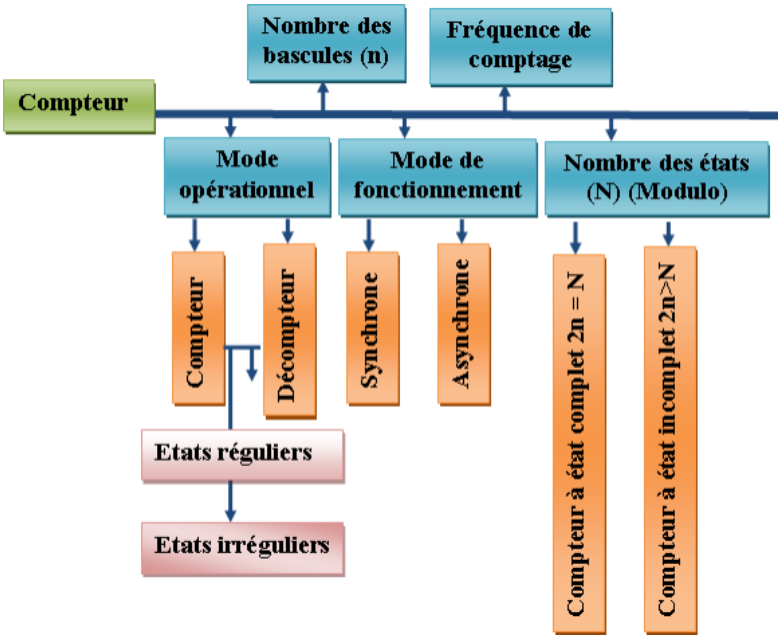


Figure 3. 21. Caractéristiques principales des compteurs [15].

3.4.3.2. Types de compteur

Suivant le mode de fonctionnement il existe deux types de compteurs :

- Les compteurs synchrones ou parallèles.
- Les compteurs asynchrones ou séries.

3.4.3.2.1. Compteurs synchrones

Dans un compteur synchrone, toutes les bascules reçoivent, en parallèle, le même signal d'horloge. C'est-à-dire le signal d'horloge synchronise toutes les bascules simultanément.

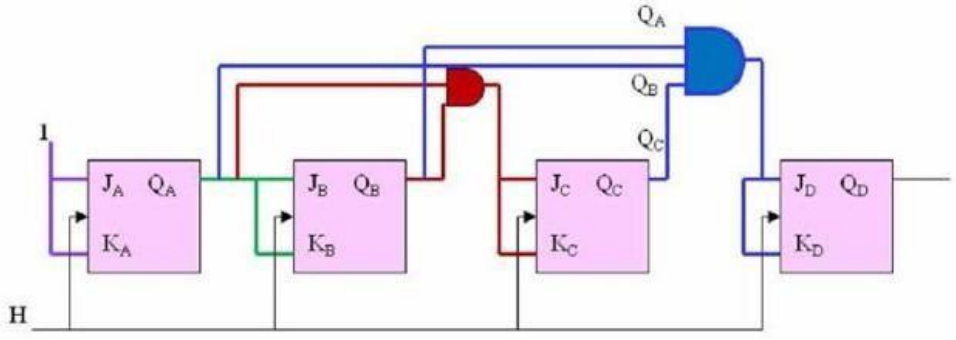


Figure 3. 22. Exemple d'un compteur synchrone à base de bascules JK.

Suivant le cycle réalisé, il existe trois types de compteurs :

a. Les compteurs synchrones modulo 2^n (cycle complet)

- n=2 : 0→1→2→3→0 : modulo 4
- n=3 : 0→1→2→3→4→5→6→7→0 : modulo 8
- n=4 : 0→1→2→3→4→5→6→7→8→9→10→11→12→13→14→15→0 : modulo 16

b. Les compteurs synchrones modulo N (cycle incomplet)

- Pour N=5 : 0→1→2→3→4→0 →modulo 5
- Pour N= 10 : 0→1→2→3→4→5→6→7→8→9→0 : modulo 10

c. Les compteurs synchrones à cycle quelconque

- Exemple : 0→2→5→6→8→10→0

3.4.3.2.1.1. Étude des compteurs synchrones modulo 2^n

Exemple1 : réalisation d'un compteur synchrone modulo 2^2 =4 à l'aide des bascules JK (le cycle :0,1,2,3)

Diagramme d'état

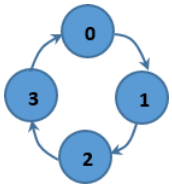


Table de vérité

Pour faire décrire au compteur une séquence déterminée, il faut, à chaque impulsion d'horloge, définir les entrées synchrones J et K. Pour cela, on utilise la table de transition de la bascule J-K (Tableau ci-dessous).

Q_1	Q_0	Q_1^+	Q_0^+	J_1	K_1	J_0	K_0
0	0	0	1	0	X	1	X
0	1	1	0	1	X	X	1
1	0	1	1	X	0	1	X
1	1	0	0	X	1	X	1

Les équations des entrées des bascules

Q_1 / Q_0	0	1
0	0	X
1	1	X

$J_1 = Q_0$

Q_1 / Q_0	0	1
0	X	0
1	X	1

$K_1 = Q_0$

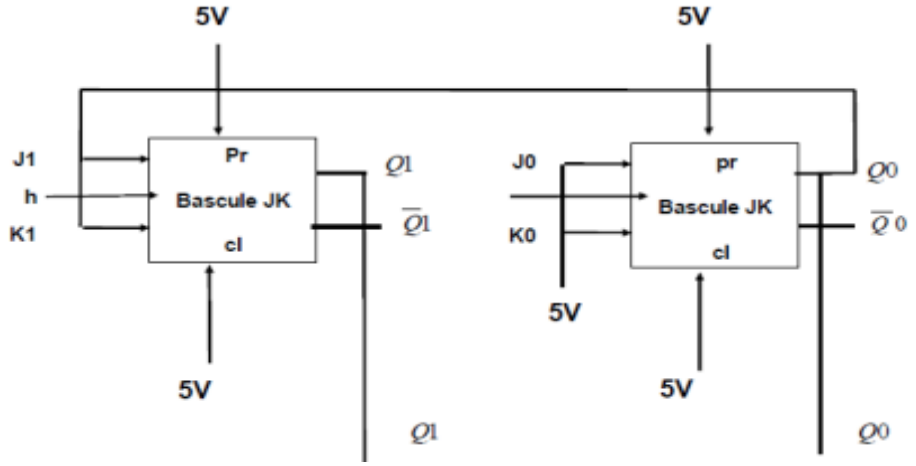
Q_1 / Q_0	0	1
0	1	1
1	X	X

$J_0 = 1$

Q_1 / Q_0	0	1
0	1	1
1	X	X

$K_0 = 1$

Schéma d'un compteur synchrone synchrones modulo 4



Exemple2 : réalisation d'un compteur synchrone modulo 2^3=8 à l'aide des bascules JK
Toutes les bascules possèdent la même horloge. Le cycle est : 0,1,2,3,4,5,6,7.

Diagramme d'état

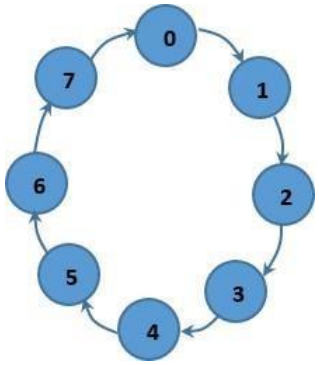


Table de vérité

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

Les équations des entrées des bascules

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$J_2 = Q_1 Q_0$

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	X	X	0	0
1	X	X	0	1

$K_2 = \overline{Q_1} Q_0$

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	0	X	X	0
1	1	X	X	1

$J_1 = Q_0$

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	X	0	0	X
1	X	1	1	X

$K_1 = Q_0$

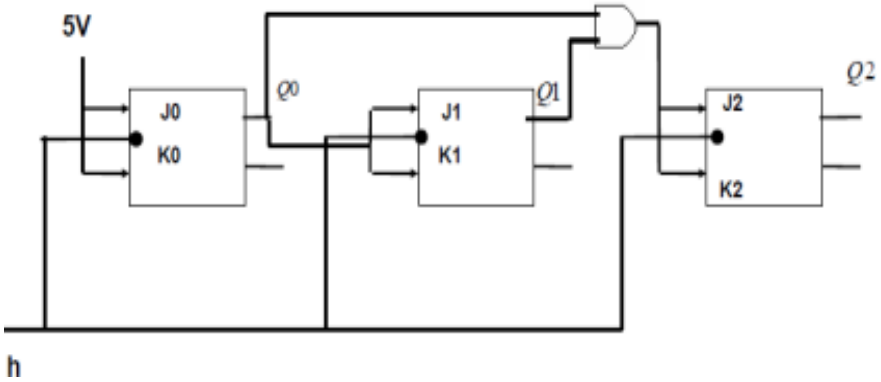
Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	1	1	1	1
1	X	X	X	X

$J_0 = 1$

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	X	X	X	X
1	1	1	1	1

$K_0 = 1$

Schéma d'un compteur modulo 8 synchrone



h

Exemple 3 : réalisation d'un compteur synchrone modulo 2^3 =8 à l'aide des bascules T (le cycle :0,1,2,3,4,5,6,7)

Une bascule T possède deux états : mémoire si T=0 et basculement si T=1.

Diagramme d'état

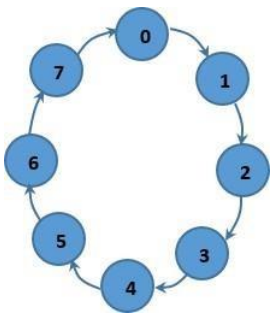


Table de vérité

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Les équations des entrées des bascules

Q_0 \ $Q_2 Q_1$	00	01	11	10
0	0	0	0	0
1	0	1	1	0

$T_2 = Q_1 Q_0$

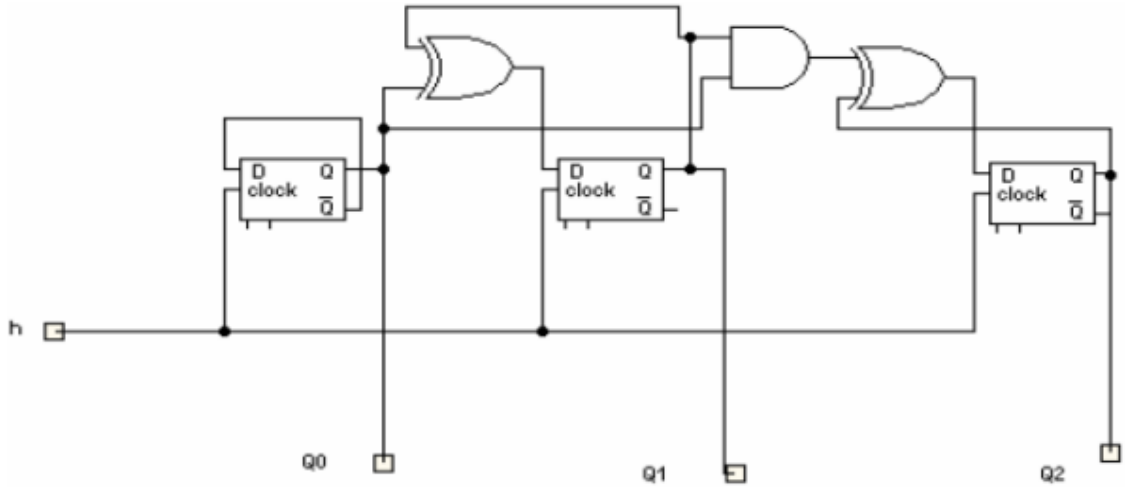
Q_0 \ $Q_2 Q_1$	00	01	11	10
0	0	0	0	0
1	1	1	1	1

$T_1 = Q_0$

Q_0 \ $Q_2 Q_1$	00	01	11	10
0	1	1	1	1
1	1	1	1	1

$T_0 = 1$

Schéma d'un compteur modulo 8 synchrone avec des bascules T



Exemple 4 : réalisation d'un compteur synchrone modulo 2^3 =8 à l'aide des bascules D (le cycle :0,1,2,3,4,5,6,7).

Une bascule D possède deux états : mémoire si D=0 et basculement si D=1.

Diagramme d'état

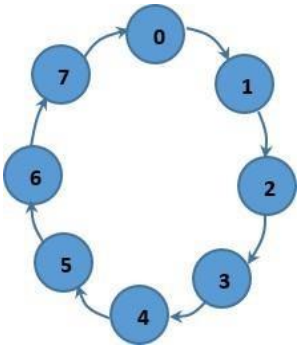


Table de vérité

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	D_2	D_1	D_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Les équations des entrées des bascules

Q_0 \ $Q_2 Q_1$	00	01	11	10
0	0	0	1	1
1	0	1	0	1

$$D_2 = Q_2Q_1Q_0 + Q_2\overline{Q_1} + Q_2\overline{Q_0}$$
$$= \overline{Q_2}Q_1Q_0 + Q_2\left(\overline{Q_1} + \overline{Q_0}\right)$$
$$= \overline{Q_2}Q_1Q_0 + Q_2\left(\overline{Q_1Q_0}\right)$$
$$= Q_2 \oplus (Q_1Q_0)$$

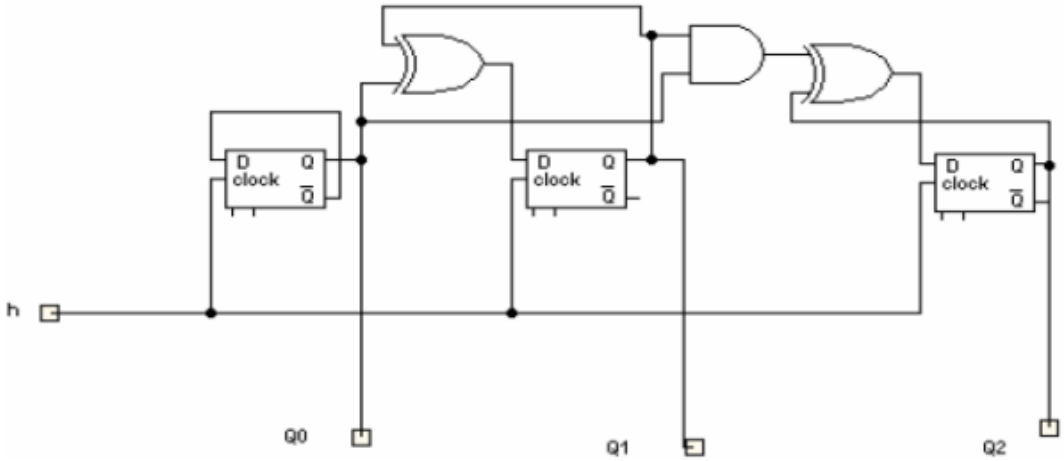
Q_0 \ $Q_2 Q_1$	00	01	11	10
0	0	1	1	0
1	1	0	0	1

$$D = Q\overline{Q} + \overline{Q}Q$$
$$= Q \oplus Q$$

Q_0 \ $Q_2 Q_1$	00	01	11	10
0	1	1	1	1
1	0	0	0	0

$$D_0 = \overline{Q_0}$$

Schéma d'un compteur modulo 8 synchrone avec des bascules D



3.4.3.2.1.2. Étude des Compteurs synchrones modulo N

Exemple 1 : réalisation d'un compteur Modulo 6 à l'aide des bascules D (le cycle :0,1,2,3,4,5)

Diagramme d'état

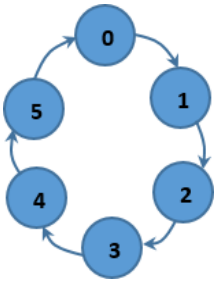


Table de vérité

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	D_2	D_1	D_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	0	0	0	0	0	0

Les équations des entrées des bascules

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	0	0	X	1
1	0	1	X	0

$$D_2 = Q_2\overline{Q_0} + Q_1\overline{Q_0}$$

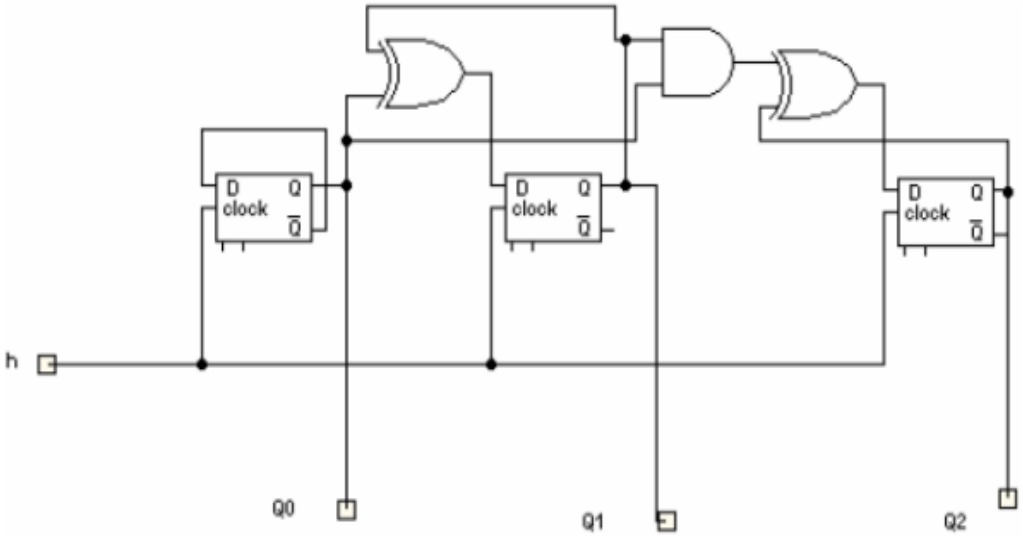
Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	0	1	X	0
1	1	0	X	0

$$D_1 = Q_2\overline{Q_1}Q_0 + Q_1\overline{Q_0}$$

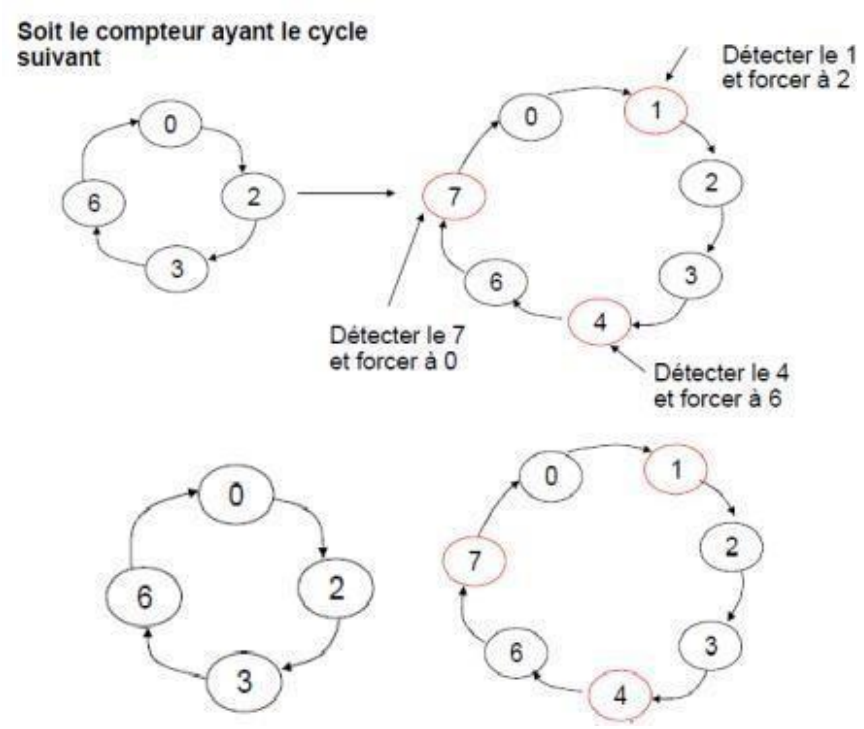
Q_0 \ $Q_2\ Q_1$	00	01	11	10
0	1	1	X	1
1	0	0	X	0

$$D_0 = \overline{Q_0}$$

Schéma d'un compteur modulo 6 synchrone avec des bascules D



3.4.3.2.1.3. Étude des compteurs synchrones à cycle quelconque



•Pour forcer le compteur d'un état à un autre il faut agir sur les entrées des bascules.

•Pour les états qui n'appartiennent pas au cycle du compteur il faut les considérer comme étant des états indéterminés

Exemple : réalisation d'un compteur synchrone **à cycle quelconque** à l'aide des bascules JK

Toutes les bascules possèdent la même horloge. Le cycle est :0,2,3,6.

Diagramme d'état

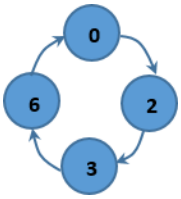


Table de vérité

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	1	0	0	X	0	X	1	X
0	1	0	0	1	1	0	X	1	X	X	1
0	1	1	1	1	0	0	X	X	0	1	X
1	1	0	0	0	0	1	X	X	1	X	1

Les équations des entrées des bascule

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$J_2 = Q_1 Q_0$

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	X	X	0	0
1	X	X	0	1

$K_2 = \overline{Q_1} Q_0$

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	0	X	X	0
1	1	X	X	1

$J_1 = Q_0$

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	X	0	0	X
1	X	1	1	X

$K_1 = Q_0$

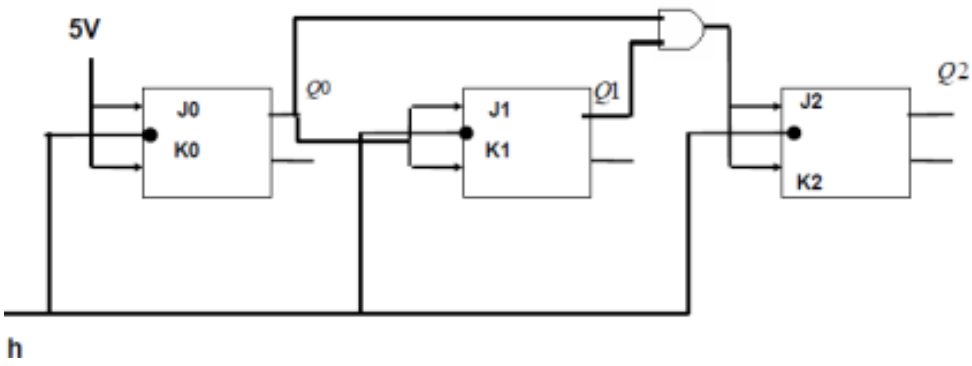
Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	1	1	1	1
1	X	X	X	X

$J_0 = 1$

Q_0 \ $Q_2\,Q_1$	00	01	11	10
0	X	X	X	X
1	1	1	1	1

$K_0 = 1$

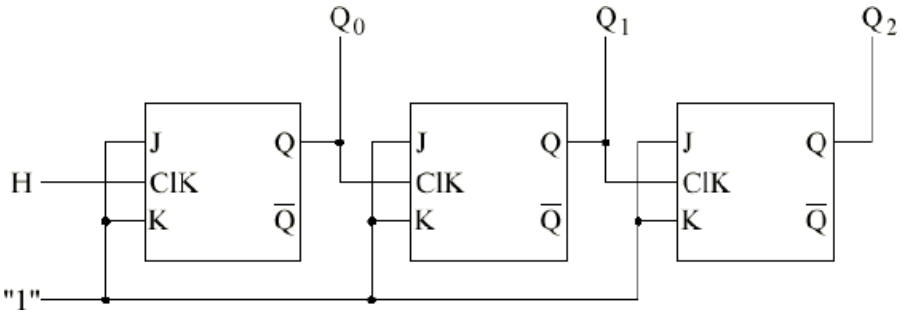
Schéma d'un compteur synchrone modulo 8



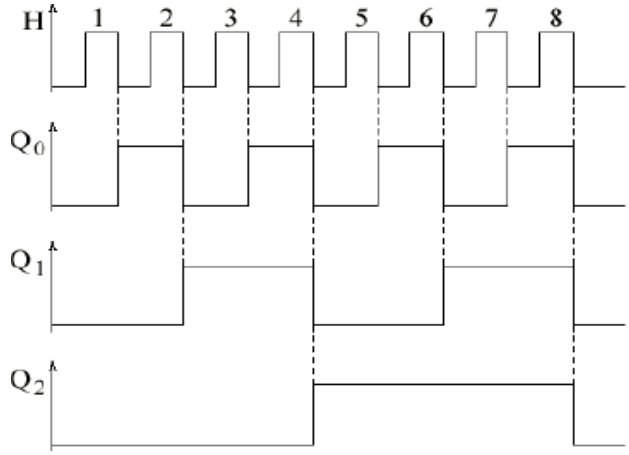
h

3.4.3.2.2. Compteurs asynchrones ou séries

Un compteur asynchrone est un système logique composé de bascules dans lesquels les impulsions que l'on applique à l'entrée doivent traverser la première bascule avant de pouvoir commander la seconde et ainsi de suite jusqu'à la dernière bascule [16]. C'est-à-dire le signal d'horloge n'est reçu que par le premier étage (bascule LSB : *Least Significant Bit*). Pour chacune des autres bascules le signal d'horloge est fourni par une sortie de la bascule de rang immédiatement inférieur. Exemple d'un compteur asynchrone à base de bascules JK.



Dans cet exemple la sortie Q_0 bascule sur chaque front descendant du signal d'horloge. La sortie Q_1 change d'état à chaque transition 1 vers 0 de la sortie Q_0. De même le basculement de la sortie Q_2 est déclenché par une transition 1 vers 0 de la sortie Q_1. Comme le montre les chronogrammes suivants.



Suivant le cycle réalisé, il existe trois types de compteurs asynchrones :

a. Les compteurs asynchrones modulo 2^n (cycle complet)

- n=2 : 0→1→2→3→0 : modulo 4
- n=3 : 0→1→2→3→4→5→6→7→0 : modulo 8
- n=4 : 0→1→2→3→4→5→6→7→8→9→10→11→12→13→14→15→0 : modulo 16

b. Les compteurs asynchrones modulo N (cycle incomplet)

- Pour N=5 : 0→1→2→3→4→0 →modulo 5
- Pour N= 10 : 0→1→2→3→4→5→6→7→8→9→0 : modulo 10

c. Les compteurs asynchrones à cycle quelconque

- Exemple : 0→2→5→6→8→10→0

Exemple 1 : réalisation d'un compteur asynchrone modulo 10 avec des bascules D.

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	**0**	**1**	**0**

Remise à zéro

On veut que le compteur passe à $(0000)_2=(0)_{10}$ lorsqu'il passe de $(1001)_2$ à $(1010)_2$.

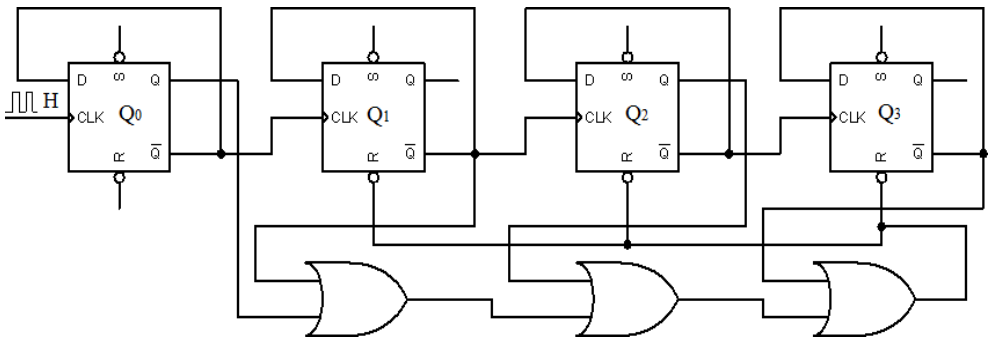
$(\mathbf{1010})_2$

$(\mathbf{0000})_2$

On veut que le compteur passe à 0 lorsqu'il atteint : 1010 = $(1010)_2$. Pour cela on peut écrire l'expression logique :

$$R = \overline{Q_3 \overline{Q_2} Q_1 \overline{Q_0}} = \overline{Q_3} + Q_2 + \overline{Q_1} + Q_0$$

Ce n'est pas suffisant de remettre à zéro Q_3 et Q_1 ($(1010)_2$ c'est-à-dire $Q_3 = 1$, $Q_2 = 0$, $Q_1 = 1$ et $Q_0 = 0$) ; il faut mettre Q_2 aussi à zéro. Sinon, lorsque Q_1 passe de 1 à 0, Q_2 passe de 0 à 1(Q_1 horloge de la bascule Q_2).



Exemple 2 : réalisation d'un compteur asynchrone modulo 11 avec des bascules D.

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	**0**	**1**	**1**

Remise à zéro

On veut que le compteur passe à $(0000)_2=(0)_{10}$ (l'entrée R (RESET) soit à 0) lorsqu'il passe de $(1010)_2$ à $(1011)_2$.

$(1011)_2$

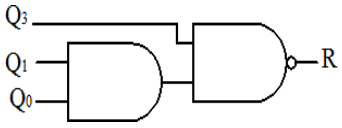
$(0000)_2$

On veut que le compteur passe à 0 lorsqu'il atteint : $(11)_{10} = (1011)_2$. Pour cela on peut écrire l'expression logique :

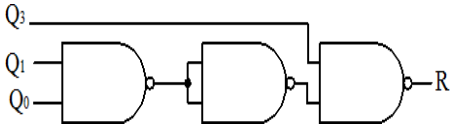
$$R = \overline{Q_3 Q_2 Q_1 Q_0} = \overline{Q_3} + Q_2 + \overline{Q_1} + \overline{Q_0}$$

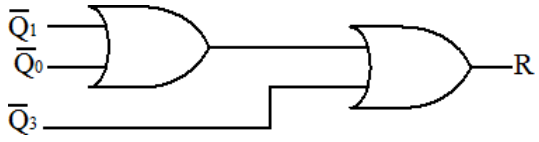
Ce n'est pas suffisant de remettre à zéro Q_3 , Q_1 et Q_0 ($(1011)_2$ c'est-à-dire $Q_3 = 1$, $Q_2 = 0$, $Q_1 = 1$ et $Q_0 = 1$) ; On peut simplifier cette relation logique en ne tenant compte que des sorties à 1. En effet c'est la 1ère fois que Q_3, Q_1 et Q_0 soient à 1. On peut donc utiliser :

$$R = \overline{Q_3 Q_1 Q_0} = \overline{Q_3} + \overline{Q_1} + \overline{Q_0}$$

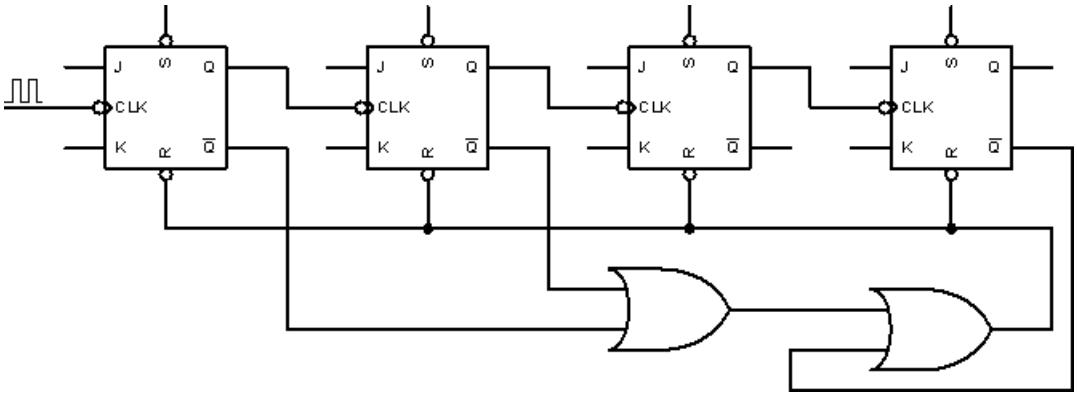


Ou





Ou



Quand des bascules D, ou JK câblées en diviseur par deux (J=K=1, ces entrées étant reliées au plus de l'alimentation), sont disposées en cascade, la première impulsion d'horloge change l'état de la première bascule, la deuxième impulsion la fait changer à nouveau d'état et ceci provoque le changement de la deuxième bascule, etc [17].

3.4.4. Utilisation des bascules pour réaliser Les décompteurs

Exemple 1 : réalisation d'un décompteur asynchrone modulo 10 à l'aide des bascules JK

Q_3	Q_2	Q_1	Q_0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0
1	**1**	**1**	**1**

Remise à neuf

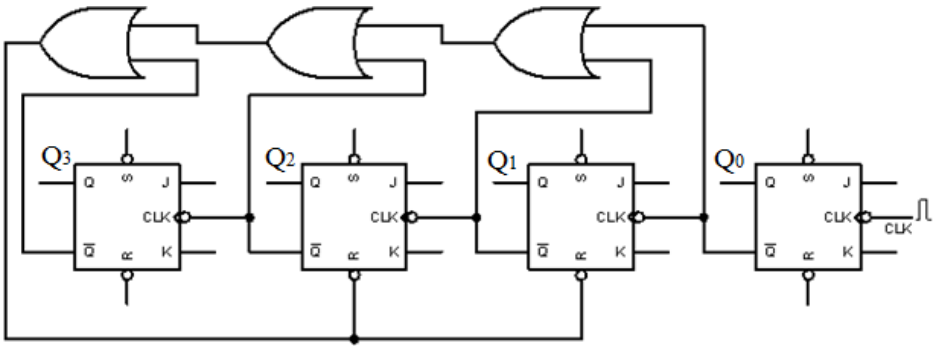
On veut que le décompteur passe à $(1001)_2=(9)_{10}$ lorsqu'il passe de $(0000)_2$ à $(1111)_2$

$(\mathbf{1111})_2$

$(\mathbf{1001})_2$

Pour cela, il suffit de remettre à zéro Q_2 et Q_1. On veut que le décompteur passe à 9 lorsqu'il atteint : $(15)_{10} = (1111)_2$. Pour cela on peut écrire l'expression logique :

$$R = \overline{Q_3 Q_2 Q_1 Q_0} = \overline{Q_3} + \overline{Q_2} + \overline{Q_1} + \overline{Q_0}$$



Exemple 2 : réalisation d'un décompteur asynchrone de cycle suivant : 9 → 8→7→ 6 → 5 → 4→ 9 à l'aide des bascules JK.

Q_3	Q_2	Q_1	Q_0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	**0**	**1**	**1**

Remise à neuf

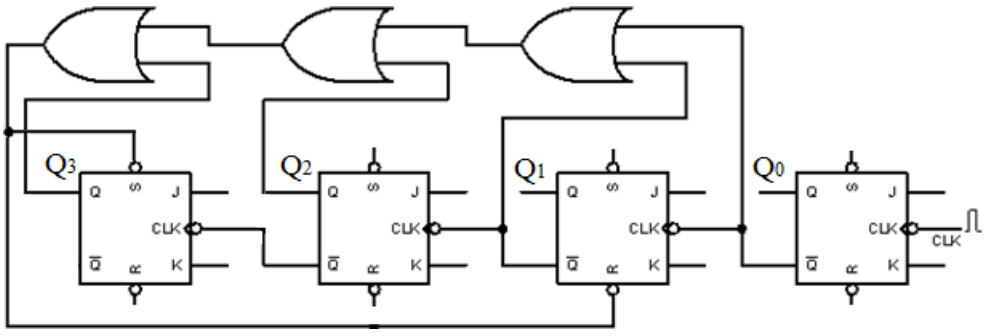
On veut que le décompteur passe à $(1001)_2=(9)_{10}$ lorsqu'il passe de $(0100)_2$ à $(0011)_2$

$(0011)_2$

$(1001)_2$

Pour cela, il suffit de remettre Q_1 à zéro (R_1) et Q_3 à 1(S_3).

$$R_1 = S_3 = \overline{\overline{Q_3}\,\overline{Q_2}\,Q_1 Q_0} = Q_3 + Q_2 + \overline{Q_1} + \overline{Q_0}$$



3.4.5. Utilisation des bascules pour réaliser Les compteurs/décompteurs

• Le même circuit peut offrir l'opération de comptage et décomptage

• Rajouter une entrée C, si C=0 alors comptage, si C=1 alors décomptage

Exemple 1 : un compteur/décompteur synchrone modulo 8 à l'aide de bascules T

Table de vérité

C	Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	T_2	T_1	T_0
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	1	1	1	1	1	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	0	1	1	0	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	0	1	1	0	1	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	0	1	0	0	0	0	0	1
1	0	0	0	1	1	1	1	1	1

Les équations des entrées des bascules

$Q_1\,Q_0$ \ C Q_2	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	0	0
10	0	0	0	0

$$T_2 = \overline{C}Q_1Q_0 + C\overline{Q_1}\,\overline{Q_0}$$

$Q_1\,Q_0$ \ C Q_2	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$$T_1 = \overline{C}Q_0 + C\overline{Q_0}$$

$Q_1\,Q_0$ \ C Q_2	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$$T_0 = 1$$

Exemple 2 : un compteur/décompteur asynchrone modulo 8 à l'aide de bascules T

Table de vérité

C	Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	T_2	T_1	T_0
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	1	1	1	1	1	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	0	1	1	0	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	0	1	1	0	1	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	0	1	0	0	0	0	0	1
1	0	0	0	1	1	1	1	1	1

Les équations des entrées des bascules

$Q_1\,Q_0$ \ C Q_2	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	0	0
10	0	0	0	0

$$T_2 = \overline{C}Q_1Q_0 + C\overline{Q_1}\overline{Q_0}$$

$Q_1\,Q_0$ \ C Q_2	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$$T_1 = \overline{C}Q_0 + C\overline{Q_0}$$

$Q_1\,Q_0$ \ C Q_2	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$$T_0 = 1$$

3.5. Synthèse de circuits séquentiels

La théorie d'automates finis représente un modèle théorique très utile pour la synthèse des circuits séquentiels trop complexes.

3.5.1. Définition d'une machine à états finis

Un automate fini (MEF : Machine à Etats Finis ou en anglais FSM : *Finite State Machine*) ou automate fini est un système séquentiel qui peut se trouver dans un nombre fini d'états et notamment des mémoires. Une machine d'états finis comporte une partie combinatoire et les bascules d'états. La partie combinatoire contient la logique de calcul des sorties et des états suivants [18]. On peut les utiliser pour mémoriser des informations. Autres appellations : (Machine à états, Machine à états finis, Séquenceur, Contrôleur, Automate, Machine séquentielle ou Machine séquentielle algorithmique).

Un automate ne peut prendre qu'un nombre fini d'état (ou noeuds). Il est caractérisé par :

- Sa sortie s.
- Sa entrée e.
- Son état q.

D'où, formellement une FSM est un sextuplait $M = (Q, U, Y, Init, R, S)$, où

Q : un ensemble fini d'états de la machine.

U : un ensemble fini de signaux d'entrée, l'alphabet d'entrée.

Y : un ensemble fini de signaux de sortie, l'alphabet de sortie.

$Init \subseteq Q$: Un ensemble d'états initiaux.

$R : Q \times U \to Q$, une fonction appelée fonction de transition.

$S : Q \times U \to Y$, une fonction appelée fonction de sortie.

Remarque : On peut avoir une FSM où tout état peut être un état initial donc, inutile de spécifier l'ensemble des états initiaux ou sans sortie donc, la fonction Y n'a aucun sens, …etc. Les automates sont dits synchrones lorsque le passage d'un état (état présent) à l'état suivant (état futur) a lieu sur une transition d'un signal appelé horloge commun à toutes les bascules de l'automate [19].

On peut représenter une machine par :

- Un chronogramme.
- Une équation logique.
- Une table de transition.
- Un diagramme d'état.
- Une structure logique.

Si les concepts de chronogramme, équation logique et structure logique sont connus les notions de tables de transitions et diagramme d'état restent à définir.

3.5.1.1. Tables de transitions

Les diagrammes de transitions ou tables de transitions ou matrices de transitions décrivent sous forme matricielle les fonctions R et S. Elles doivent contenir les états qi à l'instant t, les entrées ei, les états futurs qi^+ et les sorties si.

Exemple

Soit un automate $M=(Q,U,Y,R,S)$ définie par : $Q=U=Y=\{0,1\}$ et dont les fonctions de transition et de sortie sont définies par les tables suivantes :

R	0	1
0	0	1
1	0	1

S	0	1
0	0	0
1	1	1

-Catégories de MSA

- **Synchrone**

Un automate est dit synchrone lorsque le passage d'un état (état présent) à l'état suivant (état future) a lieu sur une transition d'un signal appelé horloge commun à toutes les bascules de l'automate.

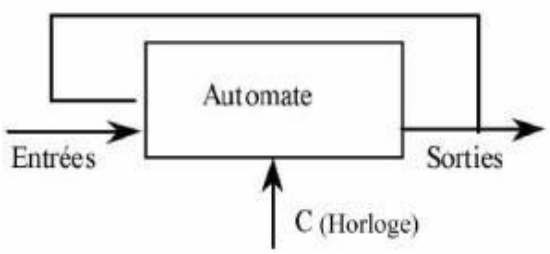


Figure 3. 23. Automate synchrone

- **Principalement synchrone**
- **Asynchrone**

3.5.1.2. Diagramme d'état

Une autre façon de représenter une machine est l'utilisation d'un graphe de transition, soit une description équivalente des tables de transitions,

– les états (contenus possibles du registre d'état) sont représentés par des cercles.

– les transitions (possibilités de passage d'un état à l'autre) par des arcs orientés, allant de l'état initial à l'état final.

-les états initiaux par deux cercles.

Figure 3. 24. Exemples de diagramme d'états.

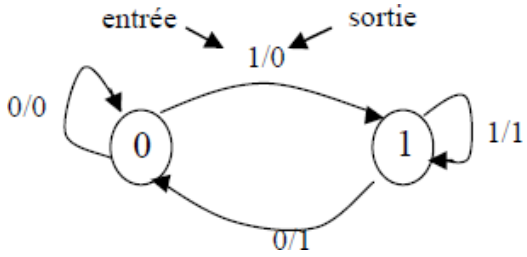
Remarques

- Une transition peut être inconditionnelle, c'est-à-dire que si le système est dans l'état source considéré, la transition se produit lors du front actif d'horloge suivant.
- Le plus souvent, une transition est conditionnelle, c'est-à-dire que quand le système est dans l'état source, la transition se produit lors du front actif d'horloge suivant si une condition sur les entrées est vérifiée.
- Il peut y avoir maintien d'un état pour certaines valeurs d'entrée. Dans ce cas, l'arc orienté qui représente la transition se referme sur le même cercle.

Exemples

1. Diagramme d'état de l'exemple précédent.

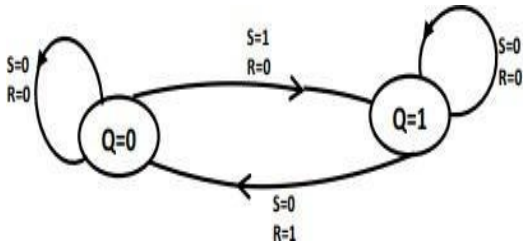
Il suffit de traduire les données des tables de transitions en diagrammes d'états.

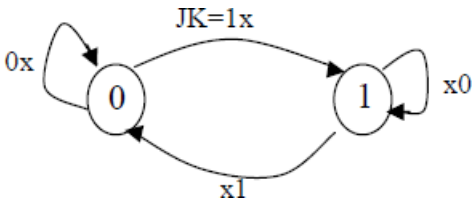


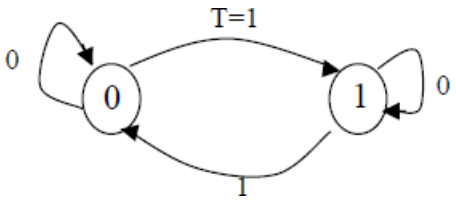
2. Diagrammes d'état des bascules RS, JK, T et D

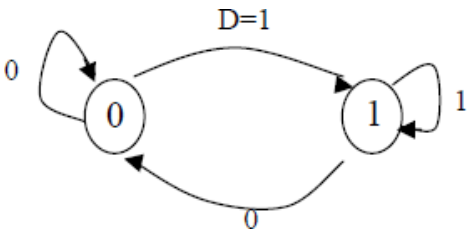
Rappelons les tables caractéristiques réduites des bascules JK, T et D

Q	Q^+	J	K	T	D
0	0	0	X	0	0
0	1	1	X	1	1
1	0	X	1	1	0
1	1	X	0	0	1









Structure des machines à états finis

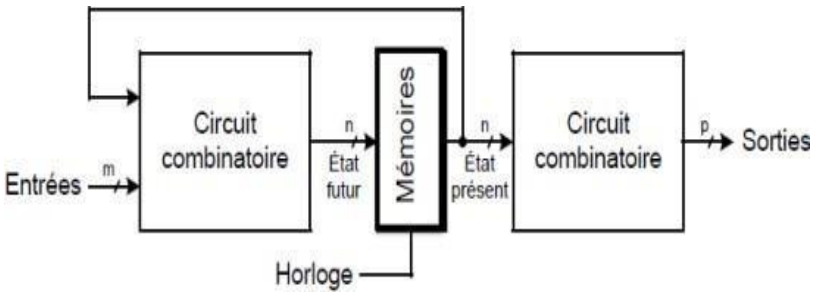


Figure 3. 25. Exemple d'une machine à états finis.

3.5.2. Classes de MSA

Souvent, on distingue entre deux modèles de machines d'états finis à savoir le modèle de Moore et modèle de Mealy [18]. Ils diffèrent seulement par la manière dont la sortie est générée.

3.5.2.1. Machine de Moore

Dans une machine de Moore l'état des sorties est fonction seulement de l'état actuel de la machine, résultant de l'état actuel des entrées et de l'état antérieur de la machine. C'est le vecteur d'état, qui permet de prendre en compte l'état antérieur de la machine.

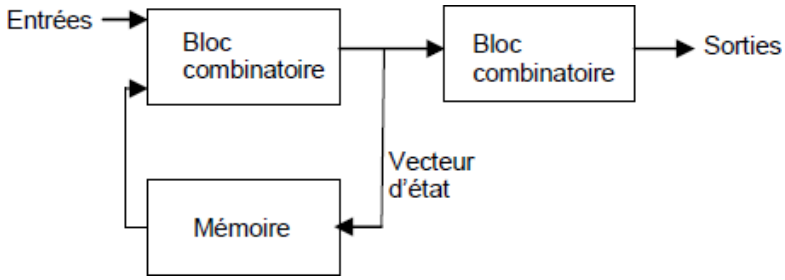


Figure 3. 26. Structure d'une machine de Moore.

3.5.2.2. Machine de Mealy

Dans une machine de Mealy, l'état des sorties est déterminé non seulement à partir de l'état actuel de la machine, matérialisé par le vecteur d'état, mais aussi de l'état actuel des entrées.

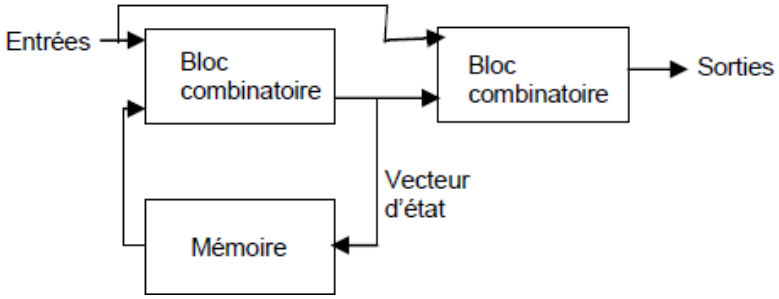


Figure 3. 27. Structure d'une machine de Mealy.

3.5.3. Analyse d'un circuit séquentiel

Analyser un circuit séquentiel c'est déterminer son rôle. Pour analyser un circuit séquentiel, on peut suivre les étapes suivantes :

1. Déterminer les fonctions des variables d'entrée.
2. Dresser la table caractéristique du circuit. De cette table, déduire les variables de sortie en se basant sur les expressions logiques des variables d'entrée. Elle a la forme suivante :

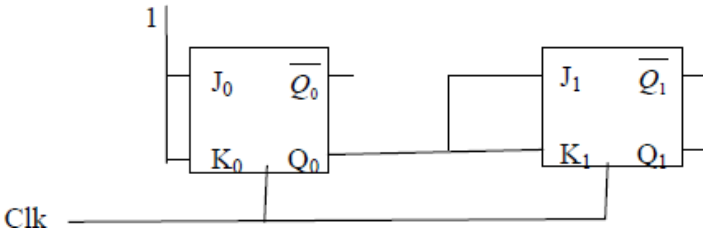
Variables d'entrée	Q	Q^+
connue	connue	A déterminer

Pour déterminer les états de sortie Q+, il faut utiliser la table caractéristique dont il est question dans le circuit à analyser.

3. Déduire le rôle du circuit analysé.

Exemple

Analysons le circuit suivant :



Détermination des fonctions d'entrée de chaque bascule.

$$\text{bascule } 0 \begin{cases} J_0 = 1 \\ K_0 = 1 \end{cases} \qquad \text{bascule } 1 \begin{cases} J_1 = Q_0 \\ K_1 = Q_0 \end{cases}$$

Table caractéristique.

Q_1	Q_0	J_1	K_1	J_0	K_0	Q_1^+	Q_0^+
0	0	0	0	1	1	0	1
0	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0

Conclusion : C'est un compteur binaire comptant de 0 jusqu'à 3.

3.5.4. Synthèse d'un circuit séquentiel

La synthèse d'un circuit consiste à élaborer le circuit logique à partir du cahier de charge. Pour faire la synthèse d'un circuit on peut suivre les étapes suivantes :

1. Etablir la table d'excitation correspondant au circuit à réaliser. Elle est de la forme :

Q	Q^+	Variables d'entrée
connue	connue	A déterminer

2. Déduire l'expression de chaque variable d'entrée.
3. Réaliser le circuit en utilisant les bascules et les portes logiques requises.

Chapitre IV : Les Circuits Intégrés

4.1. Introduction

Un circuit intégré désigne un bloc constitué par un monocristal de silicium (Puce) de quelques millimètres carrés à l'intérieur duquel se trouve inscrit en nombre variable des composants électroniques élémentaires (Transistors, diodes, résistances, condensateurs, ...).

Les circuits intégrés logiques sont classés suivant leur technologie de fabrication (bipolaire TTL, bipolaire ECL, MOS,…). Pour un fonctionnement logique identique, chaque technologie offre des performances différentes sur le plan électrique (tensions, courants, puissances) et temporel (rapidité).

Une famille logique est caractérisée par ses paramètres électriques :

- La plage des tensions d'alimentation et la tolérance admise sur cette valeur,
- La plage des tensions associée à un niveau logique, en entrée ou en sortie,
- Les courants pour chaque niveau logique, en entrée ou en sortie,
- Le courant maximum que l'on peut extraire d'une porte logique et le courant absorbé en entrée,
- La puissance maximale consommée qui dépend souvent de la fréquence de fonctionnement.

Les performances dynamiques principales sont :

- Les temps de montée (transition bas–haut) et de descente (transition haut–bas) des signaux en sortie d'une porte,
- Les temps de propagation d'un signal entre l'entrée et la sortie d'une porte logique.

Les différentes notions abordées seront illustrées de valeurs numériques issues de la technologie TTL.

4.2. Définition d'un circuit intégré

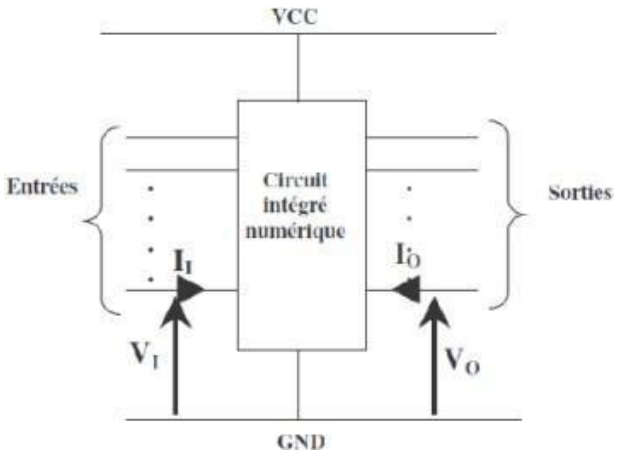


Figure 4.1. Représentation d'un circuit intégré.

- V_{CC} : tension d'alimentation : niveau de tension nécessaire pour alimenter le circuit.
- V_{IH} (min) : tension d'entrée niveau HAUT : niveau de tension nécessaire pour avoir un 1 logique en

entrée.

- V_{IL} (max) : tension d'entrée niveau BAS : niveau de tension nécessaire pour avoir un 0 logique en entrée.
- V_{OH} (min) : tension de sortie niveau HAUT : niveau de tension de la sortie d'un circuit logique correspondant à l'état logique 1.
- V_{OL} (max) : tension de sortie niveau BAS : niveau de tension de la sortie d'un circuit logique correspondant à l'état logique 0.

Les courants pour chaque niveau logique, en entrée ou en sortie,

- I_{IH} : courant d'entrée niveau HAUT : le courant qui traverse une borne d'entrée quand une tension niveau haut est appliquée à cette entrée.
- I_{IL} : courant d'entrée niveau BAS : le courant qui traverse une borne d'entrée quand une tension niveau bas est appliquée à cette entrée.
- I_{OH} : courant de sortie niveau HAUT : le courant qui traverse une borne de sortie placée au niveau logique 1 dans des conditions de charge spécifiées.
- I_{OL} : courant de sortie niveau BAS : le courant qui traverse une borne de sortie placée au niveau logique 0 dans des conditions de charge spécifiées.

4.3. Tension caractéristique

a. Tension d'alimentation

Les circuits intégrés sont alimentés sous une tension nominale Vcc :

Famille	Tension
TTL(série 74)	V_{cc}=5V +5%
TTL(série 54)	V_{cc}=5V+10%
CMOS4000	V_{dd}=3 à 15V

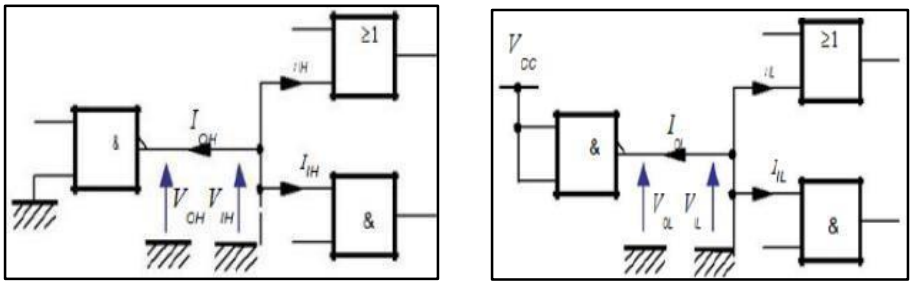


Figure 4.2. Au niveau logique haut (HIGH LEVEL), Au niveau logique bas (LOW LEVEL).

b. Classes d'intégrations

Dans l'ordre chronologique, on distingue 4 classes d'intégration :

- Les microcircuits SSI (Single Size Intégration) : ≈ 100 transistors par cm^2.
- Les circuits intégrés MSI (Médium Size Intégration) : ≈ 1000 transistors par cm^2.
- Les circuits LSI (Large Size Intégration) : ≈ 10000 à 100000 transistors par cm^2.
- Les circuits VLSI (Very Large Size Intégration) : ≈ 0.1 à 1 million transistors par cm^2.

c. Gabarit de tension

Un niveau logique correspond à une plage de tensions : le niveau logique 1 (entre Vcc et une limite inférieure à Vcc) et le niveau 0 (de 0 V à une limite supérieure).

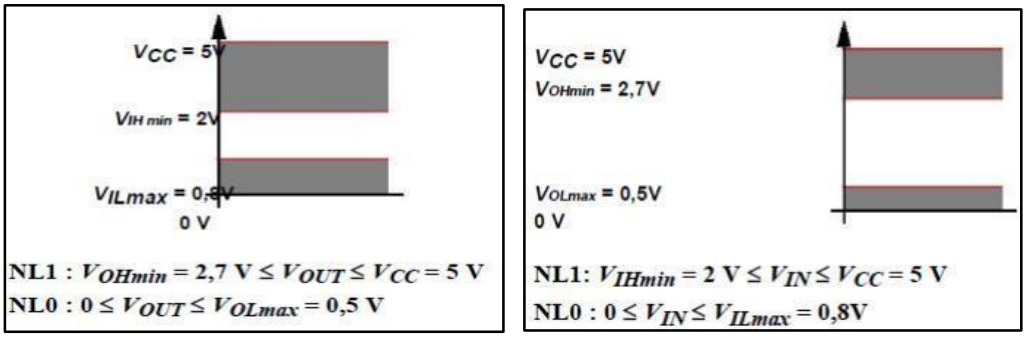


Figure 4.3. Gabarit des tensions : a) d'entrée, b) de sortie

d. Gabarit de transfert

Les deux graphes précédents sont rassemblés en un seul pour traduire la fonction logique entre ces tensions : c'est le gabarit de transfert.

Une porte satisfait le gabarit si sa courbe de transfert se trouve dans la partie non grisée.

La tension de basculement, notée V_T (T pour threshold, seuil), correspond à la tension d'entrée pour laquelle la sortie change d'état (approximativement l'intersection de la tangente au point d'inflexion de la courbe avec l'axe V_{IN}).

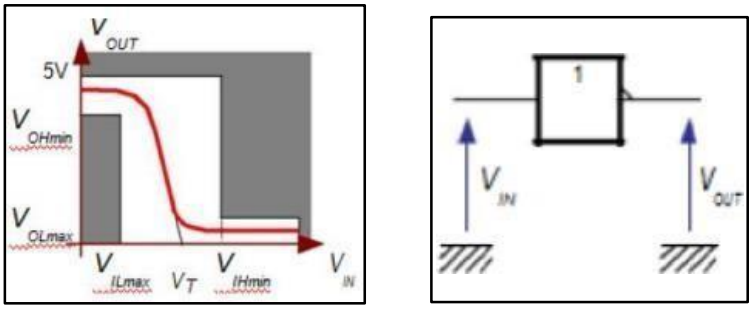


Figure 4.4. Caractéristique de transfert d'une porte inverseuse.

e. Compatibilité des niveaux logiques

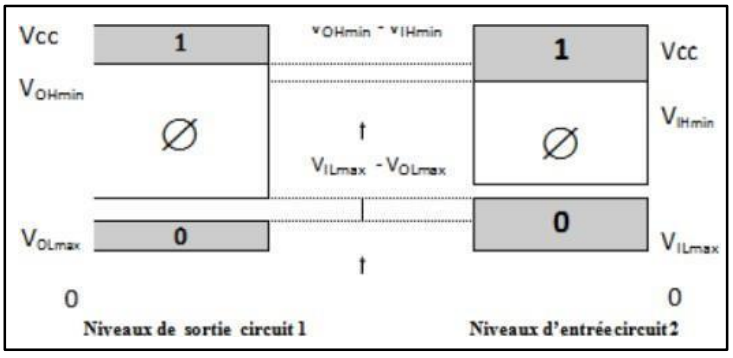


Figure 4.5. Compatibilité des niveaux logiques.

- Compatibilité au niveau haut : Il faut que $V_{OHmin} > V_{IHmin}$
- Compatibilité au niveau bas : Il faut que $V_{ILmax} > V_{Olmax}$

f. Temps moyen de propagation

Lorsqu'on applique à l'entrée d'un circuit un niveau logique, il y a un certain retard pour que la sortie réagie. Cette durée est le temps moyen de propagation t_{PD}.

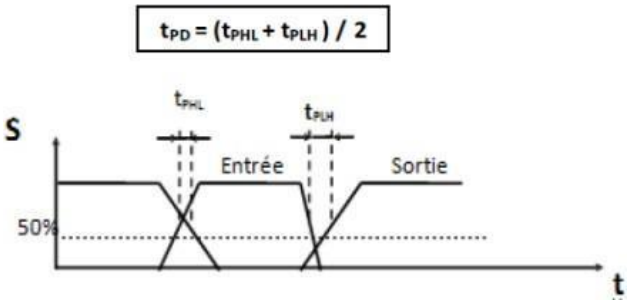


Figure 4.6. Temps moyen de propagation.

- t_{PHL}: Temps de propagation du niveau haut au niveau bas.
- t_{PLH}: Temps de propagation du niveau bas au niveau haut.

Remarque : Ce temps détermine la fréquence maximale F_{MAX} à laquelle les circuits intégrés sont capables de réagir.

g. Facteur de charge : Sortance N

Ce paramètre caractérise le nombre N maximal d'entrées de portes logiques pouvant être commandées par la sortie d'un autre opérateur logique de la même famille.

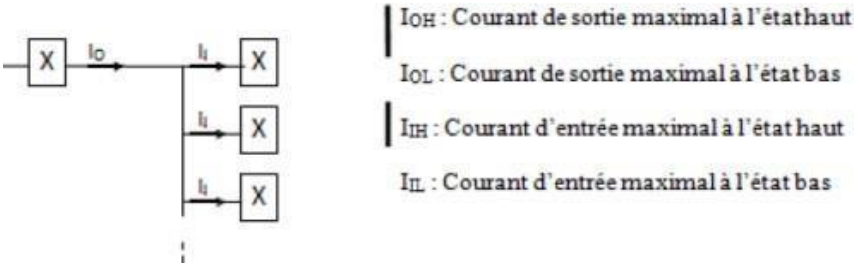


Figure 4.7. Facteur de charge.

- Sortance N (A l'état haut) = I_{OH}
- Sortance N (A l'état bas) = I_{OL}/I_{IL}

4.4. Présentation des CI

Un circuit intégré renferme plusieurs portes logiques, dont les entrées et les sorties sont accessibles sur les différentes bornes du circuit intégré. On peut remarquer sur le haut des circuits intégrés un petit creux appelé « ergo ». L'ergo permet d'orienter correctement le circuit intégré afin de repérer les différentes bornes. Les bronches (pin) d'un circuit intégré sont numérotées. Pour déterminer la position du pin n°1, il faut repérer une encoche sur le composant tel que :

En regardant le circuit intégré avec l'ergo vers le haut, la borne n°1 est la borne située en haut à gauche, les autres bornes sont numérotées en tournant dans le sens inverse des aiguilles d'une montre.

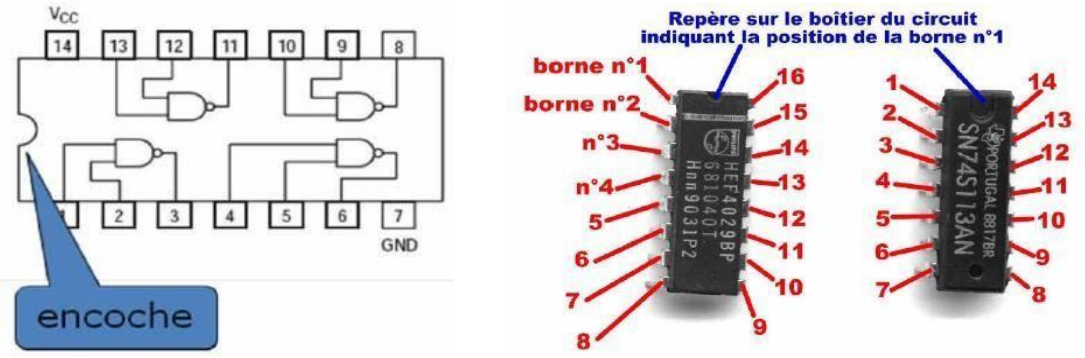


Figure 4.8. Présentation des circuits intégrés.

4.5. Identification des CI

Chaque circuit intégré possède une référence imprimée sur le dessus de son boîtier. Cette référence est composée de 4 à 7 caractères (chiffres et/ou lettres).

Il existe 2 grandes familles de circuits intégré.

- ➲ Les circuits dont la référence est de la forme 4000, appelé technologie CMOS.
- ➲ Les circuits dont la référence commence par 74, appelé technologie TTL.

4.6. Classement des circuits intégrés

Les circuits intégrés numériques les plus simples sont des portes logiques (et, ou et non), les plus complexes sont les microprocesseurs et les plus denses sont les mémoires.

On trouve de nombreux circuits intégrés dédiés à des applications spécifiques (ou ASIC pour Application Specific Iintegrated Circuit).

Notamment pour le traitement du signal (traitement d'image, compression vidéo…) on parle alors de processeur de signal numérique (ou DSP pour Digital Signal Processor).

Une famille importante de circuits intégrés est celle des composants de logique programmable (FPGA, CPLD). Ces composants sont amenés à remplacer les portes logiques simples en raison de leur grande densité d'intégration.

Les circuits intégrés ont été classés en six familles de produits selon la densité d'intégration (nombre de portes par circuit ou nombre de transistors par circuit) :

- ➲ **SSI** : (**S**mall **S**cale **I**ntegration) circuit à faible capacité d'intégration.
- ➲ **MSI** : (**M**edium **S**cale **I**ntegration) circuit à moyenne capacité d'intégration.
- ➲ **LSI** : (**L**arge **S**cale **I**ntegration) circuit à haute capacité d'intégration.
- ➲ **VLSI** : (**V**ery **L**arge **S**cale **I**ntegration) circuit à très haute capacité d'intégration.
- ➲ **ULSI** : (**U**ltra **L**arge **S**cale **I**ntegration) circuit à une extrême capacité d'intégration.
- ➲ **GLSI** : (**G**iga **L**arge **S**cale **I**ntegration) circuit à une capacité d'intégration géante.

Nom	Année	Nombre de portes logique	Exemple
SSI	1961-1966	1-10	Portes logiques : AND, OR, NOT,….etc.
MSI	1967-1971	10-100	Bascules, compteurs, multiplexeurs, décodeurs,…etc.
LSI	1972-1980	100-10000	Mémoire de petite capacité, circuit logique programmable.
VLSI	1981-1990	10000-100000	Mémoire de capacité importante, microprocesseur.
ULSI	1990-2000	100000-1000000	Microprocesseur graphique.
GLSI	2000-aujourd'hui	>1000000	Pentium Dual Core microprocesseur.

Figure 4.9. Classement des circuits intégrés.

4.7. Technologies de fabrication de CI

Le Die d'un circuit intégré comprend sous des formes miniaturisées principalement des transistors, des diodes, des résistances, des condensateurs, plus rarement des inductances, car elles sont plus difficilement miniaturisables.

DTL (**D**iode **T**ransistor **L**ogic), rapidement éliminée en faveur du TTL (plus rapide) : technologie obsolète.

TTL (**T**ransistor **T**ransistor **L**ogic), (V_{CC}=5V, H=5V, L=0V) meilleur compromis vitesse/consommation avant l'apparition du CMOS.

ECL (**E**mitter **C**oupled **L**ogic) consommation importante, mais très rapide (communications rapides Gbit/s).

CMOS (**C**omplementary **M**etal **O**xide **S**emiconductor) (V_{CC}=0.8V à 18V, H=V_{CC}, L=0V) très basse consommation : technologie actuellement dominante.

Notons que :

- **V_{cc}** : tension d'alimentation du circuit ;
- **H** : niveau de tension haut équivalent à 1 logique ;
- **L** : niveau de tension bas équivalent à 0 logique.

4.7.1. Circuits logiques TTL

Présentation

Transistor-Transistor Logic ou TTL est une famille de circuits logiques utilisée en électronique inventée dans les années 1960. Cette famille est réalisée avec la technologie du transistor bipolaire et tend à disparaître du fait de sa consommation énergétique élevée (comparativement aux circuits CMOS).

➲ Les avantages de cette famille :

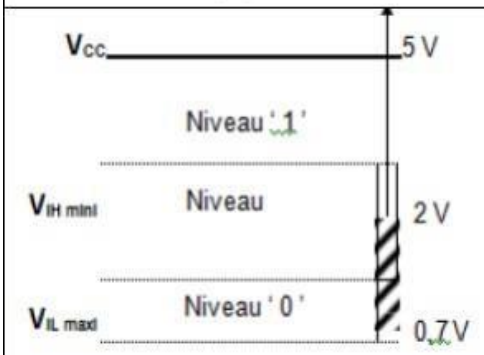
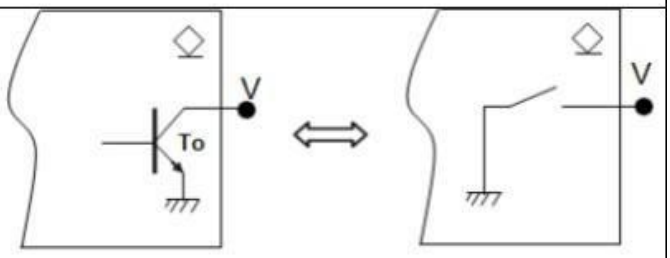
- Les entrées laissées en 'l'air' ont un état logique à 1 par défaut.
- Une bonne immunité au bruit.
- Un temps de propagation faible.

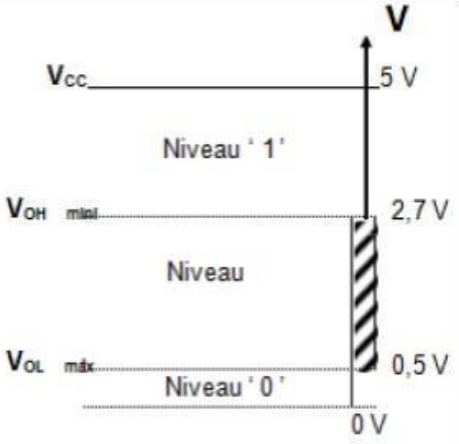
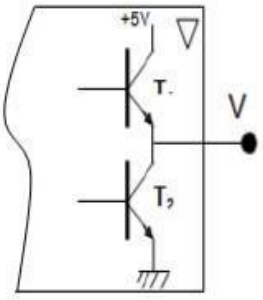
➲ Les inconvénients de cette famille :

- L'alimentation doit être précise à 5V +/- 5 % sinon on risque de détruire le circuit.

Du fait qu'elle est réalisée avec des transistors bipolaires elle consomme pas mal de courant comparer à la famille CMOS. (Car les transistors bipolaires sont commandés en courant).

Caractéristiques

Référence de boitier	**Caractéristiques de fonctionnement**
•TTL standard (n 'est plus utilisée) : 74 XX • TTL Low Power: 74 L XX • TTL Schottky (Rapide): 74 SXX • TTL Low Power Schottky:74 LSXX • TTL Advanced Schottky: 74 ASXX TTL • Advanced Low Power Schottky:74ALSXX	Gamme d'alimentation : 5 V +/- 5%. • Gamme de température : de 0 °C à + 70 °C. • Puissance dissipée : environ 2 mW par porte (série LS). Fréquence de fonctionnement : jusqu'à 3 MHz. • Sortance : jusqu'à 20 (série LS).(Nombre d'entrées que l'on peut relier à une sortie de porte)
Niveaux Logiques d'une porte logique TTL (LS) en entrée	**Sortie à collecteur ouvert (Open collector)**
	 On sort directement sur le collecteur du transistor de sortie. Obligation de connecter une résistance R de tirage au +5 V. La sortie est équivalente à un interrupteur.

Niveaux Logiques d'une porte logique TTL(LS) en sortie	**Sortie 3 états (3-state)**
	 EN T1 T2 Etat 1 Passant Bloqué Haut 1 Bloqué Passant Bas 0 Bloqué Bloqué Haute impédance (Sortie 'en l'air'). • Dans une porte classique, l'un des 2 transistors T1 ou T2 du totem pôle est conducteur. • ans une porte 3 états, il est possible de bloquer simultanément les 2 transistors T1 et T2 par l'entrée de validation EN (EN = 0).

Exemple : circuit TTL nommé SN74LS00.

SN : signifie que le constructeur est Texas Instruments.

74 : désigne les circuits intégrés grands publics qui supportent une température ambiante comprise entre 0 et 70 degré.

LS ou HCT : indiquent la sous famille du circuit TTL.

00 : les derniers chiffres indiquent la fonction logique réalisé par le composant (00=Porte NAND, 02=Porte NOR, 08=Porte AND …. etc.).

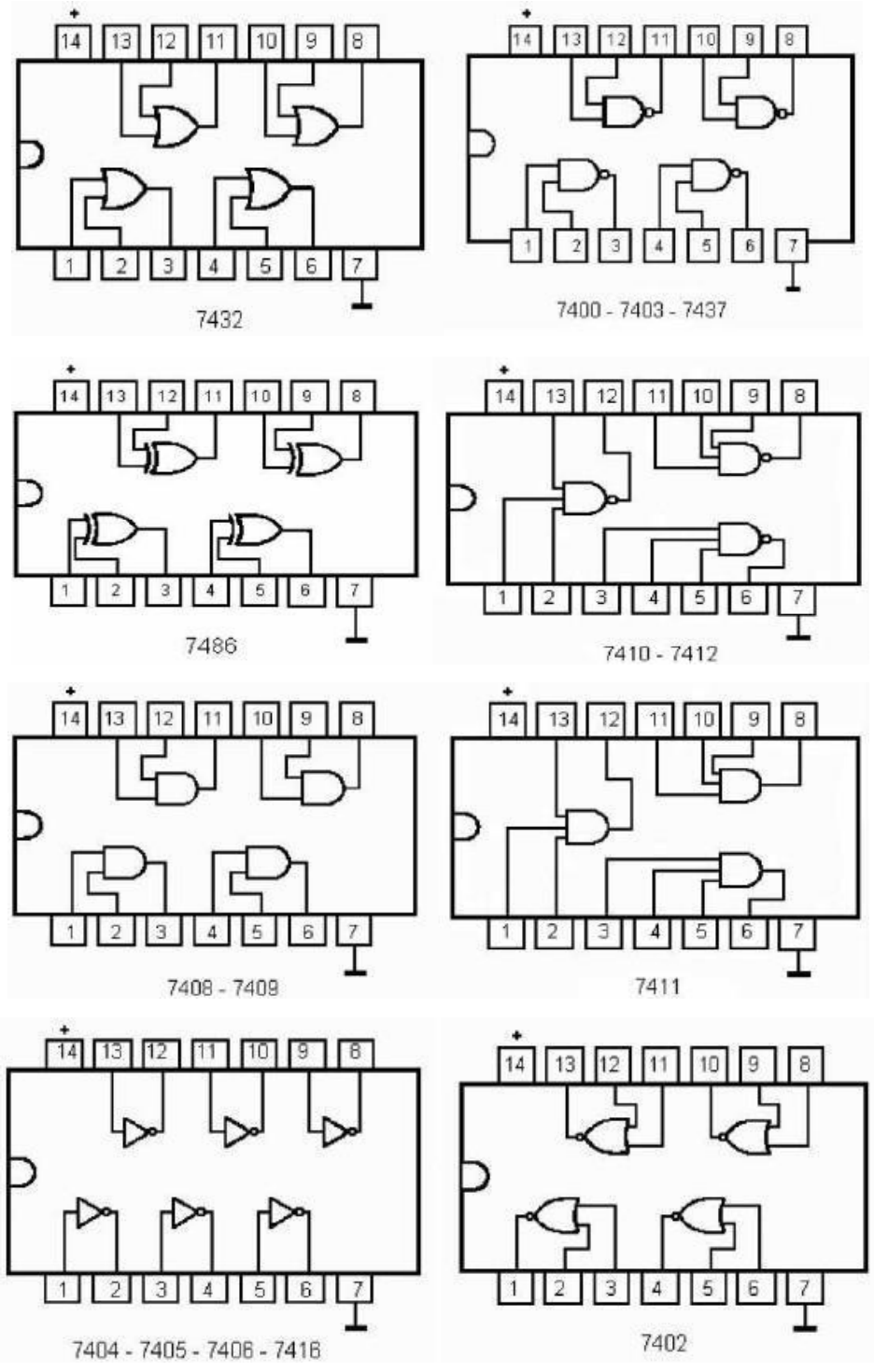


Figure 4.10. Exemples de circuits intégrés TTL.

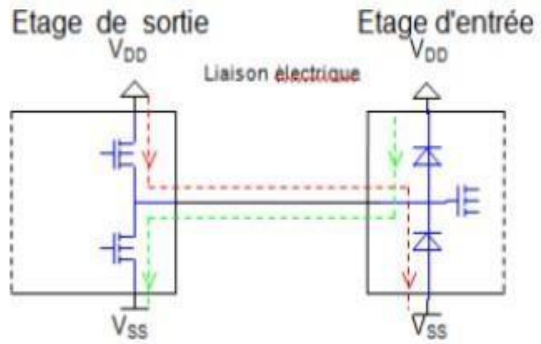
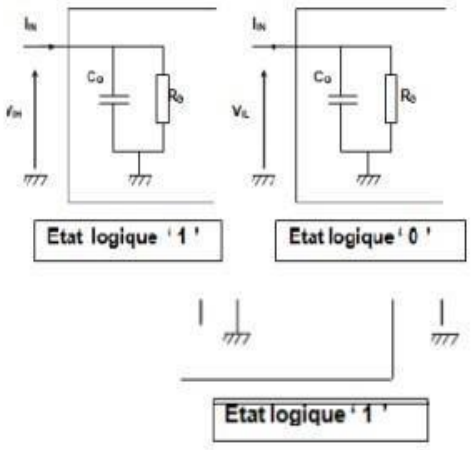
4.7.2. Circuits logiques CMOS

Présentation

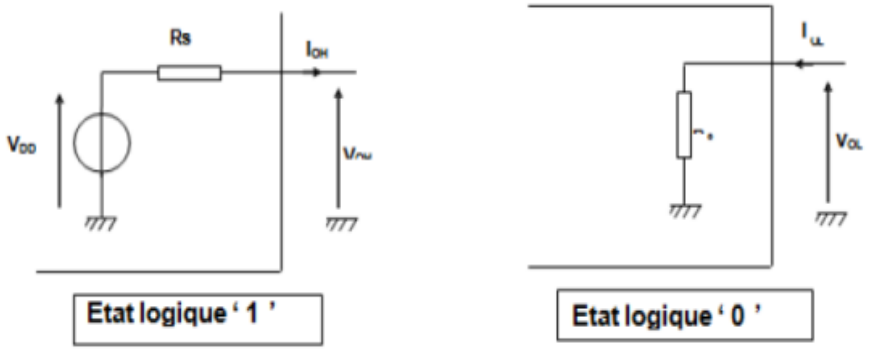
CMOS est l'abréviation de "Complementary Metal Oxide Semi-conductor". Le premier dispositif MOS est apparu en 1960. Les premiers circuits CMOS sur substrat SOI ont été mis sur le marché en 1999 [20]. Comme il existe, pour des transistors classiques, le type NPN et le Type PNP, les transistors MOS se déclinent en canal N et en canal P. les circuits dénommés C-MOS intègrent à la fois des canaux N et des canaux P [21]. Son développement a été rendu possible par les progrès réalisés par la technologie TTL. Cette famille est réalisée avec des transistors à effet de champs. Les avantages de cette famille :

- L'alimentation peut aller de 3V à 18V.
- Le courant d'entrée est nul, car elle est réalisée avec des transistors à effet de champs. (Les transistors à effet de champs sont commandés en tension).
- Une excellente immunité au bruit. Les inconvénients de cette famille.
- La vitesse de commutation est plus faible que pour la technologie TTL.

Caractéristiques

Référence de boitier	**Caractéristiques de fonctionnement**
✓ Série 4000 • 40 00 B (sorties bufférisées : amplifiées) • 40 00 UB (sorties non-bufférisées) ✓ Série 74 : • 74 C 00 (identique à la série 4000) • 74 HC 00 (High-speed CMOS : CMOS rapides)	• Gamme d'alimentation : de 3 V à 15 V. Gamme de température : de – 40 °C à + 85 °C. Puissance dissipée : environ 10 nW par porte. • Fréquence de fonctionnement : jusqu'à 12 MHz. • Sortance : jusqu'à 50 (série 4000B).(Nombre d'entrées que l'on peut relier à une sortie de porte) • Excellente immunité aux bruits.
Schéma des étages d'entrée et de sortie	**Modélisation de l'entrée d'une porte logique CMOS**
Schéma des étages d'entrée et de sortie connectés 	

Modélisation de la sortie d'une porte logique CMOS



Exemple : circuit CMOS nommé CD4011BE.

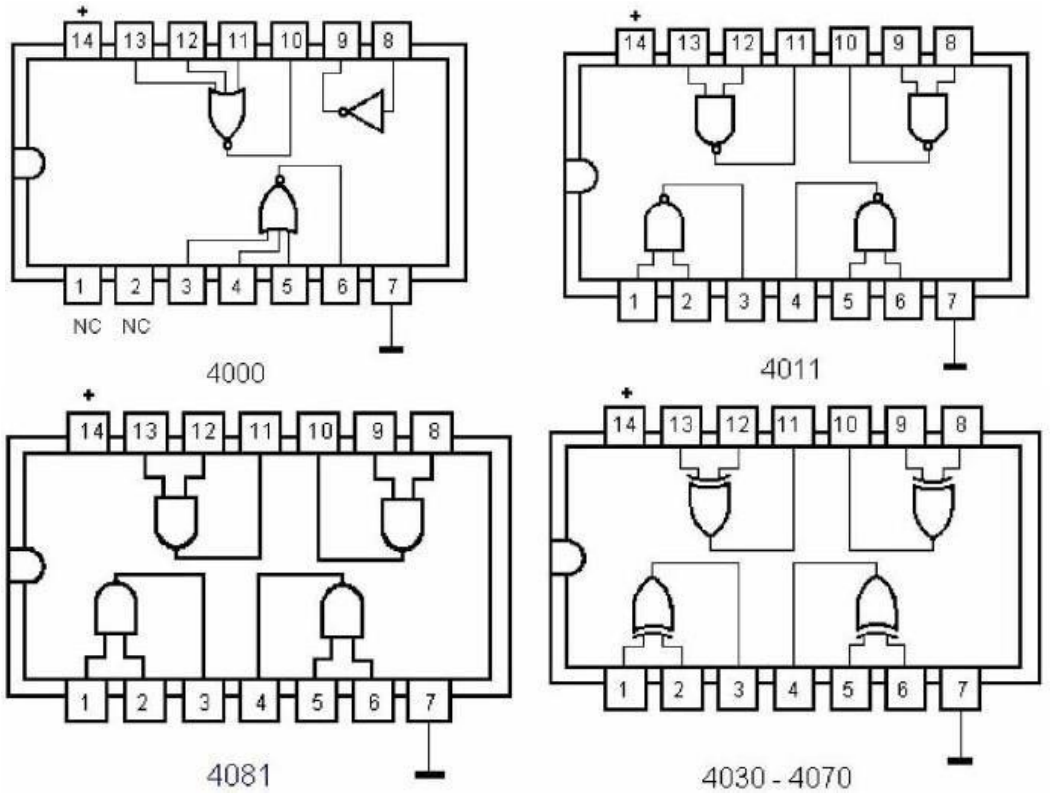
CD : préfixe utilisé par le constructeur Texas Instruments.

4011 : numéro du circuit. Il s'agit ici d'un quadruple porte NAND (NON-ET) à deux entrées chacune.

B : indique que la tension maximale est de 18V.

E : indication que le circuit est encapsulé dans un boitier DIP.

Codes circuit = préfixe fabricant + numéro du circuit + suffixe + code boîtier



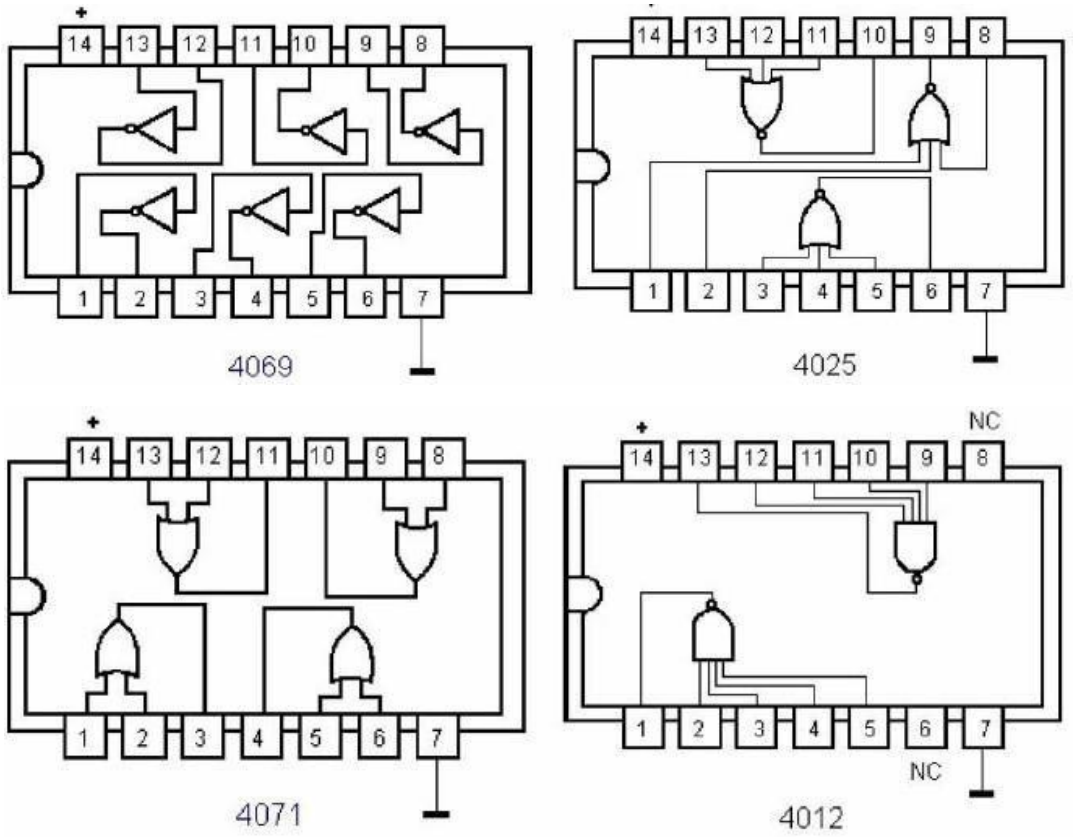


Figure 4.11. Exemples de circuits intégrés CMOS.