CHAPITRE 1

Algèbre de Boole et

Simplification des fonctions logiques

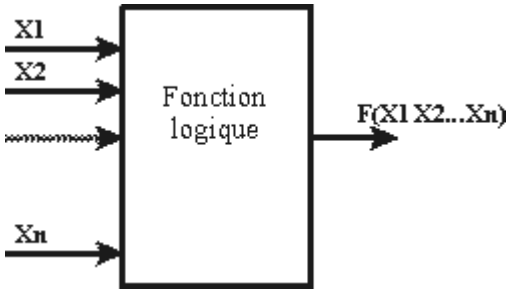
1. Rappels sur l'algèbre de BOOLE

1.1 Historique :

Georges BOOLE, philosophe et mathématicien anglais, publia en 1854 un essai sur les raisonnements logiques portant sur les propositions auxquelles les seules réponses possibles sont oui ou non. L'ensemble des opérations découlant de ces propositions forme une structure mathématique, donc une algèbre, appelée " algèbre de BOOLE ».

1.2 Définitions :

- **Variable logique** : grandeur, représentée par un identificateur (lettre ou nom) qui peut prendre les seules valeurs 0 ou 1.
- **Algèbre de BOOLE** : Ensemble de variables à 2 états, de valeur, ou état "1" (vrai) ou 0 (faux) et muni d'un petit nombre d'opérateurs fondamentaux : NON, ET, OU
- **Fonction logique de n variables binaires** : groupe de variables reliées par des opérateurs logiques (NON, ET, OU)



1.3 Notion de table de vérité

Une table de vérité permet de décrire le fonctionnement d'un système combinatoire, l'état de chaque entrée est représenté par sa valeur logique, de même pour les sorties. Il est possible de déterminer l'équation de fonctionnement en recherchant toutes les valeurs pour lesquelles la sortie=1.
L'équation de fonctionnement est égale à la somme logique de toutes les combinaisons pour lesquelles la sortie vaut 1.

Exemple 1 : Fonction entièrement définie

Fonction majorité F de trois variables logiques x,y,z. F(x,y,z)=1 si la majorité absolue des variables prend l'état '1' , '0' si non .

n	z	y	x	F(x,y,z)
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Exemple 2 : Fonction incomplètement définie
Certaines fonctions ne sont pas spécifiées pour toutes les combinaisons des variables ou certaines combinaisons de variables sont physiquement impossibles.
On notera par **'Ø'** l'état pris par ces fonctions pour les combinaisons pour lesquelles elles ne peuvent pas être spécifiées.
Fonction majorité F de quatre variables logiques w, x, y, z. F(w,x,y,z)=1 si la majorité absolue des variables prend l'état '1' , '0' si non .

état	z	y	x	w	F(w,x,y,z)
0	1	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	Ø
4	1	1	0	0	Ø
5	1	1	0	1	1
6	0	1	1	0	Ø
7	1	1	1	1	1

1.4 Représentations Canoniques
Une fonction logique F(x,y,z,....) peut s'écrire sous deux formes particulières appelées : Somme Canonique, Produit Canonique
- **F somme canonique** : F s'écrit sous la forme (somme de produits). On s'intéresse ici à l'ensemble des monômes où la fonction vaut '1'.
- **F produit canonique** : F s'écrit sous la forme (produit de sommes). On s'intéresse ici à l'ensemble des termes où la fonction vaut '0'.
La forme littérale la plus utilisée est la forme Somme canonique.

Exemple 1 : Fonction majorité F(x,y,z)
F(x,y,z) : somme canonique

n	z	y	x	F(x,y,z)
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Combinaisons où la fonction F(x,y,z) = 1 (z,y,x)	Monômes correspondants
0 1 1	$\overline{z}.y.x$
1 0 1	$z.\overline{y}.x$
1 1 0	$z.y.\overline{x}$
1 1 1	$z.y.x$

D'où :

$$F(x,y,z) = \overline{z}.y.x + z.\overline{y}.x + z.y.\overline{x} + z.y.x$$

Exemple 2 : Fonction majorité F(x,y,z)
F(x,y,z) : produit canonique

n	z	y	x	F(x,y,z)
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	**1**
4	1	0	0	0
5	1	0	1	**1**
6	1	1	0	**1**
7	1	1	1	**1**

Combinaisons où la fonction F(x,y,z) = 0 (z,y,x)	Monaux correspondants
0 0 0	$z+y+x$
0 0 1	$z+y+\overline{x}$
0 1 0	$z+\overline{y}+x$
1 0 0	$\overline{z}+y+x$

D'où :

$$F(x,y,z) = (z+y+x).(z+y+\overline{x}).(z+\overline{y}+x).(\overline{z}+y+x)$$

1.5 Fonctions logiques élémentaires :

Trois fonctions élémentaires suffisent pour définir une algèbre de Boole : la fonction complément ou NON, la fonction ET ou produit logique et la fonction OU ou addition logique.

1.5.1 La fonction complément ou NON

Table de vérité

a	F(a)= $\overline{a}$
0 1	1 0

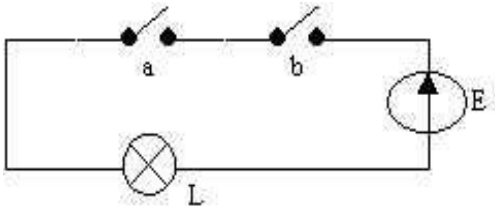
1.5.2 La fonction ET ou produit logique

Cette fonction prend la valeur 1 si toutes les variables sont simultanément à 1.

Table de vérité

a	b	F(a,b) = a . b
0	0	0
0	1	0
1	0	0
1	1	1

Interprétation électrique :



La lampe L n'est alimentée que si les interrupteurs a et b sont fermés simultanément.

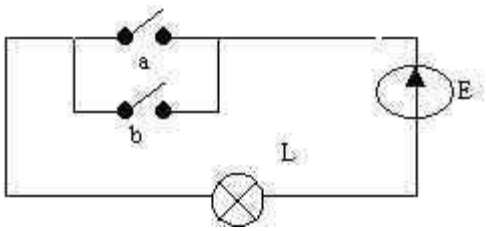
1.5.3 La fonction OU ou addition logique

Cette fonction prend la valeur 1 si au moins une des variables est égale à 1.

Table de vérité

a	b	F(a,b) = a + b
0	0	0
0	1	1
1	0	1
1	1	1

Interprétation électrique :



La lampe L est alimentée si au moins un des interrupteurs est fermé.

1.6 Les propriétés issues de la structure d'algèbre :

Propriétés : L'ensemble des variables binaires muni des opérations NON, OU et ET possède une structure d'algèbre. Les propriétés de cette algèbre se traduisent par le tableau ci–après :

	OU	ET
commutativité	$a+b = b+a$	$a.b = b.a$
associativité	$a+(b+c) = (a+b)+c$	$a.(b.c) = (a.b).c$
distributivité	$a+(b.c) = (a+b).(a+c)$	$a.(b+c) = (a.b)+(a.c)$
élément neutre	$a+0 = a$	$a.1 = a$
élément absorbant	$a+1 = 1$	$a.0 = 0$
complémentaire	$a+\overline{a} = 1$	$a.\overline{a} = 0$

Ces propriétés se démontrent aisément, en utilisant par exemple, les tables de vérité associées. D'autres propriétés intéressantes sont utilisées dans cette algèbre. Le tableau suivant en fait un descriptif.

	OU	ET
involution	$\overline{\overline{a}} = a$	
idempotence	$a+a = a$	$a.a = a$
absorption 1	$a+ab = a$	$a.(a+b)=a$
absorption 2	$a+\overline{a}b = a+b$	$a.(\overline{a}+b) = ab$
consensus	$ab+\overline{a}c+bc = ab+\overline{a}c$	$(a+b).(\overline{a}+c).(b+c) = (a+b).(\overline{a}+c)$
De Morgan	$\overline{a+b} = \overline{a}.\overline{b}$	$\overline{a.b} = \overline{a}+\overline{b}$

1.7 Opérateurs Combinatoires :

1.7.1 Opérateurs logiques simples :

Ce sont les opérateurs fabriqués à partir des fonctions logiques élémentaires.

a/ Opération NON (NOT) : négation ou complémentaire

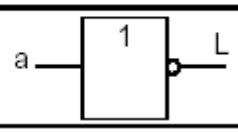
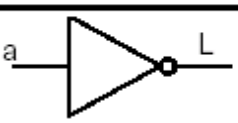
Opérateur	Fonction logique	Norme IEC	Norme américaine
NON	$L = \overline{a}$		

Table de vérité

a	L
0	1
1	0

b/ Opération ET (AND) : produit logique tel que *a*.*b* = 1 si et seulement si les deux variable a et b sont égales à 1.

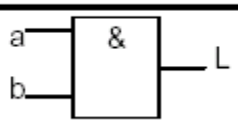
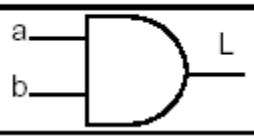
Opérateur	Fonction logique	Norme IEC	Norme américaine
ET	$L = a \cdot b$		

Table de vérité

a	b	L = a . b
0	0	0
0	1	0
1	0	0
1	1	1

La porte ET détecte le cas où toutes ses entrées sont à l'état haut (1).

c/ Opération OU (OR) : appelé somme logique tel que *a* + *b* = 1 si et seulement si l'une au moins des variables a ou b est égale à 1.

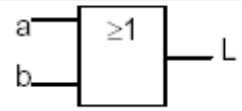
Opérateur	Fonction logique	Norme IEC	Norme américaine
OU	$L = a + b$		

Table de vérité

a	b	L = a + b
0	0	0
0	1	1
1	0	1
1	1	1

La porte OU détecte le cas où toutes ses entrées sont à l'état bas (0).

1.7.2 Opérateurs logiques complets (universels) :

a/ Opération NON-ET (NAND)

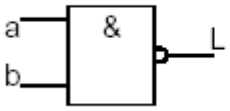
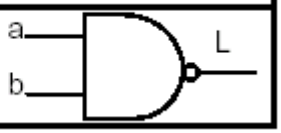
Opérateur	Fonction logique	Norme IEC	Norme américaine
NAND	$L = \overline{a \cdot b}$		

Table de vérité

a	*b*	$L = \overline{a.b}$
0	0	1
0	1	1
1	0	1
1	1	0

b/ Opération NON-OU (NOR)

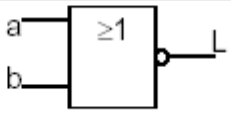
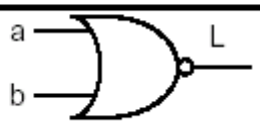
Opérateur	Fonction logique	Norme IEC	Norme américaine
NOR	$L = \overline{a + b}$		

Table de vérité

a	*b*	$s = \overline{a + b}$
0	0	1
0	1	0
1	0	0
1	1	0

Conclusion : les opérateur NON ET et NON OU sont universels dans le sens où leur association permet de réaliser les 3 fonctions logiques de base.

1.7.3 Opération OU EXCLUSIF (XOR)

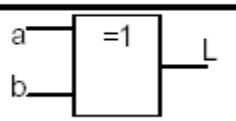
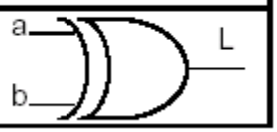
Opérateur	Fonction logique	Norme IEC	Norme américaine
OU exclusif	$L = a \oplus b = \overline{a}b + a\overline{b}$		

Table de vérité

a	b	$L = a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

1.7.4 Opération NON OU EXCLUSIF (XNOR)

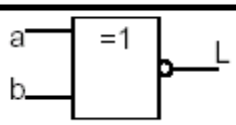
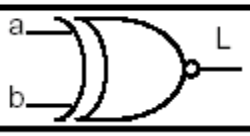
Opérateur	Fonction logique	Norme IEC	Norme américaine
NON OU exclusif IDENTITE	$L = \overline{a \oplus b} = \overline{a}\,\overline{b} + ab$		

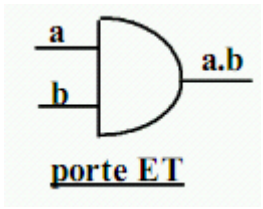
Table de vérité

a	b	$L = \overline{a \oplus b}$
0	0	1
0	1	0
1	0	0
1	1	1

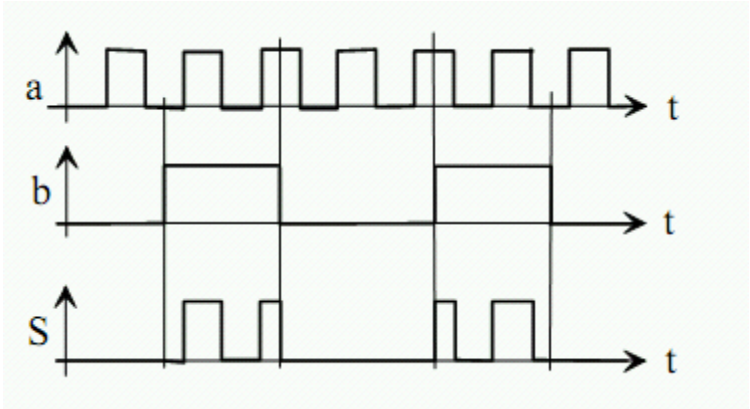
1.8 Détermination de chronogrammes.

Il est indispensable de savoir construire les chronogrammes à la sortie des portes logiques élémentaires, connaissant ceux sur les entrées.

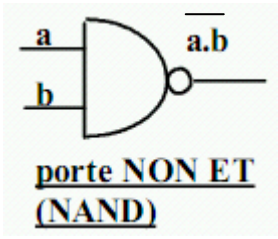
- **La porte logique ET (AND) :**

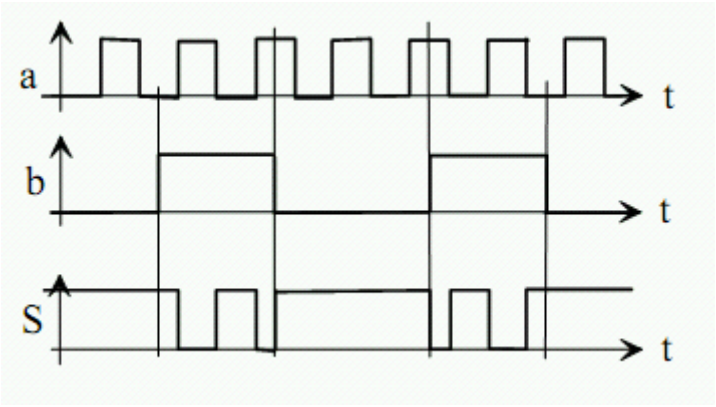


= S

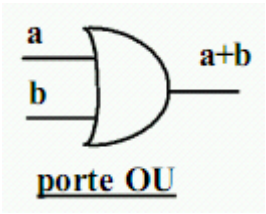


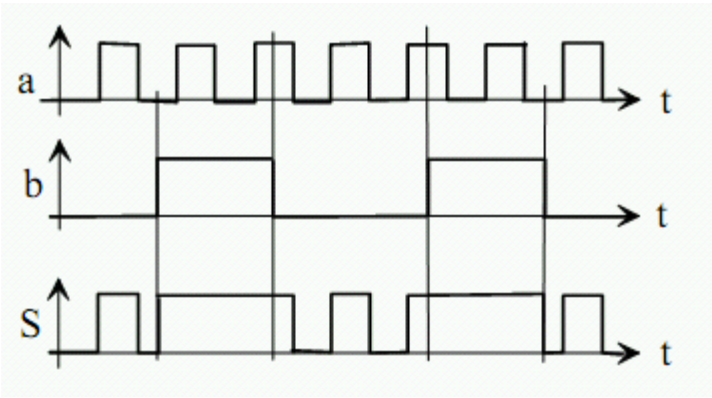
- **La porte logique NON ET (NAND) :**



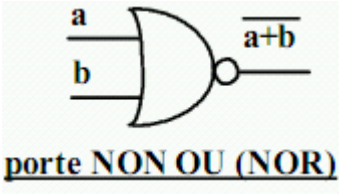


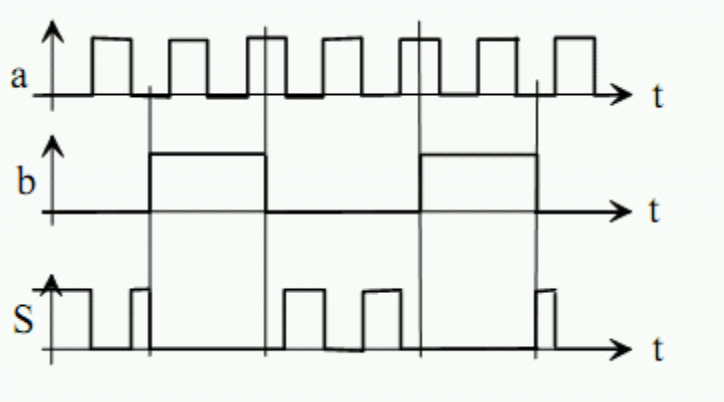
- **La porte logique OU (OR) :**





- **La porte logique NON OU (NOR) :**

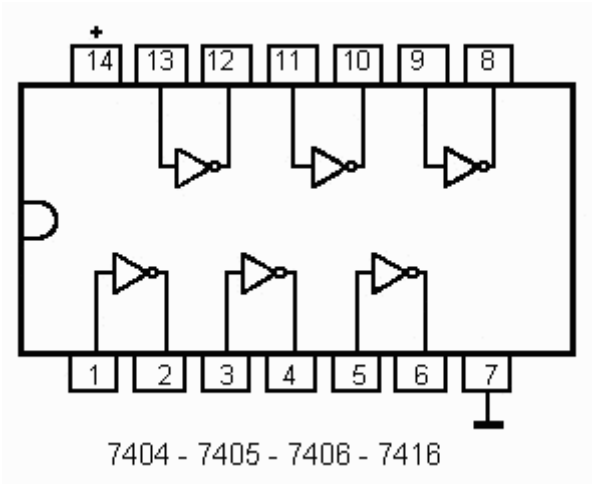




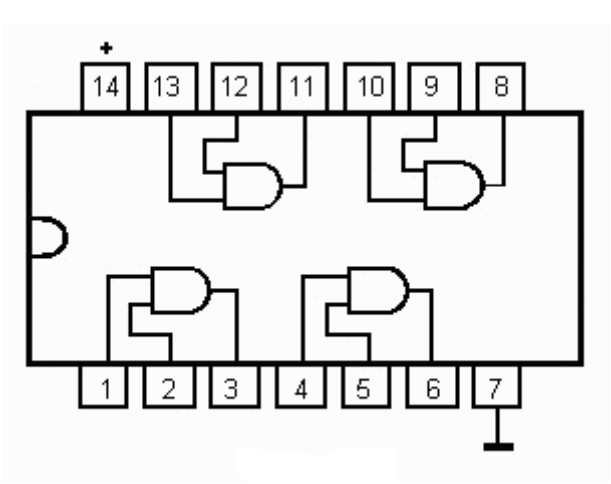
1.9 Transformation en NAND ou en NOR.

Les équations font souvent apparaître des fonctions OU, ET et des inverseurs. Il faut alors de nombreux boîtiers de circuits intégrés pour réaliser la fonction logique, sachant que les fonctions logiques sont regroupées par 2, 3, 4, ou 6 fonctions identiques dans un même boîtier (Ex: 6 inverseurs dans un même boîtier, ou 4 portes à 2 entrées, ou 3 portes à 3 entrées, ou 2 portes à 4 entrées).

On transforme alors les équations pour ne conserver que des fonctions de même type (NAND ou NOR).



7404 - 7405 - 7406 - 7416



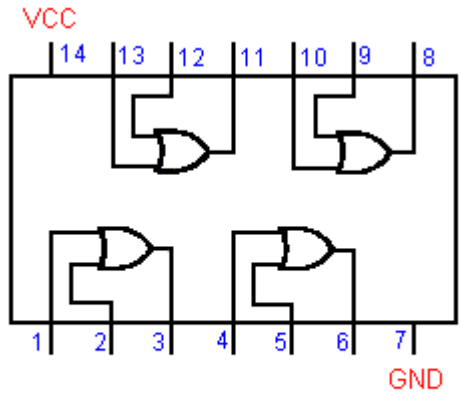


Fig. 5. - Schéma du circuit intégré MM 74C32

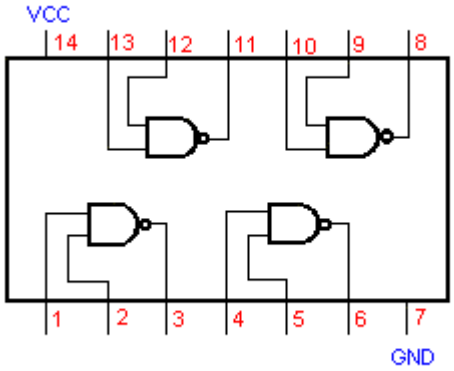


Fig. 26. - Schéma du circuit intégré MM 74C00.

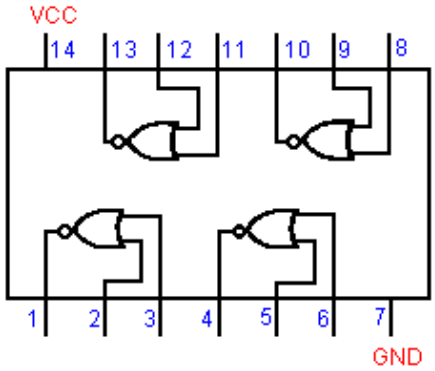


Fig. 8. - Schéma du circuit intégré MM 74C02.

Remarque: (On cherche plutôt à minimiser le nombre de boîtiers).
Pour la transformation en NAND, il faut éliminer toutes les fonctions OU par une double complémentation, puis utilisation des règles de Morgan.

Exemple : $S = \bar{b}.c + a.b$

$$S = \overline{\overline{\bar{b}.c + ab}} = \overline{\overline{\bar{b}.c} \,.\, \overline{a.b}}$$

Il faudra donc 4 portes NAND (1 par barre) ce qui correspond à 1 seul boîtier.

La transformation en NOR est identique en supprimant toutes les fonctions ET à l'aide des règles de Morgan.

Exemple : $S = \bar{b}.c + a.b$

$$S = \overline{\overline{\bar{b}.c}} + \overline{\overline{a.b}} = \left(\overline{\overline{\bar{b} + \bar{c}}}\right) + \left(\overline{\overline{a} + \overline{b}}\right) = \left(\overline{b + \bar{c}}\right) + \left(\overline{\bar{a} + b}\right)$$

$$S = \overline{\left(\overline{b + \bar{c}}\right) + \left(\overline{\bar{a} + b}\right)}$$

2. Logique combinatoire

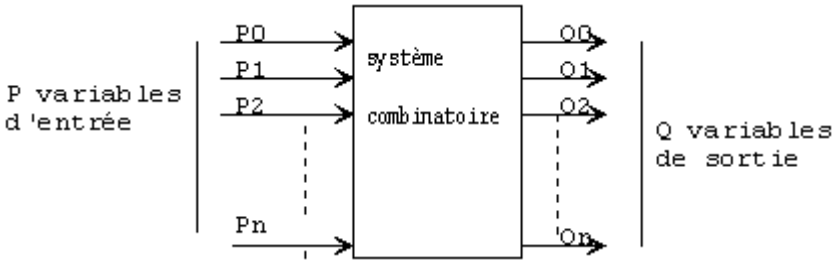
2.1. Définition

On appelle circuit logique ou porte logique un dispositif à plusieurs entrées et plusieurs sorties qui délivre un signal de sortie si et seulement si une certaine combinaison de signaux est appliquée à l'entrée.

Un circuit logique combinatoire est un circuit dont l'état de sa sortie dépend uniquement de la combinaison des signaux à l'entrée et pas de l'état antérieur de sa sortie.

Les grandeurs binaires peuvent être des grandeurs physiques : tension, courant, pression d'un liquide, d'un gaz, force, température,…..

A tout instant, on peut représenter logiquement un système combinatoire en faisant une liste des entrées et des sorties : la table de vérité.



2.2 Méthodes de simplification des fonctions logiques

Une fonction peut être représentée par plusieurs expressions algébriques différentes mais équivalentes. Nous allons appliquer deux méthodes aboutissant à « minimiser » les expressions des fonctions logiques.

2.2.1 – La méthode algébrique

On procède par des regroupements qui permettent (en utilisant les propriétés des opérateurs logiques) d'éliminer des variables :
Elle repose sur une application astucieuse des propriétés algébriques des variables

Exemple 1

F(a,b,c,d) = $\overline{a}\,\overline{b}\,\overline{c}d + \overline{a}b\overline{c}d + \overline{a}bcd + a\overline{b}\,\overline{c}d + a\overline{b}cd + ab\overline{c}\,\overline{d} + abc\overline{d} + abcd$

1 2 3 4 5 6 7 8

Simplifions cette expression :

* (1+2) donne $\overline{a}\,\overline{c}d$
* (3+8) donne bcd
* (1+4) donne $\overline{b}\,\overline{c}d$ d'où le résultat : $f(a,b,c,d) = \overline{a}\,\overline{c}d + bcd + \overline{b}\,\overline{c}d + acd + ab\overline{d}$
* (5+8) donne acd
* (6+7) donne $ab\overline{d}$

On pourrait, en simplifiant d'une manière différente, montrer que :

$$f(a,b,c,d) = \overline{a}\,\overline{c}d + bcd + a\overline{b}\,d + ab\overline{d}$$

ou

$$f(a,b,c,d) = \overline{b}\,\overline{c}d + \overline{a}bd + ab\overline{d} + acd$$

Exemple 2

$$F(x,y,z) = \overline{z}.y.x + z.\overline{y}.x + z.y.\overline{x} + z.y.x$$

$$F(x,y,z) = \overline{z}.y.x + z.\overline{y}.x + z.y.\overline{x} + z.y.x + z.y.x + z.y.x$$

$$F(x,y,z) = (\overline{z}.y.x + z.y.x) + (z.\overline{y}.x + z.y.x) + (z.y.\overline{x} + z.y.x)$$

$$F(x,y,z) = (\overline{z} + z).y.x + (\overline{y} + y).z.x + (\overline{x} + x).z.y$$

$$F(x,y,z) = (1).y.x + (1).z.x + (1).z.y$$

D'où l'expression simplifiée : $F(x,y,z) = y.x + z.x + z.y$

Exemple 3

$$f = \overline{x}\,\overline{y}\,\overline{z}\,\overline{t} + \overline{x}\,\overline{y}z\overline{t} + \overline{x}y\overline{z}\,t + x\overline{y}\,\overline{z}\,\overline{t} + x\overline{y}\,\overline{z}\,t + x\overline{y}z\overline{t} + x\overline{y}zt + xy\overline{z}\,t$$
$$f = \overline{x}\,\overline{y}\,\overline{t}\,(\overline{z} + z) + \overline{x}y\overline{z}\,t + x\overline{y}\,\overline{z}\,(\overline{t} + t) + x\overline{y}z(\overline{t} + t) + xy\overline{z}\,t$$
$$f = \overline{x}\,\overline{y}\,\overline{t} + \overline{x}y\overline{z}\,t + x\overline{y}\,\overline{z} + x\overline{y}z + xy\overline{z}\,t$$
$$f = \overline{x}\,\overline{y}\,\overline{t} + x\overline{y}(\overline{z} + z) + (\overline{x} + x)y\overline{z}\,t$$
$$f = (\overline{x}\,\overline{t} + x)\overline{y} + y\overline{z}\,t$$
$$f = (\overline{t} + x)\overline{y} + y\overline{z}\,t$$
$$f(x, y, z, t) = \overline{y}\,\overline{t} + x\overline{y} + y\overline{z}\,t.$$

Remarque : méthode difficile à mettre en œuvre. Il existe d'autres méthodes algébriques qui sont plus performantes mais dont la complexité est grande pour les faire manuellement. Elles sont généralement implantées sur des ordinateurs.

2.2.2 - La méthode de Karnaugh

a/ Tableau De Karnaugh

Les tableaux de KARNAUGH sont une représentation particulière de la table de vérité. Les cases représentant l'état des variables d'entrée doivent être adjacentes (Une seule variable change d'état).

Les tableaux de KARNAUGH permettent la simplification des équations logiques. Ils comportent 2^n cases, n étant le nombres de variables d'entrée. (Ex : 4 variables donnent 16 cases).

Chaque case correspond à une combinaison possible des variables d'entrée;

Chaque combinaison exprimée dans l'équation sera représentée par un « 1 » dans la case correspondante.

Il est ensuite possible de regrouper les cases par 2, 4, 8, 2^n afin d'éliminer les variables qui change d'état dans le regroupement :

- un regroupement de 2 cases élimine 1 variable;
- un regroupement de 2^x cases élimine x variables.

Les plus grands regroupements donnant les équations les plus simples .

b/ Simplification par les tableaux de KARNAUGH.

- Simplification d'une fonction complètement définie

Pour une fonction exprimée en somme canonique, la méthode consiste à rechercher les groupements de 2 cases, 4 cases, 8 cases adjacentes contenant des '1' de façon à éliminer 1, 2, 3 variables dans l'écriture des monômes.

Exemple 1 : Expression simplifiée de la fonction majorité F(x,y,z)

z\yx	00	01	11	10
0	0	0	1	0
1	0	1	1	1

C

B

A

Groupement A : monôme associé = z.x
Groupement B : monôme associé = z.y
Groupement C : monôme associé = y.x
D'où :

$$F(x,y,z) = z.x + z.y + y.x$$

Exemple 2 : soit la table de vérité suivante :

A	B	C	D	E
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

La résolution par Karnaugh donne

CD \ AB	00	01	11	10
00	1	0	0	0
01	1	1	1	0
11	0	1	1	1
10	0	0	0	0

On obtient pour l'expression de la sortie : E = B.D + A.C.D + $\overline{A} \cdot \overline{B} \cdot \overline{C}$

Exemple 3 :

xy \ zt	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	0	0	1
10	1	0	0	1

donc $f(x, y, z, t) = y\overline{z}\,t + x\overline{y} + \overline{y}\,\overline{t}$.

Compléter les tableaux vides

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

S = a

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

S = b

a \ bc	00	01	11	10
0				
1				

S = c

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

S = a + b

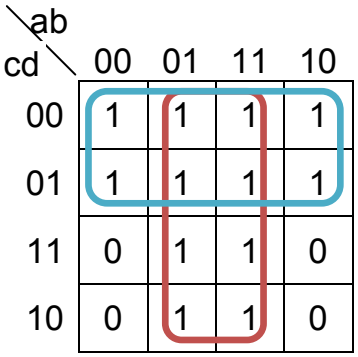
a \ bc	00	01	11	10
0				
1				

S= a + b.c

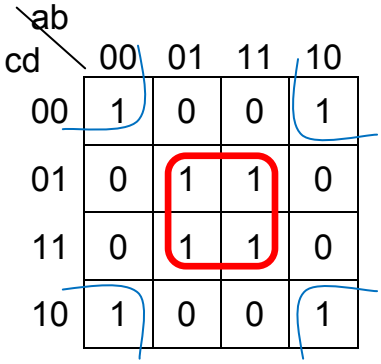
a \ bc	00	01	11	10
0	0	1	1	1
1	0	1	1	0

S= c + /a.b

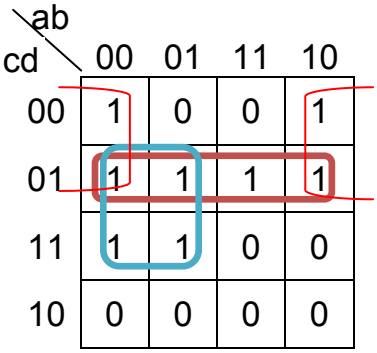
Sortir les équations simplifiées en utilisant les tableaux de KARNAUGH.



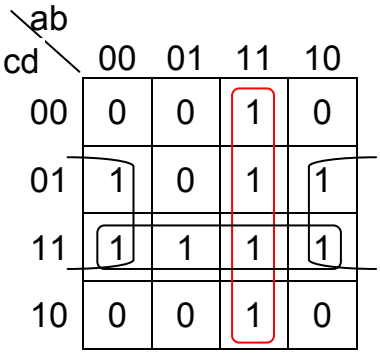
M = b + $\bar{c}$



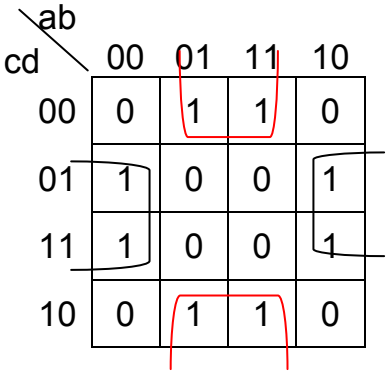
N = bd + $\bar{b}\bar{d}$



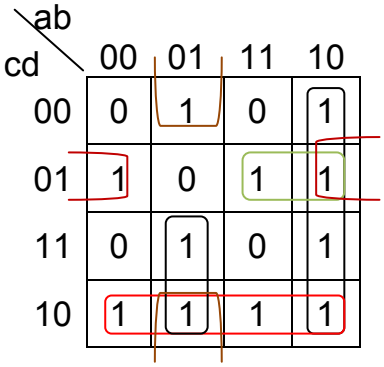
P = $\bar{c}d + \bar{a}d + \bar{b}\bar{c}$



R = $ab + cd + \bar{b}d$



S = $b\bar{d} + \bar{b}d$

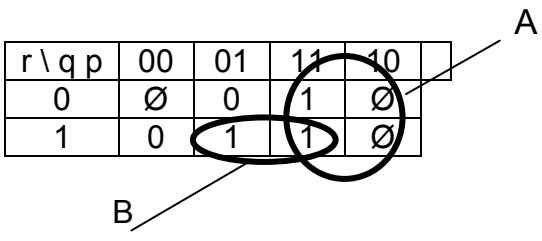


T = $\bar{b}\bar{c}d + a\bar{b} + c\bar{d} + \bar{a}bc + \bar{a}b\bar{d} + a\bar{c}d$

- Simplification d'une fonction incomplètement définie

Afin de faciliter des regroupements de taille importante, on utilisera les cases contenant les 'Ø' soit comme des '1', soit comme des '0'.

Exemple : Simplification de la fonction G(p,q,r) définie par le tableau de Karnaugh suivant



Groupement A : monôme associé = q
Groupement B : monôme associé = r.p
D'où :

G(p,q,r)= q + r.p

CHAPITRE 2

Systèmes de numération et codage de lôinformation

1. Représentation des nombres

Le nombre de symboles utilisés caractérise le numéro de la base.
Ex.:
Åen base 10, nous avons les 10 symboles (0, 1,..,9)
Åen base 2, nous avons les 2 symboles (0, 1)
Åen base 3, nous avons les 3 symboles (0, 1, 2)
Åen base 8, nous avons les 8 symboles (0, 1,..,7)
Åen base 16, nous avons besoin de 16 symboles,
nous utiliserons les 10 chiffres plus les lettres de A à F, soit
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Le poids d'un chiffre dépend de sa position dans le nombre. Nous parlons de numération de position, soit: Un nombre dans une base "b" entière positive s'écrit:

$$N_b = a_n a_{n-1} \ldots a_1 a_0, a_{-1} \ldots a_{-m} \qquad (1)$$

ce qui correspond aux opérations:

$$N_B = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + \ldots + a_{-m} \cdot b^{-m} \qquad (2)$$

L'indice de *b* indique la base dans laquelle le nombre est calculé.
N.B.: La formule (2) donne N dans la base dans laquelle on effectue les opérations (ici la base B).

2. Conversion Binaire - Décimal

Le système de numération binaire est un système de numération de position où le poids de chaque bit est un multiple de puissance de 2 (base).

$$\begin{array}{ccccccccccc} 1 & & 1 & & 0 & & 1 & & 1 & & \text{(binaire)} \\ 1\cdot 2^4 & + & 1\cdot 2^3 & + & 0\cdot 2^2 & + & 1\cdot 2^1 & + & 1\cdot 2^0 & & \\ 16 & + & 8 & + & 0 & + & 2 & + & 1 & = 27_{10} & \text{(décimal)} \end{array}$$

Voyons un autre exemple pour un nombre ayant un plus grand nombre de bits.

$$\begin{array}{cccccccccc} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & & \text{binaire} \\ 1\cdot 2^7 + & 0\cdot 2^6 + & 1\cdot 2^5 + & 1\cdot 2^4 + & 0\cdot 2^3 + & 1\cdot 2^2 + & 0\cdot 2^1 + & 1\cdot 2^0 & & \\ 128 + & 0 + & 32 + & 16 + & 0 + & 4 + & 0 + & 1 & = 181_{10} & \text{décimal} \end{array}$$

3. Conversion Décimal - Binaire

Il existe deux façons de convertir un nombre décimal en son équivalent binaire. Une méthode qui convient bien aux petits nombres est une démarche qui est basée sur la numération de position en binaire. Le nombre décimal est simplement exprimé comme une somme de puissances de 2, puis on inscrit des 1 et des 0 vis-à-vis des positions binaires appropriées. Voici un exemple :

$$45_{10} = 32 + 8 + 4 + 1 = 2^5 + 0 + 2^3 + 2^2 + 0 + 2^0 = 1\,0\,1\,1\,0\,1_2$$

3.1 Conversion de la partie entière

L'autre méthode convient mieux aux grands nombres décimaux; il s'agit de répéter la division par 2. La partie entière d'un nombre peut s'exprimer comme suit :

$$N_B = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots + a_1 \cdot b^1 + a_0 \cdot b^0$$

Si nous divisons N_B par la base b, nous obtenons l'expression suivante

$$\frac{N_B}{b} = (a_n \cdot b^{n-1} + a_{n-1} \cdot b^{n-2} + \ldots + a_1 \cdot b^0) \ldots et \ldots (a_0)$$

a_0 apparaît comme le reste de la division de N (entier) par b; a1 est le reste de la division du quotient par b; a_2 est le reste de la division du nouveau quotient par b. On opère donc par divisions successives par b.

Exemple :

25/2 =	12 reste	1	Poids faible (LSB)
12/2 =	6 reste	0	
6/2 =	3 reste	0	
3/2 =	1 reste	1	
1/2 =	0 reste	1	Poids fort (MSB)

$25_{10} = 1\,1\,0\,0\,1_2$

3.2 Conversion de la partie fractionnaire

La conversion de la partie fractionnaire s'obtient par l'opérateur inverse, soit la multiplication. La partie fractionnaire d'un nombre peut s'exprimer comme suit :

$$N_B = a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} + \ldots + a_{-m+1} \cdot b^{-m+1} + a_{-m} \cdot b^{-m}$$

Si nous multiplions N_B par la base b, nous obtenons l'expression suivante :

$$b \cdot N_B = a_{-1} \ldots (a_{-2} \cdot b^{-1} + \ldots + a_{-m+1} \cdot b^{-m+2} + a_{-m} \cdot b^{-m+1})$$

a_{-1} apparaît comme la partie entière de la multiplication N (fractionnaire) par b; a_{-2} est la partie entière de la multiplication par b du reste; a_{-3} est la partie entière de la multiplication par nouveau reste par b. On opère donc par multiplication successives par b.

0,375 x 2 = 0,75	partie entière = 0 reste = 0,75	Poids fort (MSB)
0,75 x 2 = 1,5	partie entière = 1 reste = 0,5	
0,5 x 2 = 1,0	partie entière = 1 reste = 0	Poids faible (LSB)

N_{10} = 0,375 correspond à N_2 = 0,011

4. Système de numération Octal

Le système de numération octal a comme base huit, ce qui signifie qu'il comprend huit symboles possibles, soit 0, 1, 2, 3, 4, 5, 6 et 7. Ainsi, chaque chiffre dans un nombre octal a une valeur comprise entre 0 et 7. Voici les poids de chacune des positions d'un nombre octal.

....	8^3	8^2	8^1	8^0	.	8^{-1}	8^{-2}	8^{-3}	....

4.1 Conversion octal décimal

On convertit un nombre octal en son équivalent décimal en multipliant chaque chiffre octal par son poids positionnel. Voici un exemple:

$$372_8 = 3\cdot(8^2) + 7\cdot(8^1) + 2\cdot(8^0)$$
$$= 3\cdot64 + 7\cdot8 + 2\cdot1$$
$$= 250_{10}$$

4.2 Conversion décimal octal

Il est possible de convertir un nombre décimal entier en son équivalent octal en employant la méthode de la répétition de divisions, la même qu'on a utilisée pour la conversion décimal binaire, mais cette fois-ci en divisant par 8 plutôt que par 2.

266/8 = 33 reste 2
33/8 = 4 reste 1
4/8 = 0 reste 4

$266_{10} = 412_8$

Notez que le premier reste devient le chiffre de poids le plus faible du nombre octal et que le dernier reste devient le chiffre de poids le plus fort.

4.3 Conversion octal binaire

Le principal avantage du système de numération octal réside dans la facilité avec laquelle il est possible de passer d'un nombre octal à un nombre binaire. Cette conversion s'effectue en transformant chaque chiffre du nombre octal en son équivalent binaire de trois chiffres.

Chiffre octal	0	1	2	3	4	5	6	7
Équivalent binaire	000	001	010	011	100	101	110	111

Au moyen de ce tableau, tout nombre octal est converti en binaire par la transformation de chacun des chiffres. Par exemple, la conversion de 472_8 va comme suit:

4 7 2
100 111 010

Donc le nombre octal 472_8 est équivalent au nombre binaire $(100111010)_2$.

4.4 Conversion binaire octal

La conversion d'un nombre binaire en un nombre octal est tout simplement l'inverse de la marche à suivre précédente. Il suffit de faire avec le nombre binaire des groupes de trois bits en partant du chiffre de poids le plus faible, puis de convertir ces triplets en leur équivalent octal. A titre d'illustration, convertissons 100111010_2 en octal.

100 111 010
4 7 2_8

Parfois, il arrivera que le nombre binaire ne forme pas un nombre juste de groupes de trois. Dans ce cas, on pourra ajouter un ou deux zéros à gauche du bit de poids le plus fort pour former le dernier triplet (si on lit de droite à gauche). Voici une illustration de ceci avec le nombre binaire 11010110.

011 010 110
3 2 6_8

5. Système de numération Hexadécimal

Le système hexadécimal a comme base 16, ce qui implique 16 symboles de chiffres possibles, qui, dans ce cas, sont les dix chiffres 0 à 9 plus les lettres majuscules A, B, C, D, E et F. Le tableau expose les rapports entre les systèmes hexadécimal, décimal et binaire. Remarquez que chaque chiffre hexadécimal a comme équivalent binaire un groupe de quatre bits.

Il ne faut surtout pas oublier que les chiffres hexadécimaux A à F correspondent aux valeurs décimales 10 à 15.

Hexadécimal	Décimal	Binaire
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

La représentation hexadécimale est principalement utilisée pour représenter un nombre binaire sous forme plus compact. Un nombre en hexadécimal comprend 4 fois moins de chiffres !

5.1 Conversion hexadécimal décimal

Un nombre hexadécimal peut être converti en son équivalent décimal en exploitant le fait qu'à chaque position d'un chiffre hexadécimal est attribué un poids; dans ce cas-ci le nombre 16 élevé à une certaine puissance. Le chiffre de poids le plus faible a un poids de $16^0 = 1$, le chiffre immédiatement à gauche a un poids de $16^1 = 16$, l'autre chiffre immédiatement à gauche, un poids de $16^2 = 256$, et ainsi de suite.

$$\begin{aligned} 356_{16} &= 3\cdot16^2 + 5\cdot16^1 + 6\cdot16^0 \\ &= 768 + 80 + 6 \\ &= 854_{10}. \end{aligned}$$

$$\begin{aligned} 2AF_{16} &= 2\cdot16^2 + 10\cdot16^1 + 15\cdot16^0 \\ &= 512 + 160 + 15 \end{aligned}$$

$= 687_{10}$

5.2 Conversion décimal hexadécimal

Vous vous rappelez peut-être que pour la conversion décimale binaire nous avons eu recours à la répétition de divisions par 2, que pour la conversion décimale octale, à la répétition de division par 8. Donc, pour convertir un nombre décimal en un nombre hexadécimal, il faut procéder de la même façon, mais cette fois en divisant par 16. Les exemples qui suivent illustrent cette technique. Remarquez comment les restes des divisions deviennent les chiffres du nombre hexadécimal; de plus, voyez, comment les restes supérieurs à 9 sont exprimés au moyen des lettres A à F. Exemple, conversion de 42310 en hexadécimal:

423/16 = 26 reste 7
26/16 = 1 reste 10
1/16 = 0 reste 1
$423_{10} = 1A7_{16}$

5.3 Conversion hexadécimal binaire

Comme le système de numération octal, le système de numération hexadécimal se veut une façon abrégée de représenter les nombres binaires. La conversion d'un nombre hexadécimal en un nombre binaire ne pose vraiment pas de difficulté, puisque chaque chiffre hexadécimal est remplacé par son équivalent binaire de 4 bits (tableau 2 2). Voici un exemple avec 9F216.

$9F2_{16}$ = 9 F 2
1001 1111 0010
= $(100111110010)_2$

5.4 Conversion binaire hexadécimal

Cette conversion est tout simplement l'inverse de la précédente. Le nombre binaire est divisé en groupes de quatre bits, puis on substitue à chaque groupe son chiffre Hexadécimal équivalent. Au besoin, on ajoute des zéros à gauche pour obtenir un dernier groupe de 4 bits.

1110100110_2 = 0011 1010 0110
3 A 6
$= 3A6_{16}$

5.5 Comptage hexadécimal

Lorsque l'on compte selon le système de numération hexadécimal, la valeur dans une position du nombre croît par pas de 1 depuis 0 jusqu'à F. Quand le chiffre dans une position est F, le chiffre suivant dans cette position est 0 et le chiffre immédiatement à gauche est augmenté de 1. C'est ce qu'on voit dans les suites de nombres hexadécimaux suivantes:

a. 38, 39, 3A, 3B, 3C, 3D, 3E, 3F, 40, 41, 42
b. 6F8, 6F9, 6FA, 6FB, 6FC, 6FD, 6FE, 6FF, 700,

5.6 Utilité du système hexadécimal

La facilité avec laquelle se font les conversions entre les systèmes binaire et hexadécimal explique pourquoi le système hexadécimal est devenu une façon abrégée d'exprimer de grands nombres binaires. Dans un ordinateur, il n'est pas rare de retrouver des nombres binaires ayant jusqu'à 64 bits de longueur. Ces nombres binaires, comme nous le verrons, ne sont pas toujours des valeurs numériques, mais peuvent correspondre à un certain code représentant des renseignements non numériques. Dans un ordinateur, un nombre binaire peut être: 1) un vrai nombre; 2) un nombre correspondant à un emplacement (adresse) en mémoire; 3) un code d'instruction; 4) un code correspondant à un caractère alphabétique ou non numérique; ou 5) un groupe de bits indiquant la situation dans laquelle se trouvent des dispositifs internes et externes de l'ordinateur.
Quand on doit travailler avec beaucoup de nombres binaires très longs, il est plus commode et plus rapide d'écrire ces nombres en hexadécimal plutôt qu'en binaire.

6. Code BCD, soit Binary Coded Decimal

Le BCD s'appelle en français code Décimal Codé Binaire (DCB). Si on représente chaque chiffre d'un nombre décimal par son équivalent binaire, on obtient le code dit décimal codé binaire . Comme le plus élevé des chiffres décimaux est 9, il faut donc 4 bits pour coder les chiffres.
Illustrons le code BCD en prenant le nombre décimal 874 et en changeant chaque chiffre pour son équivalent binaire; cela donne:

8	7	4	décimal
1000	0111	0100	BCD

De nouveau, on voit que chaque chiffre a été converti en son équivalent binaire pur. Notez qu'on fait toujours correspondre 4 bits à chaque chiffre.
Le code BCD établit donc une correspondance entre chaque chiffre d'un nombre décimal et un nombre binaire de 4 bits. Évidemment, seuls les groupes binaires 0000 à 1001 sont utilisés. Le code BCD ne fait pas usage des groupes 1010, 1011, 1100, 1101, 1110 et 1111.

6.1 Comparaison entre code BCD et nombre binaire

Prenons le nombre 137_{10} et trouvons son nombre binaire pur puis son équivalent BCD:

$137_{10} = 10001001_2$ (Binaire)
$137_{10} = 0001\ 0011\ 0111$ (BCD)

donc un nombre BCD n'est pas un nombre binaire pur.

6.2 Conversion BCD binaire

Une conversion est nécessaire pour convertir un nombre exprimé en BCD en binaire. La seule possibilité est de passer par la valeur décimale. Voici la démarche à suivre :
N_{BCD} => l'exprimé en décimal => convertir en binaire.

6.3 Conversion binaire BCD

La conversion d'une valeur binaire en BCD demande de passer aussi par la valeur décimale. Voici la démarche à suivre :

N_2 => convertir en décimal => l'exprimé en BCD

7. Récapitulatif de différents codes

Nous donnerons un tableau des principaux codes. Il faut toutefois mentionner le code GRAY ou binaire réfléchi. Ce code présente l'avantage qu'il n'y a qu'un seul bit qui change à la fois.

Les codes Excédent 3 et AIKEN ne sont pratiquement plus utilisés.

Décimal	binaire	octal	hexadécimal	Gray ou BR	Excédent 3	AIKEN
00	0000	00	0	0000	0011	0000
01	0001	01	1	0001	0100	0001
02	0010	02	2	0011	0101	0010
03	0011	03	3	0010	0110	0011
04	0100	04	4	0110	0111	0100
05	0101	05	5	0111	1000	1011
06	0110	06	6	0101	1001	1100
07	0111	07	7	0100	1010	1101
08	1000	10	8	1100	1011	1110
09	1001	11	9	1101	1100	1111
10	1010	12	A	1111		
11	1011	13	B	1110		
12	1100	14	C	1010		
13	1101	15	D	1011		
14	1110	16	E	1001		
15	1111	17	F	1000		

8. Les codes alphanumériques

Un ordinateur ne serait pas d'une bien grande utilité s'il était incapable de traiter l'information non numérique. On veut dire par-là qu'un ordinateur doit reconnaître des codes qui correspondent à des nombres, des lettres, des signes de ponctuation et des caractères spéciaux. Les codes de ce genre sont dit alphanumériques. Un ensemble de caractères complet doit renfermer les 26 lettres minuscules, les 26 lettres majuscules, les dix chiffres, les 7 signes de ponctuation et entre 20 à 40 caractères spéciaux comme +, /, #, %. On peut conclure qu'un code alphanumérique reproduit tous les caractères et les diverses fonctions que l'on retrouve sur un clavier standard de machine à écrire ou d'ordinateur.

Le Code ASCII

Le code alphanumérique le plus répandu est le code ASCII (American Standard Code for Information Interchange); on le retrouve dans la majorité des micro-ordinateurs et des mini-ordinateurs et dans beaucoup de gros ordinateurs. Le code ASCII (prononcé "aski") standard est un code sur 7 bits, on peut donc représenter 2^7 = 128 éléments codés. C'est amplement suffisant pour reproduire toutes les lettres courantes d'un clavier et les fonctions de contrôle comme (RETOUR) et (INTERLIGNE).

9. Le code p parmi n

Le code p parmi n est un code à n bits dont p bits sont à 1 et n ï p bits sont à 0. Ce code est auto correcteur car la lecture du code peut être associée à la vérification du nombre de 1 et de 0 dans lôinformation et permet ainsi la détection dôune éventuelle erreur. Exemple : le code 3 parmi 5 est utilisé dans le code postal.

LE CODE ASCII

American Standard Code for Information Interchange

Table en codage hexadécimal

HEX	0	1	2	3	4	5	6	7
0		(DEL)	SP	0	@	P	`	p
1	(SOH)	DC1	!	1	A	Q	a	q
2	(STX)	DC2	«	2	B	R	b	r
3	(ETX)	DC3	#	3	C	S	c	s
4	(EOT)	DC4	$	4	D	T	d	t
5	(ENQ)	(NAK)	%	5	E	U	e	u
6	(ACK)	(SYN)	&	6	F	V	f	v
7	BEL	(ETB)	'	7	G	W	g	w
8	(BS)	CAN	(	8	H	X	h	x
9	(HT)	EM	)	9	I	Y	i	y
A	(LF)	SUB	*	:	J	Z	j	z
B	(VT)	ESC	+	;	K	[	k	{
C	(FF)	(→)	,	<	L	\	l	|
D	(CR)	(←)	–	=	M	]	m	}
E	SO	(↑)	.	>	N	^	n	~
F	SI	(↓)	/	?	O	_	o	

Exemple dôutilisation : lettre G ⇒ code ASCII : 47
lettre k ⇒ code ASCII : 6B

Légende:

SOH	Début d'en-tête	US	Séparateur de sous-articles
STX	Début de texte	RS	Séparateur d'articles
ETX	Fin de texte	GS	Séparateur de groupes
EOT	Fin de transmission	RS	Séparateur de fichiers
ENQ	Demande		
ACK	Accusé de réception	BEL	Sonnerie
DLE	Echappement de transmis-sion	SO	Hors code
NAK	Accusé de réception négatif	SI	En code
SYN	Synchronisation	CAN	Annulation
ETB	Fin de bloc de transmission	EM	Fin de support
BS	Espace arrière	SUB	Substitution
HT	Tabulateur horizontal	ESC	Echappement
LF	Interligne	SP	Espace
CR	Retour de chariot	NUL	Nul
DC1	Marche lecteur	DEL	Oblitération
DC2	Embrayage perforateur		
DC3	Arrêt lecteur		
DC4	Débrayage perforateur	Nat	usage national

10. Représentation des nombres entiers signés

Nous devons définir une convention pour représenter le signe en binaire. La plupart des ordinateurs traitent aussi bien les nombres négatifs que les nombres positifs. La première solution consiste à ajouter un bit au nombre. Celui-ci est appelé bit de signe. La convention la plus simple consiste à attribuer au signe positif l'état 0 et au signe négatif l'état 1. Nous appelons cette convention comme étant une représentation signe-grandeur. On utilise le bit de signe pour indiquer si le nombre binaire mémorisé est positif ou négatif.

Bit de signe ↗ 0 1 0 0 0 1 1 0 = $+70_{10}$

Bit de signe ↗ 1 0 1 1 0 0 1 0 = -78_{10}

11. Notation en complément à 1

Le complément à 1 d'un nombre binaire s'obtient en changeant chaque 0 par un 1 et chaque 1 par un 0. Autrement dit, en complémentant chaque bit du nombre. Voici une illustration de cette marche à suivre: 1 0 1 1 0 1 nombre binaire initial 0 1 0 0 1 0 complément de chaque bit pour obtenir le complément à 1, on dit que le complément à 1 de 101101 est 010010. Le complément à 1 d'un nombre est donc l'inversion de chaque

bit à l'aide de la fonction logique NON. Nous pouvons donc exprimer le complément à 1 par l'équation ci-dessous. Complément à 1 de N : C1(N) = not N

12. Notation en complément à 2

Le complément à 2 est très largement utilisé car c'est la représentation naturelle des nombres négatifs. Si nous faisons la soustraction de 2 - 3 nous obtenons immédiatement -1 représenté en complément à 2.

```
    0010    nombre 2 en binaire sur 4 bits
-   0011    nombre 3 en binaire sur 4 bits
=   1111    résultat de la soustraction, il y a un emprunt
```

Nous allons voir que "1111" est la représentation du nombre -1 sur 4 bits. Le complément à 2 d'un nombre binaire s'obtient simplement en prenant le complément à 1 de ce nombre et en ajoutant 1 au bit de son rang de poids le plus faible. Voici une illustration de cette conversion pour le cas $101101_2 = 45_{10}$.

```
    101101      équivalent binaire de 45
    010010      inversion de chaque bit pour obtenir le complément à 1
+        1      addition de 1 pour obtenir le complément à 2
  = 010011      le complément à 2 du nombre binaire initial
```

On dit que 010011 est le complément à 2 de 101101. Le complément à 2 d'un nombre est donc l'inversion de chaque bit à l'aide de la fonction logique NON, puis l'addition de 1. Nous pouvons donc exprimer le complément à 2 par l'équation ci-dessous.
Complément à 2 de N : C2(N) = C1(N) + 1 = not N + 1

Voici la valeur du nombre -1 sur 4 bits. Nous commençons par exprimer le nombre 1 sur 4 bits puis nous appliquons la règle de calcul du complément à 2.

```
    0001    équivalent binaire de 1 sur 4 bits
    1110    inversion de chaque bit pour obtenir le complément à 1
+      1    addition de 1 pour obtenir le complément à 2
=   1111    le complément à 2 du nombre 1, soit -1
```

13. Etude de nombres binaires signés en complément à 2

Voici comment on écrit des nombres binaires signés en utilisant la notation en complément à 2. Si le nombre est positif, sa grandeur est la grandeur binaire exacte et son bit de signe est un 0 devant le bit de poids le plus fort.

```
                0  0 1 0 1 1 0 1   = + 45(10)
Bit de signe ↗     \_____________/
                      Grandeur exacte
```

```
                1  1 0 1 0 0 1 1   = - 45(10)
Bit de signe ↗     \_____________/
                     Complément à 2
```

Si le nombre est négatif, sa grandeur est le complément à 2 de la grandeur exacte et son bit de signe est un 1 à gauche du bit de poids le plus fort. La complémentation à 2 d'un nombre signé transforme un nombre positif en un nombre négatif et vice versa. La notation en complément à 2 est employée pour exprimer les nombres binaires signés parce que, comme nous le verrons, on parvient grâce à elle à soustraire en effectuant en réalité une addition. Cela n'est pas négligeable dans le cas des ordinateurs, puisque avec les mêmes circuits, nous parvenons à soustraire et à additionner. Dans de nombreuses situations, le nombre de bits est fixé par la longueur des registres qui contiennent les nombres binaires, d'où la nécessité d'ajouter des 0 pour avoir le nombre de bits requis:
Comme exemple nous allons exprimer +2 au moyen de 5 bits:

```
+2  =   00010
        11101     (complément à 1)
      +     1     (ajouter 1)
       = 11110     (complément à 2 du chiffre -2 sur 5 bits)
```

14. Addition en complément à 2

La notation en complément à 2 et la notation en complément à 1 sont très semblables. Toutefois, la notation en complément à 2 jouit généralement de certains avantages quand vient le temps de construire des circuits. Nous allons maintenant étudier comment les machines numériques additionnent et soustraient quand les nombres négatifs sont écrits dans la notation en complément à 2. Dans tous les cas étudiés, il est important que vous remarquiez que le bit de signe de chaque nombre est traité sur le même pied que les bits de la partie grandeur.

14.1 Cas 1: deux nombres positifs

L'addition de deux nombres positifs est immédiate. Soit l'addition de +9 et +4:

```
 + 9          0  1001
 + 4          0  0100
------       ------
 + 13         0  1101   (Somme)
        Bit de signe ↗
```

14.2 Cas 2: nombre positif et nombre négatif plus petit

Soit l'addition de +9 et de -4. Rappelez-vous que -4 est exprimé dans la notation en complément à 2. Donc +4 (00100) doit être converti en -4 (11100)

```
 + 9          0  1001
 - 4          1  1100
------       ------
 + 5       1  0  0101
   Bit rejeté ↗    ↖ Bit de signe
```

Remarquez que les bits de signe sont aussi additionnés. En fait, un report est produit au moment de l'addition du dernier rang. Ce report est toujours rejeté d'où la somme finale de 00101, soit le nombre décimal +5.

Cas 3: nombre positif et nombre négatif plus grand

Soit l'addition de -9 et de +4:

- 9	1 0111
+ 4	0 0100
- 5	1 1011

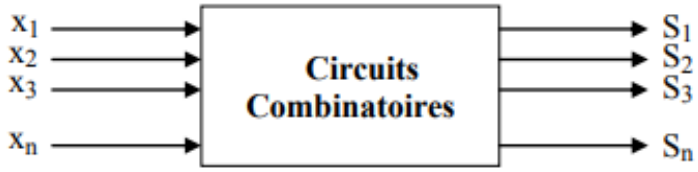
Dans ce cas-ci le bit de signe de la somme est 1, ce qui indique un nombre négatif. Comme la somme est un nombre négatif, la réponse est le complément à 2 de la grandeur exacte. Donc 1011 est en réalité le complément à 2 de la somme. Pour trouver la grandeur exacte de la somme, on doit prendre le complément à 2 de 1011, ce qui donne 0101 (5); la réponse est donc 11011 = -5.

CHAPITRE 3

Circuits combinatoires transcodeurs

1. Introduction

La transmission de données nécessite fréquemment des opérations de conversion, de transpostage et d'aiguillage. On utilise pour cela des circuits combinatoires. Pour réaliser un circuit logique combinatoire, le concepteur doit utiliser plusieurs portes logiques élémentaires. Pour faciliter sa tache, les fabricants fournissent des circuits sous forme intégrés comportant chacun plusieurs portes à des degrés d'intégration différents. Il existe plusieurs dispositifs logiques combinatoires couramment utilisé dans les systèmes numériques. On peut citer les codeurs, les décodeurs, les transcodeurs, les multiplexeurs, les démultiplexeurs, les comparateurs …



2. Analyse de circuit logique:

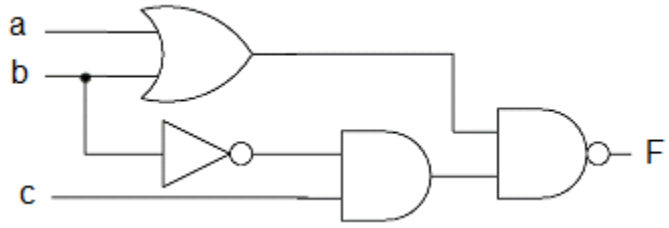
A partir du logigramme d'un circuit, Trouver sa fonction logique

- **Principe**

Donner l'expression des sorties de chaque porte/composant en fonction des valeurs de ses entrées, En déduire au final la (ou les) fonction(s) logique(s) du circuit.
- On peut ensuite Déterminer la table de vérité du circuit, Simplifier la fonction logique à l'aide des propriétés de l'algèbre de Boole ou les tableaux de Karnaugh

- **Exemple**

Analyse d'un circuit 3 entrées, 1 sortie



- Quelle est la fonction logique de ce circuit ?
  A partir de son logigramme
  $F(a, b, c) = \overline{(a + b).(\overline{b} . c)}$

- Après simplification, on obtient
  $F(a, b, c) = \overline{a} + b + \overline{c}$
  En nombre de portes minimales
  $F(a, b, c) = \overline{a.c} + b$

- **Table de vérité**

a	b	c	a + b = x	$\overline{b}$	$\overline{b} \cdot c = y$	x.y	$\overline{x \cdot y}$
0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1
0	1	0	1	0	0	0	1
0	1	1	1	0	0	0	1
1	0	0	1	1	0	0	1
1	0	1	1	1	1	1	0
1	1	0	1	0	0	0	1
1	1	1	1	0	0	0	1

- **Table de vérité de la fonction simplifiée**

a	b	c	$\overline{a}$	$\overline{c}$	$\overline{a} + b + \overline{c}$
0	0	0	1	1	1
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	0	0	1

- On Trouve les mêmes valeurs dans les 2 tables

3. Synthèse d'une fonction logique :

La synthèse de fonctions combinatoires consiste, à partir d'une table de vérité ou d'une expression booléenne, trouver le logigramme correspondant à cette fonction

- **Principe**

Simplifier la fonction logique avec une des deux méthodes :

- La méthode algébrique (algèbre de Boole)
- La méthode des tableaux de Karnaugh

En déduire le logigramme correspondant

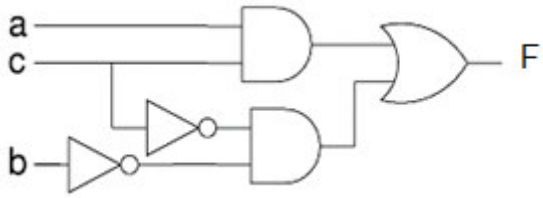
- **Exemple 1**

- Soit la fonction

F(a, b, c) = $abc + a\overline{b}\overline{c} + \overline{a}\overline{b}\overline{c} + a\overline{b}c$

- Après simplification, on obtient

F(a, b, c) = $a\,c + \overline{b}\,\overline{c}$



- **Exemple 2**

E	a	b	c	F
E_0	0	0	0	0
E_1	0	0	1	1
E_3	0	1	1	0
E_2	0	1	0	1
E_6	1	1	0	1
E_7	1	1	1	0
E_5	1	0	1	0
E_4	1	0	0	1

Pour établir l'expression booléenne deux possibilités nous sont offertes :
- soit considérer les états Ei de E pour lesquels F est égal à 1(développement de F suivant les états 1)
- soit considérer les états Ei de E pour lesquels F est égal à 0(développement de F suivant les états 0).

a) Développement de F d'après les états 1 :

F = 1.

$E_1 = \bar{a}\,\bar{b}\,c$

$E_2 = \bar{a}\,b\,\bar{c}$ $\Longrightarrow$ $E_1 + E_2 + E_4 + E_6 = 1$

$E_6 = a\,b\,\bar{c}$ $\bar{a}\,\bar{b}\,c + \bar{a}\,b\,\bar{c} + a\,b\,\bar{c} + a\,\bar{b}\,\bar{c} = 1$

$E_4 = a\,\bar{b}\,\bar{c}$

c.a.d. **F =** $\bar{a}\,\bar{b}\,c$ **+** $\bar{a}\,b\,\bar{c}$ **+** $a\,b\,\bar{c}$ **+** $a\,\bar{b}\,\bar{c}$ (forme canonique de F_1 (a,b,c))

b) Développement de F d'après les états 0 :

F = 0.

$E_0 = \bar{a}\,\bar{b}\,\bar{c}$;

$E_3 = \bar{a}\,b\,c$; $\Longrightarrow$ $E_0 + E_3 + E_7 + E_5 = 0$

$E_7 = a\,b\,c$; $\bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,b\,c + a\,b\,c + a\,\bar{b}\,c = 0$

$E_5 = a\,\bar{b}\,c$;

c.a.d. $\bar{F} = \bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,b\,c + a\,b\,c + a\,\bar{b}\,c$

Le complément de $\overline{F}$ est $\overline{\overline{F}} = \overline{\bar{a}\,\bar{b}\,\bar{c} + \bar{a}\,b\,c + a\,b\,c + a\,\bar{b}\,c}$

D'où F = $(a+b+c)(a+\bar{b}+\bar{c})(\bar{a}+\bar{b}+\bar{c})(\bar{a}+b+\bar{c})$ (forme canonique de F_0(a,b,c))

Remarque : Généralement on utilise le développement de F d'après ces états 1.

- **Autres exemples**

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

F = 1, donc $F = \bar{a}\,\bar{b}\,c + a\,\bar{b}\,c + a\,b\,c$

a	b	P
0	0	0
0	1	0
1	0	0
1	1	1

P = 1, donc P = a b.

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

$\Longrightarrow$

S = 1, $S = \bar{a}\,b + a\,\bar{b} + a\,b$

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

$S = b + a\,\bar{b}$

$S = (b + a).(b + \bar{b})$

S = a + b

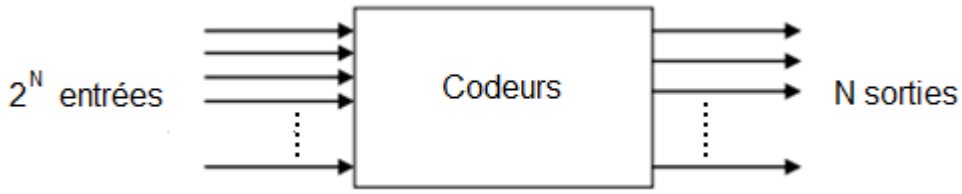
ou bien S = 0 , $\bar{S} = \bar{a}\,\bar{b} = \overline{a + b}$ $\Longrightarrow$ S = a + b

4. Circuits logiques de base

- Codeur : pour 1 entrée active, fournit un code
- Décodeur : active une des X sorties selon un code en entrée
- Transcodeur : pour un code A fournit un code B
- Multiplexeur : une des X entrées vers 1 sortie (voir chapitre 4)
- Démultiplexeur : 1 entrée vers une des X sorties (voir chapitre 4)
- Comparateur : comparaison entre deux nombres binaires A et B (voir chapitre 5)
- Additionneur : Pour réaliser l'addition de deux nombres binaires A et B

5. Les Codeurs

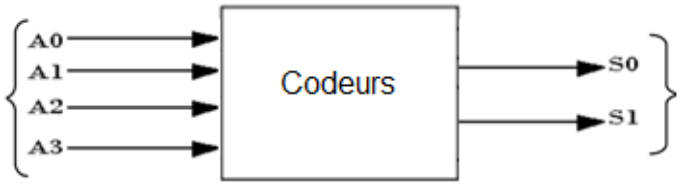
5.1. Définition : Le codeur (ou encodeur) est un circuit logique qui possède 2^N voies entrées, dont une seule est activée et N voies de sorties. Il fournit en sortie le code binaire correspondant.



5.2. Principe d'un codeur 4 voies d'entrées et 2 bits de sortie

- Schéma fonctionnel

4 entrées dont une seule est activée à la fois



Représentation binaire de l'entrée activée (n bits)

- Table de vérité

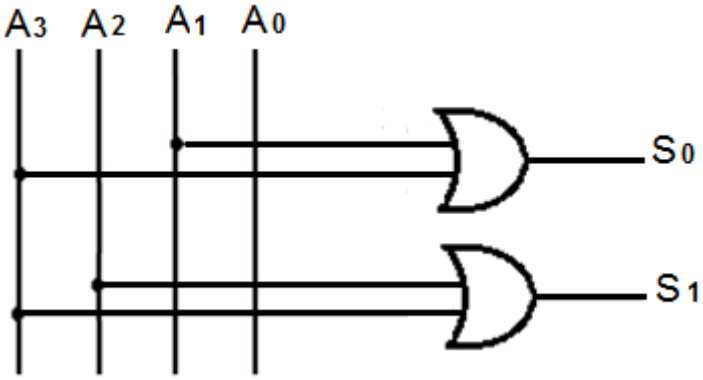
Entées				Sorties	
Codage 1 parmi 4				Nombre binaire de 2 bits	
A_3	A_2	A_1	A_0	S_1	S_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- Equation des sorties

S_0=1 si (A_1=1) ou (A_3=1) donc $S_0=A_1+A_3$

S_1=1 si (A_2=1) ou (A_3=1) donc $S_1=A_2+A_3$

- Logigramme



5.3. Codeur sur 3 bits

Il a 8 entrées E_y et 3 bits S_x en sortie

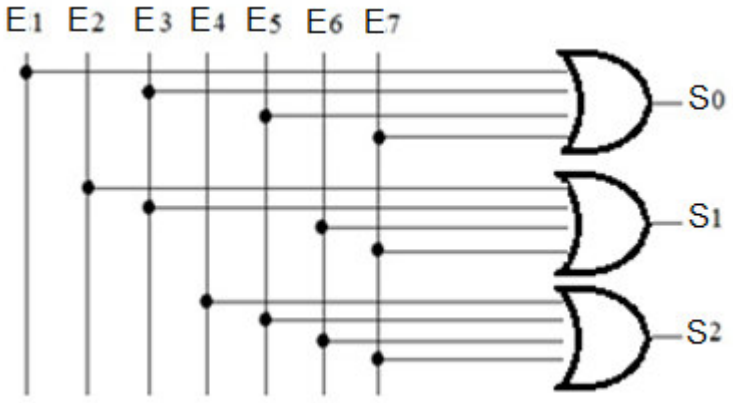
- Table de vérité

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

- Equation des sorties

$S_0 = E_1+E_3+E_5+E_7$
$S_1 = E_2+E_3+E_6+E_7$
$S_2 = E_4+E_5+E_6+E_7$

- Logigramme



5.4. Codeur de priorité

C'est un dispositif qui réalise le codage du numéro le plus élevé dans le cas ou plusieurs entrées seraient actionnées. Pour cette raison, ce codeur possède des circuits logiques en plus, de sorte que le code de sortie choisi quand deux entrés sont actives soit celui qui correspond au nombre supérieur

- Table de vérité

Entrées									Sorties			
E_9	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	X	0	0	1	0
0	0	0	0	0	0	1	X	X	0	0	1	1
0	0	0	0	0	1	X	X	X	0	1	0	0
0	0	0	0	1	X	X	X	X	0	1	0	1
0	0	0	1	X	X	X	X	X	0	1	1	0
0	0	1	X	X	X	X	X	X	0	1	1	1
0	1	X	X	X	X	X	X	X	1	0	0	0
1	X	X	X	X	X	X	X	X	1	0	0	1

5.5. Codeurs en circuits intégrés

- Codeur BCD de priorité 74147

Le circuit intégré 74147 est un codeur de priorité à 9 entrées. Il est actif à l'état bas et produit à la sortie le code BCD inversé.

- Table de vérité

Entrées									Sorties			
$\overline{E}_9$	$\overline{E}_8$	$\overline{E}_7$	$\overline{E}_6$	$\overline{E}_5$	$\overline{E}_4$	$\overline{E}_3$	$\overline{E}_2$	$\overline{E}_1$	$\overline{S}_3$	$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	0	X	1	1	0	1
1	1	1	1	1	1	0	X	X	1	1	0	0
1	1	1	1	1	0	X	X	X	1	0	1	1
1	1	1	1	0	X	X	X	X	1	0	1	0
1	1	1	0	X	X	X	X	X	1	0	0	1
1	1	0	X	X	X	X	X	X	1	0	0	0
1	0	X	X	X	X	X	X	X	0	1	1	1
0	X	X	X	X	X	X	X	X	0	1	1	0

Les sortie de 71747 sont à 1 quand aucune des entrés n'est à son niveau vrai (bas), cela correspond au code inversé du chiffre 0.

Pour obtenir le code B.C.D à partir des sorties de 74147, il faut ajouter un inverseur à chacune des sorties.

5.6. Codeur prioritaire à 3 bits 74148

Le 74148 est un codeur de priorité à huit entrés, actifs à l'état bas. Le code de sortie est un code en binaire inversé. C'est un codeur très utile car il permet non seulement le codage d'un nombre à huit entrées mais un nombre supérieur.

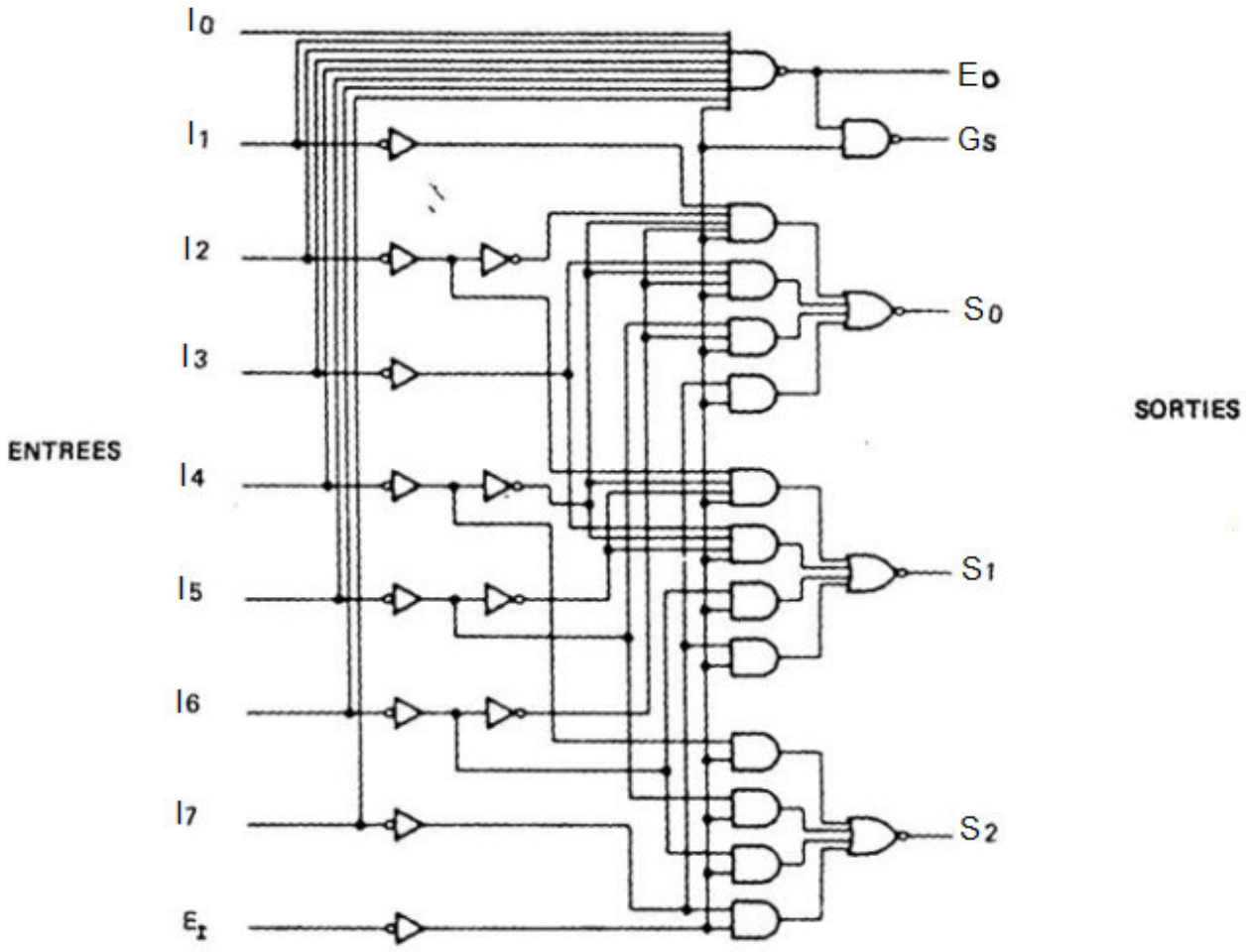
- Table de vérité

Entrées									Sorties				
$\overline{E}_I$	$\overline{I}_7$	$\overline{I}_6$	$\overline{I}_5$	$\overline{I}_4$	$\overline{I}_3$	$\overline{I}_2$	$\overline{I}_1$	$\overline{I}_0$	$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	$\overline{E}_O$	$\overline{G}_S$
0	1	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	0	1	1	1	1	0
0	1	1	1	1	1	1	0	X	1	1	0	1	0
0	1	1	1	1	1	0	X	X	1	0	1	1	0
0	1	1	1	1	0	X	X	X	1	0	0	1	0
0	1	1	1	0	X	X	X	X	0	1	1	1	0
0	1	1	0	X	X	X	X	X	0	1	0	1	0
0	1	0	X	X	X	X	X	X	0	0	1	1	0
0	0	X	X	X	X	X	X	X	0	0	0	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

Ce codeur possède en plus des entres classiques du codeur de priorité, trois broches supplémentaires E_O , E_I et Gs . Le rôle de chacune de ces broches est décrit ainsi :

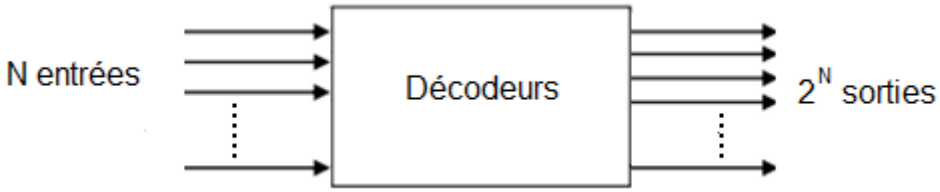
- Si l'entrée $\overline{E}_I = 1$, alors le codeur n'est pas validé et les sorties sont à 1 quelles que soient les entrés c'est-à-dire $\overline{S}_2 = \overline{S}_1 = \overline{S}_0 = \overline{G}_S = \overline{E}_O = 1$
- Si l'entrée $\overline{E}_I = 0$, alors le codeur est validé et fournit le code correspondant à l'entrée prioritaire qui se trouve à l'état bas.
- Si l'entrée $\overline{E}_I = 0$, et si toutes les entrés Ii sont à 1(pas d'informations sur les entrées), alors 1 sortie $\overline{E}_O$ est à l'état bas.
- Les conditions $\overline{G}_S = 0$ et $\overline{E}_I$ = 0, indiquent la présence d'au moins une information sur une entrées

- Logigramme



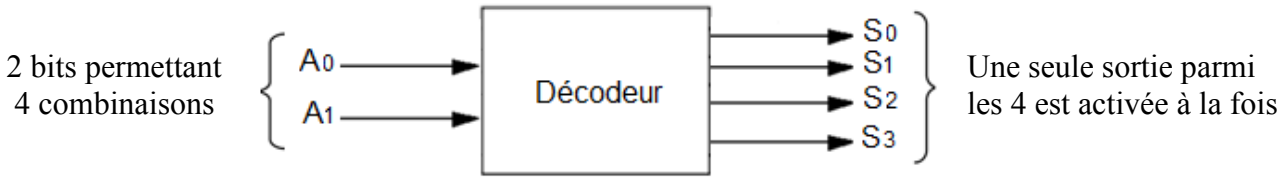
6. Les décodeurs

6.1. Définition : Le décodeur est un circuit logique qui possède N entrées et en général 2^N sorties dont une seule est activée à la fois.



6.2. Principe d'un décodeur 2 entrées et 4 sorties

- Schéma fonctionnel



- Table de vérité

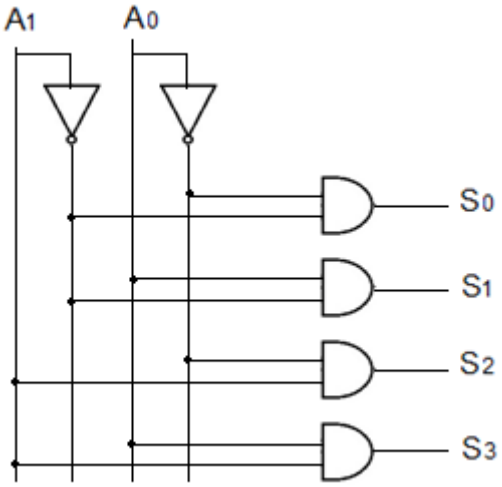
Entées		Sorties			
Nombre binaire de 2 bits		Codage 1 parmi 4			
A_1	A_0	S_3	S_2	S_1	S_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

- Equation des sorties

$S_0 = \overline{A_1}\ \overline{A_0}$ $S_2 = A_1\ \overline{A_0}$

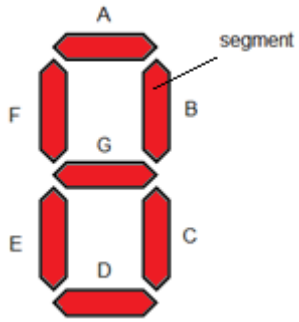
$S_1 = \overline{A_1}\ A_0$ $S_3 = A_1\ A_0$

- Logigramme



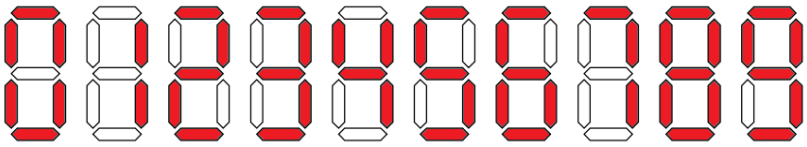
6.2. Décodeurs BCD 7 segments

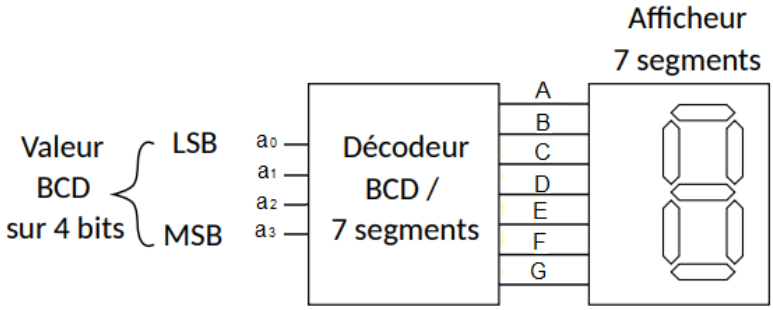
Les 10 chiffres décimaux (0 à 9) peuvent être configurés au moyen de 7 segments. Chaque segment est constitué d'un matériau qui émet de la lumière lorsqu'il est traversé par un courant.



- Schéma fonctionnel du décodeur

On propose ici de réaliser un décodeur BCD / 7 segments pour pouvoir afficher une valeur BCD sur un afficheur. Ci-dessous sont représentées les différentes configurations désirées de l'afficheur en fonction des valeurs d'entrées (de 0000_2 à 1001_2)





- Table de vérité

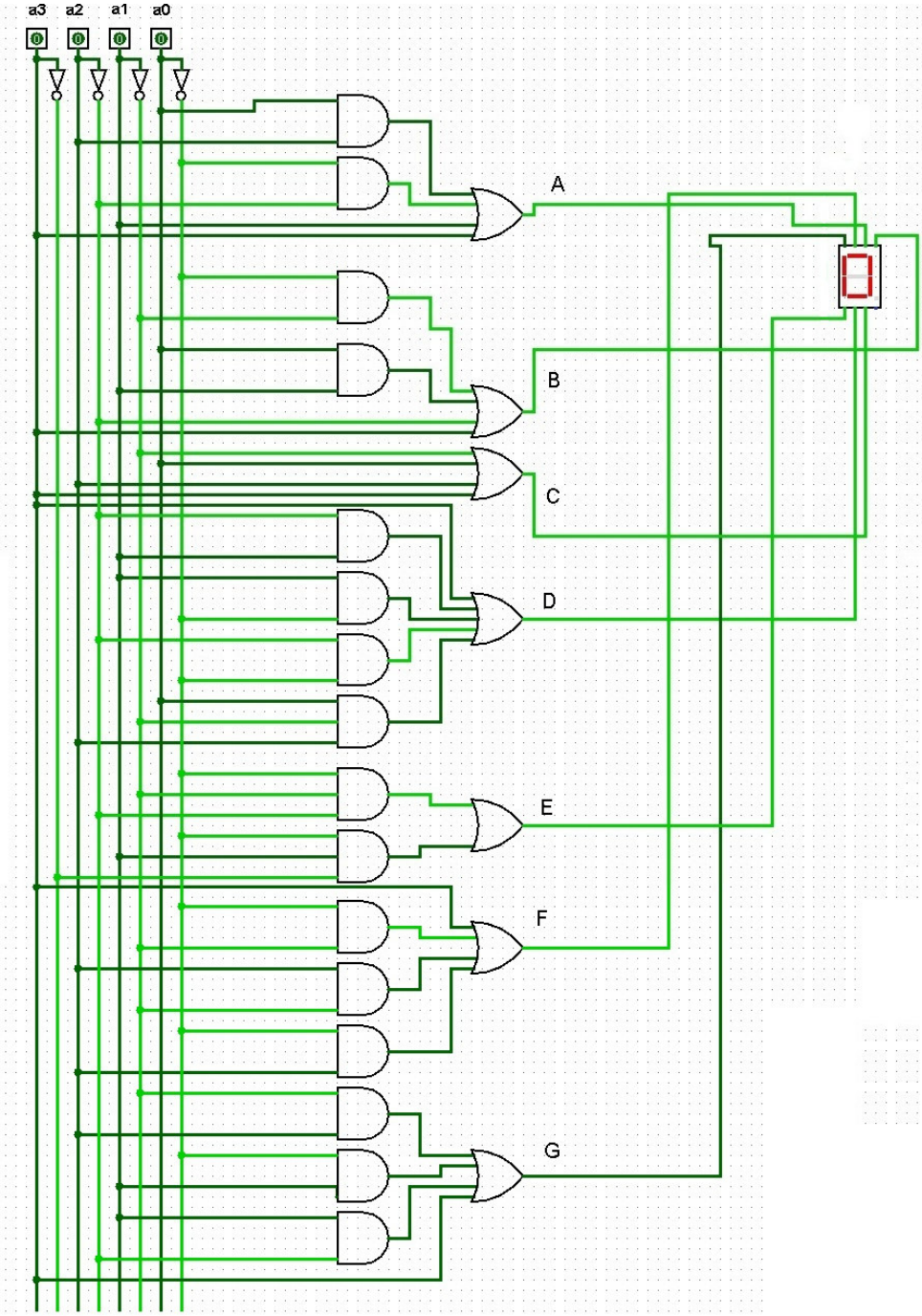
Entrées				Sorties							
a_3	a_2	a_1	a_0	A	B	C	D	E	F	G	Affichage
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9

- Equation des sorties

Après simplification à l'aide de tableau de karnaugh on obtient :

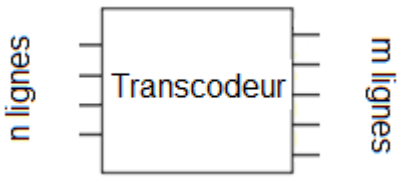
$$A = a_3 + a_1 + a_2a_0 + \overline{a_2}\,\overline{a_0}$$
$$B = a_3 + \overline{a_2} + \overline{a_1}\,\overline{a_0} + a_1a_0$$
$$C = \overline{a_1} + a_0 + a_2 + a_3$$
$$D = a_3 + \overline{a_2}\,a_1 + a_1\overline{a_0} + \overline{a_2a_0} + a_2\overline{a_1}a_0$$
$$E = \overline{a_2}\overline{a_1}\overline{a_1} + \overline{a_3}a_1\overline{a_0}$$
$$F = a_3 + \overline{a_1}\overline{a_0} + a_2\overline{a_1} + a_2\overline{a_0}$$
$$G = a_3 + a_2\overline{a_1} + a_1\overline{a_0} + \overline{a_2}a_1$$

- Logigramme



7. Transcodeur

Un transcodeur fait correspondre un code A en entrée sur n lignes à un code B en sortie sur m lignes.



7.1. Exemple d'un transcodeur n = 3 et m = 5

- Table de vérité

a2	a1	a0	b4	b3	b2	b1	b0
0	0	0	1	0	0	1	0
0	0	1	0	1	0	0	1
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	0	0	1	0	1
1	0	1	1	0	0	1	0
1	1	0	0	1	0	0	1
1	1	1	1	0	1	0	0

- Tableau de Karnaugh

a2 \ a1 a0	00	01	11	10
0	0	1	0	0
1	1	0	0	1

a2 \ a1 a0	00	01	11	10
0	1	0	1	0
1	0	1	0	0

a2 \ a1 a0	00	01	11	10
0	0	0	0	1
1	1	0	1	0

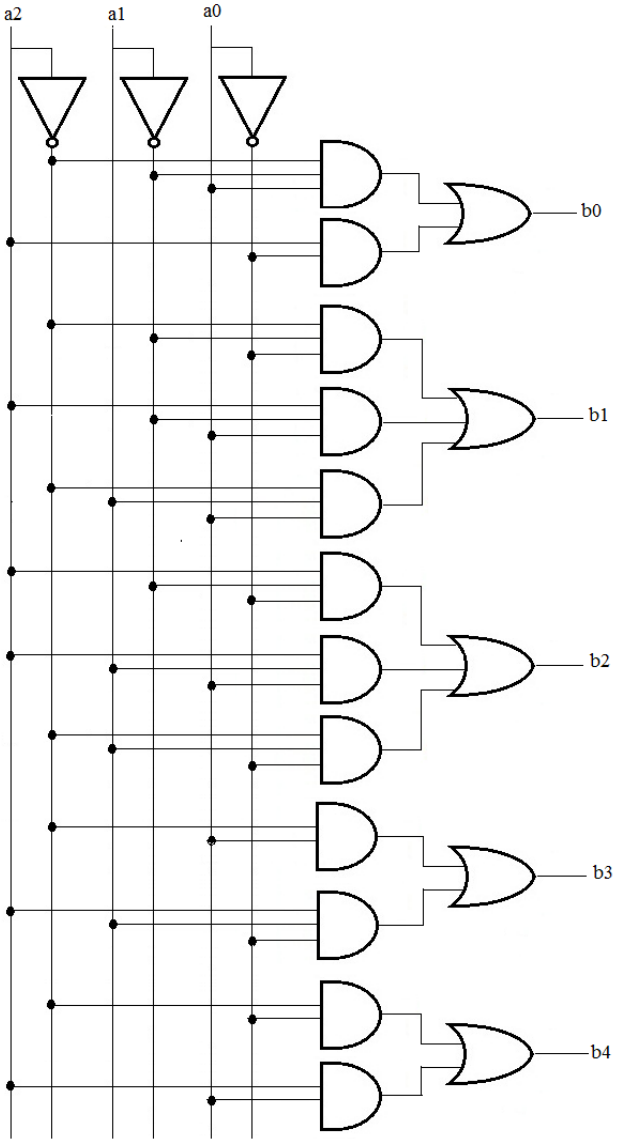
$b0 = \overline{a2}\,\overline{a1}\,a0 + a2\,\overline{a0}$ $b1 = \overline{a2}\,\overline{a1}\,\overline{a0} + a2\,\overline{a1}\,a0 + \overline{a2}\,a1\,a0$ $b2 = a2\,\overline{a1}\,\overline{a0} + a2\,a1\,a0 + \overline{a2}\,a1\,\overline{a0}$

a2 \ a1 a0	00	01	11	10
0	0	1	1	0
1	0	0	0	1

a2 \ a1 a0	00	01	11	10
0	1	0	0	1
1	0	1	1	0

$b3 = \overline{a2}\,a0 + a2\,a1\,\overline{a0}$ $b4 = \overline{a2}\,\overline{a0} + a2\,a0$

- Logigramme



8. Liste des circuits intégrés de décodage

8.1. Quelques circuits intégrés TTL série 7400

Code	Description
7441	décodeur BCD vers décimal
7443	décodeur 4 bits vers décimal incrémenté de 3 (3 à 12)
7445	décodeur BCD vers décimal
7446	décodeur BCD à 7 segments
74184	Convertisseur BCD vers binaire
74185	Convertisseur binaire vers BCD

8.2. Quelques circuits intégrés CMOS série 4000

Code	Description
4026	Décodeurs 7 segments
4033	Décodeurs 7 segments
4511	Décodeurs 7 segments
4513	Décodeurs 7 segments
4543	Décodeurs 7 segments
4558	Décodeurs 7 segments

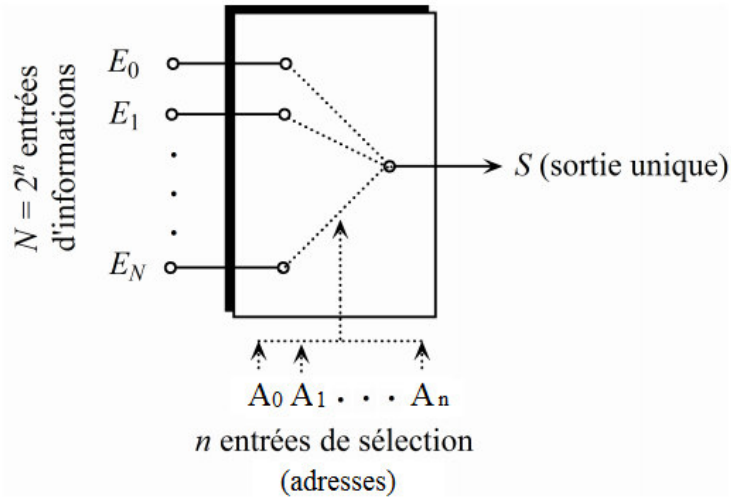
CHAPITRE 4

Circuits combinatoires aiguilleurs

1. Multiplexeur

1.1. Définition

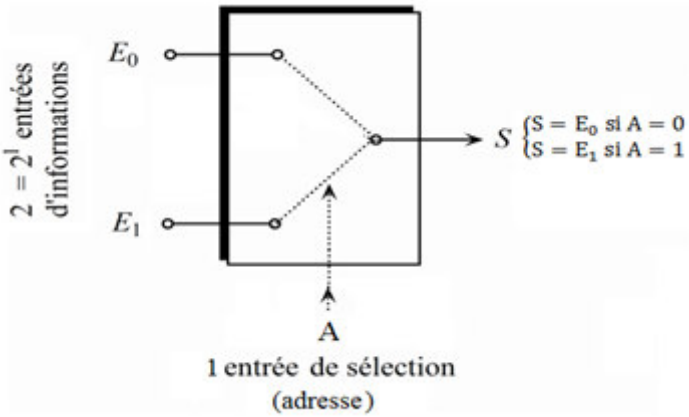
Le multiplexeur (MUX) est un circuit combinatoire permettant de réaliser un aiguillage de l'une des entrées (E1, E2,…, En) en une sortie unique (S) à l'aide des entrées de sélection (A1, A2, ….) appelées adresses, dont la représentation est donnée par le schéma suivant :



1 .2 Multiplexeur 2 vers 1

Il s'agit d'un multiplexeur à 2 entrées (qu'on note E_0 et E_1), qui nécessite une (1) entrée de commande ou ligne d'adresse (qu'on nomme A) et une seule sortie (S).
Son fonctionnement se résume par :

$S = E_0$ si $A = 0$
$S = E_1$ si $A = 1$



- Table de vérité

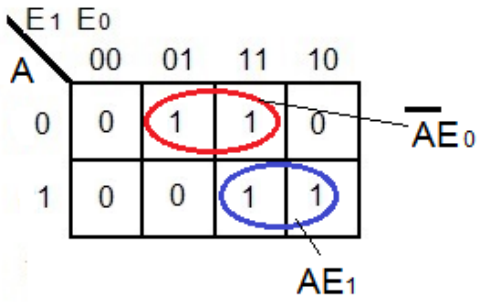
A	E_1	E_0	S	
0	0	0	0	$S= E_0$
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	0	$S=E_1$
1	0	1	0	
1	1	0	1	
1	1	1	1	

On peut réduire la table de vérité comme suit :

A	E_1	E_0	S
0	X	0/1	$E_0 = 0/1$
1	0/1	X	$E_1 = 0/1$

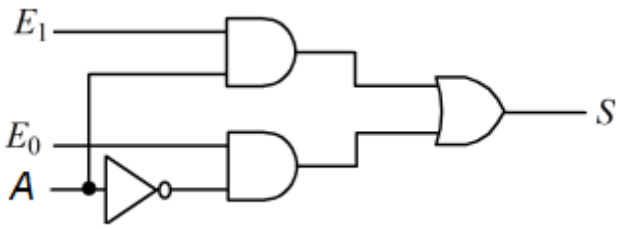
- Equation de sortie

En utilisant le tableau de Karnaugh



$$S = \overline{A}\,E0 + A\,E1$$

- Logigramme



1 .3 Multiplexeur 4 vers 1

C'est un multiplexeur à 4 (2^2) entrées (E0, E1, E2 et E3), qui nécessite 2 entrées de commande (A_0 et A_1) et une seule sortie (S).

- Table de vérité

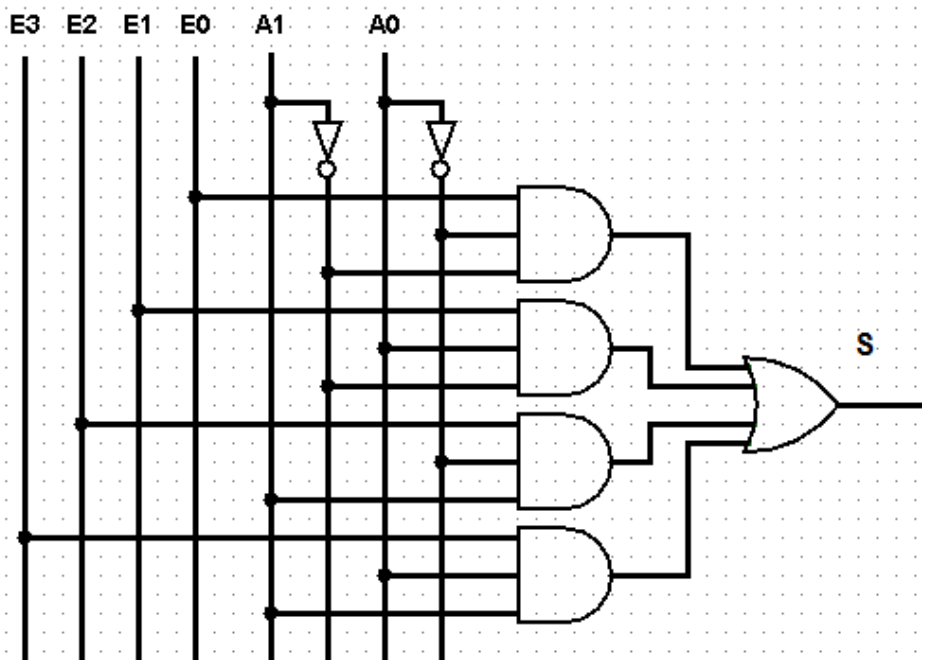
La table de vérité simplifiée est :

A_1	A_0	E_3	E_2	E_1	E_0	S
0	0	X	X	X	0/1	E_0
0	1	X	X	0/1	X	E_1
1	0	X	0/1	X	X	E_2
1	1	0/1	X	X	X	E_3

- Equation de sortie

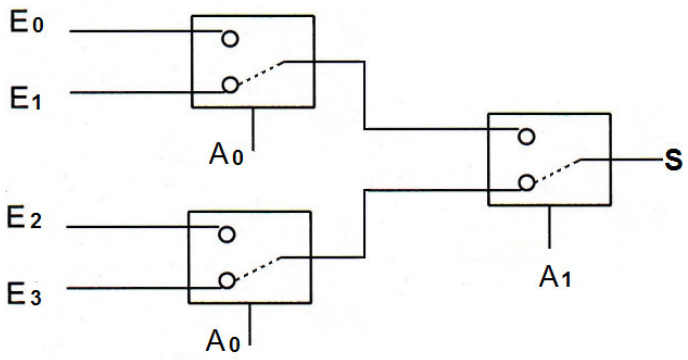
$$S = \overline{A_1}\,\overline{A_0}\,E_0 + \overline{A_1}\,A_0\,E_1 + A_1\,\overline{A_0}\,E_2 + A_1\,A_0\,E_3$$

- Logigramme



2. Mise en cascade des multiplexeurs

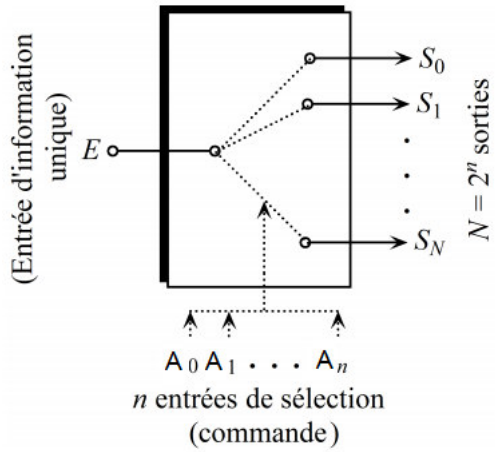
Réalisation d'un multiplexeur 4 vers 1 en utilisant des multiplexeurs 2 vers 1.



3. Démultiplexeur

3.1. Définition

Le démultiplexeur réalise l'opération inverse de celle du multiplexeur. Il comporte une seule entrée d'information E, n entrées de commande Ai avec i = 0,1,...,n (appelées aussi entrées d'adresse ou de sélection) et N = 2^n sorties (S_0, S_1, ... , S_N). Son schéma est illustré par la figure suivante :



3 .2 Démultiplexeur 1 vers 2

C'est un démultiplexeur à 2 sorties (S_0, S_1), qui nécessite 1 entrées de commande (A) et une seule entrée (E).

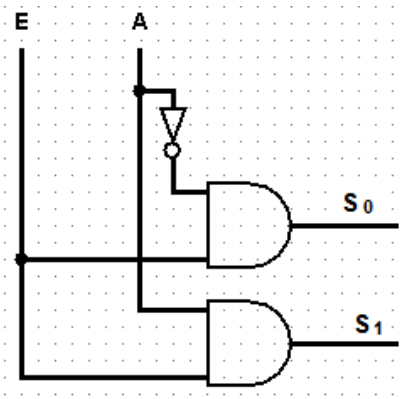
- Table de vérité

A	E	S_1	S_0
0	0/1	0	E= 0/1
1	0/1	E=0/1	0

- Equations des sorties

$S_0 = \overline{A}\,E$, $S_1 = A\,E$

- Logigramme



3 .3 Démultiplexeur 1 vers 4

C'est un démultiplexeur à 4 (2^2) sorties (S_0, S_1, S_2 et S_3), nécessitant 2 entrées de commande (A_0 et A_1) et une seule entrée (E).

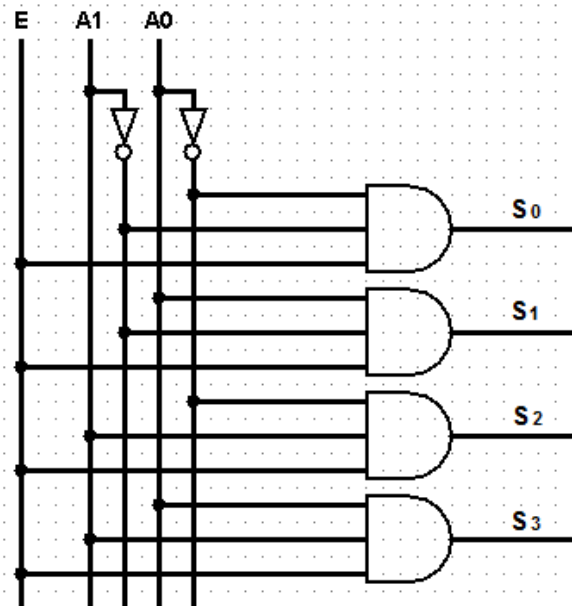
- Table de vérité

A1	A0	E	S3	S2	S_1	S_0
0	0	0/1	0	0	0	E= 0/1
0	1	0/1	0	0	E=0/1	0
1	0	0/1	0	E=0/1	0	0
1	1	0/1	E=0/1	0	0	0

- Equations des sorties

$S_0 = \overline{A_1}\,\overline{A_0}\,E$, $S_1 = \overline{A_1}\,A_0\,E$, $S_2 = A_1\,\overline{A_0}\,E$, $S_3 = A_1A_0E$

- Logigramme



4. Applications des multiplexeurs

- Conversion parallèle/série : aiguiller les informations présentes en parallèle à l'entrée du MUX en des informations de type série en sortie,
- Réalisation de fonctions logiques : toute fonction logique de N variables est réalisable avec un multiplexeur de 2^N vers 1,
- La concentration de données et leur transmission parallèle,
- L'affichage multiplexé sur des afficheurs 7 segments etc….

5. Circuits intégrés

On trouve chez les constructeurs, les circuits intégrés suivants :

- 2 vers 1 : 74157 (4 Mux 2 vers 1)
- 4 vers 1 : 74153 (2 Mux 4 vers 1)
- 8 vers 1 : 74151 (2 Sorties complémentaires), 74152 (1 Sortie complémentée)
- 16 vers 1 : 74150 (1 Sortie complémentée)
- 1 vers 4 : 74139 (2 DMux 1 vers 4, Sorties complémentées)
- 1 vers 8 : 74137, 74138 (Sorties complémentées)
- 1 vers 16 : 74154,74159 (Sorties complémentées)

Chapitre 5 : Circuits combinatoires de comparaison

1. Introduction :

Un comparateur logique est un circuit logique qui effectue la comparaison entre deux nombres binaires généralement notés A et B. IL possède 3 sorties possibles notées :

A = B (A égal à B)

A > B (A est strictement supérieur au nombre B)

et A < B (A est strictement inférieur au nombre B)

Dont :

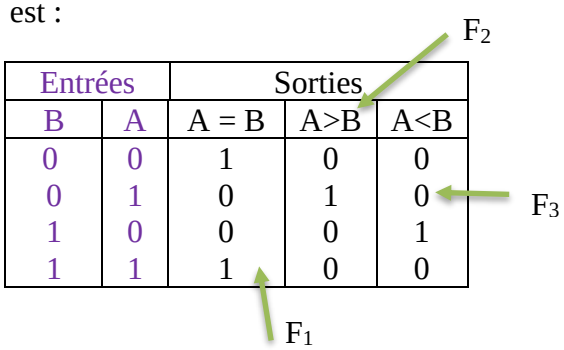
Si A = B, la sortie A = B passe à l'état **1** tandis que les sorties A > B et A < B passent à l'état **0**.

Si le nombre A est strictement supérieur au nombre B, alors la sortie A> B passe à l'état **1** tandis que les sorties A = B et A < B passent à l'état **0**.

Si le nombre A est strictement inférieur au nombre B, seule la sortie A < B passe à l'état **1**.

2. Comparateur de deux Chiffres Binaires sur un bit chacun

Soit à comparer les deux chiffres binaires A et B ; alors la table de vérité de ce comparateur est :



Entrées		Sorties		
B	A	A = B	A>B	A<B
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

On considère que :

F_1 : est la sortie de comparateur lorsque A = B

F_2 : est la sortie de comparateur lorsque A > B

et

F_3 :est la sortie de comparateur lorsque A < B

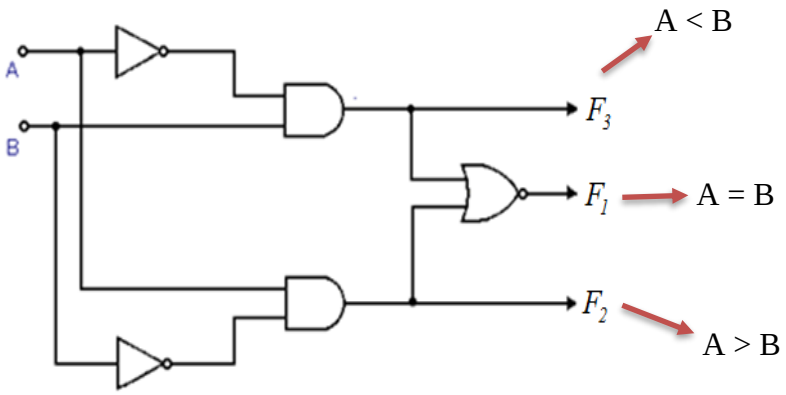
Don d'après la table de vérité on trouve les équations de sorties de ce comparateur :

$F_1 = \overline{A}\,\overline{B} + AB = \overline{A \oplus B}$

$F_2 = A\overline{B}$

$F_3 = \overline{A}B$

Logigramme de ce comparateur (à un bit) est ainsi donné par la figure suivante :



3. Comparateur de deux Chiffres Binaires sur deux bits chacun

Soit à comparer les deux chiffres binaires A et B , dont A =a_1a_0 et B =b_1b_0

Dans ce cas, il faut faire la comparaison entre a_1 et b_1 d'une part et entre a_0 et b_0 d'autre part.

Alors :

A = B si ($a_1 = b_1$) et ($a_0 = b_0$)

A > B si ($a_1 > b_1$) ou ($a_1 = b_1$) et ($a_0 > b_0$)

A < B si ($a_1 < b_1$) ou ($a_1 = b_1$) et ($a_0 < b_0$)

Ce qui donne logigramme suivant :

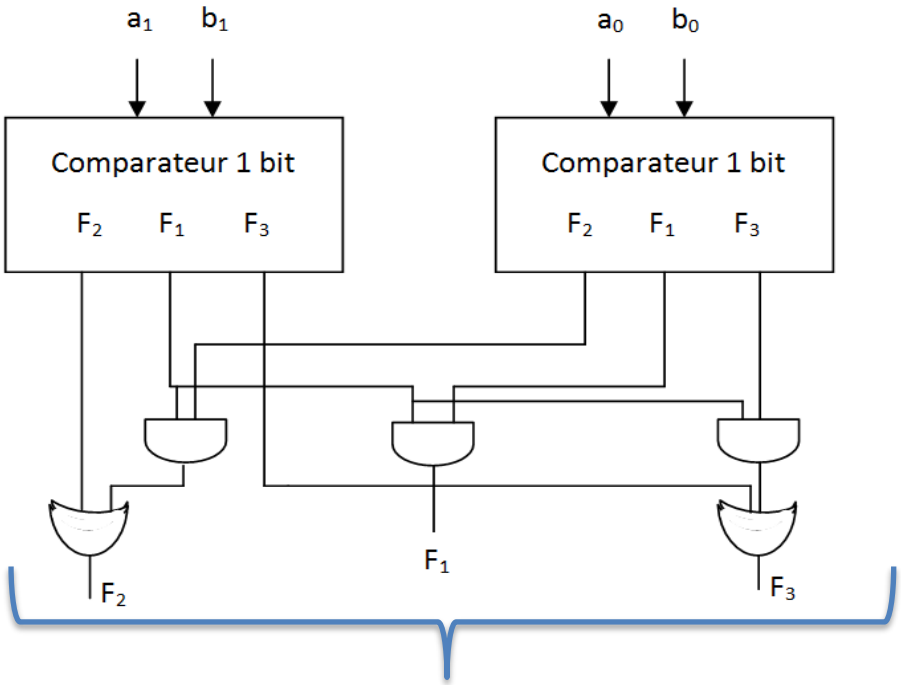


Schéma d'un Comparateur de deux Chiffres Binaires sur deux bits à base d'un comparateur à un bit

Avec :

F_1 : est la sortie de comparateur lorsque A = B

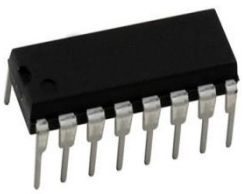
F_2 : est la sortie de comparateur lorsque A > B

et

F_3 :est la sortie de comparateur lorsque A < B

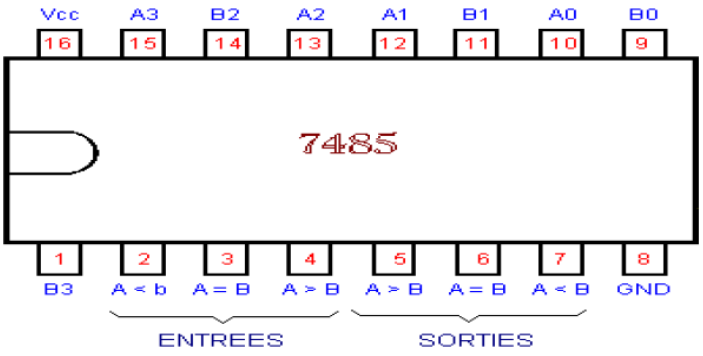
4. Comparateur de deux Chiffres Binaires sur quatre bits chacun

4..1. Exemple : le Comparateur Intégré: 74LS85



Le circuit intégré 74LS85 (ou 7485) est un comparateur 4 bits, c'est-à-dire qu'il effectue la comparaison de deux nombres de 4 bits. De plus, il dispose de 3 entrées notées A = B, A > B et A < B qui autorisent la mise en cascade de plusieurs circuits comparateurs du même type. Ainsi, on peut comparer des nombres de 8, 12, 16 bits....

Le brochage de ce circuit est donné par la figure suivante :



Avec ce circuit, on compare le nombre A composé des bits A3, A2, A1 et A0 (A3 est le bit de poids le plus fort et A0 est le bit de poids le plus faible) avec le nombre B composé des bits B3, B2, B1 et B0 (B3 est le bit de poids le plus fort B0 est le bit de poids le plus faible).

Le circuit logique 7485 est donné par la figure suivante :

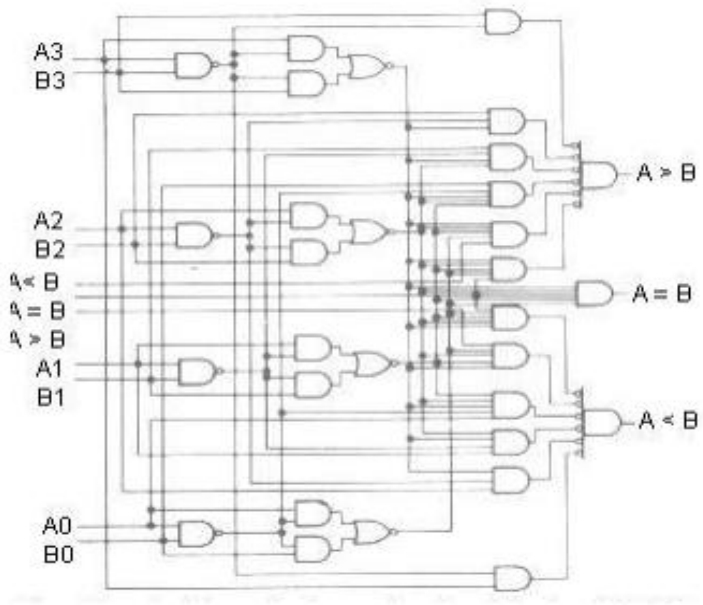


Schéma logique du circuit intégré 7485

Tandis que la table de vérité de ce circuit est donnée par le tableau suivant :

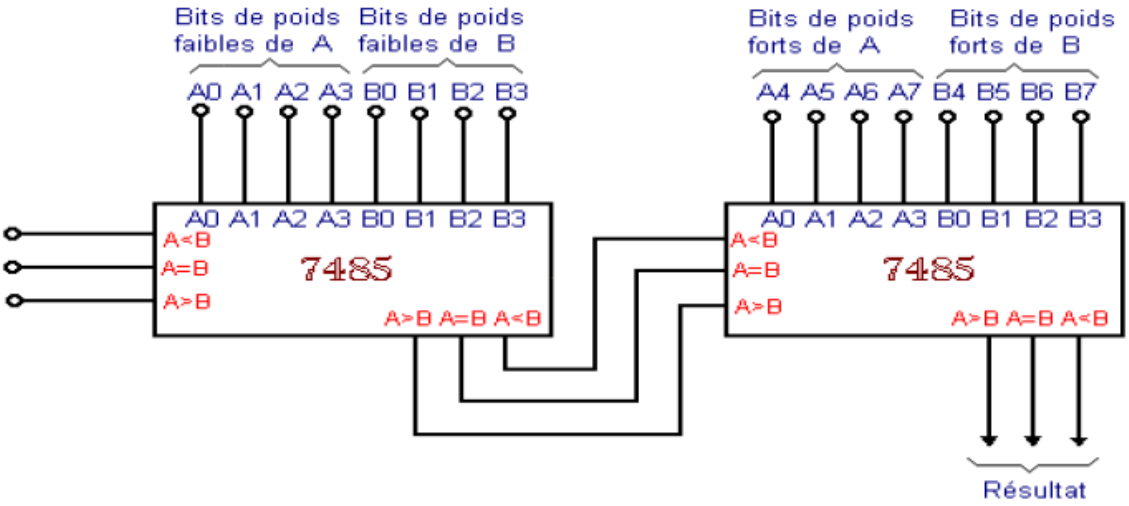
Entrées des nombres				Entrées cascadables			Sorties		
A3, B3	**A2, B2**	**A1, B1**	**A0, B0**	**A > B**	**A < B**	**A = B**	**A > B**	**A < B**	**A = B**
A3 > B3	X	X	X	X	X	X	1	0	0
A3 < B3	X	X	X	X	X	X	0	1	0
A3 = B3	A2 > B2	X	X	X	X	X	1	0	0
A3 = B3	A2 < B2	X	X	X	X	X	0	1	0
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	1	0	0
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	1	0	0
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	0	0	1	0	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	1	0	0	1	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	0	1	0	0	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	1	0	0	1
A3 = B3	A2 = B2	A1 = B1	A0 = B0	1	1	0	0	0	0
A3 = B3	A2 = B2	A1 = B1	A0 = B0	0	0	0	1	1	0

Le **X** peut prendre les valeurs : 0 ou 1

5. Mise en cascade de deux circuits intégrés 7485

Ce comparateur compare le nombre A formé des 8 bits A7 à A0 (dont A7 est le bit de poids le plus fort et A0 est le bit de poids le plus faible) et le nombre B formé des 8 bits B7 à B0 (dont B7 est le bit de poids le plus fort et B0 est le bit de poids le plus faible).

Le premier circuit (voir la figure suivante) compare les poids faibles de A avec le poids faibles de B. Le résultat de cette comparaison est transmis aux entrées A < B, A = B et A > B du deuxième circuit. Celui-ci compare les poids forts de A avec les poids forts de B et, en fonction du résultat de la comparaison des bits de poids faibles de A et B, indique sur ses sorties A > B, A = B et A < B le résultat de la comparaison des nombres A et B.



Mise en cascade de deux circuits intégré 7485

Chapitre 6 : Les bascules القلابات

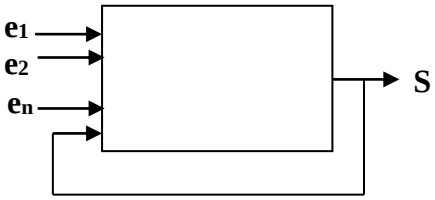
2019-2020

1. Introduction :

En logique combinatoire, la sortie ‘’S’’ dépend directement des entrées : e_1, e_2, e_3,…..e_n

Circuit logique combinatoire

En logique séquentielle, la sortie (ou les sorties) certes lié aux entrée, mais dépend aussi des états antérieurs.



Circuit séquentiel

2. les bascules (flip-flops) القلابات :

- Les bascules sont les circuits de base de la logique séquentielle.

- Chaque bascule possède la fonction de **mémorisation** et de **basculement**.

- Chaque bascule possède des entrées et deux sorties (complémentaires Q *et* $\overline{Q}$).

Il existe plusieurs types de bascules comme la bascule : RS, RSH, D, JK et T.

2.1. La bascule RS :

C'est une bascule à deux entrées R (reset : remise à zéro) et S (set :mise à un) tels que :

Si les deux entrées sont inactives alors la sortie $Q_+ = Q$ → **à l'instant précédent**

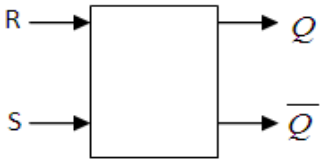
Si S est active seule $\Rightarrow Q_+ = 1$. → **à l'instant actuel**

Si R est active seule $\Rightarrow Q_+ = 0$.

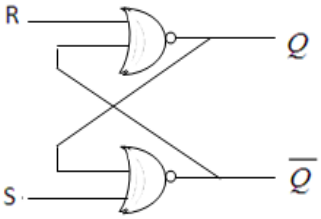
Un cas à éviter si les deux entrées sont actives à la fois, dans ce cas-là on obtient :

$\Rightarrow Q_+ = \overline{Q_+}$

a. Symbole :



b. logigramme :



c. Table de vérité :

R S	Q_+ $\overline{Q_+}$	La fonction
0 0	Q $\overline{Q}$	Mémorisation (sorties inchangées)
0 1	1 0	Set : mise à 1
1 0	0 1	Reset : remise à 0
1 1	* *	Interdite (indéterminé)

Pour comprendre bien cette bascule on utilise (étude) sa table de transition (ou excitation) c'est-à-dire les sortie en fonction des entrées)

d. Table de transition (excitation) :

Q Q_+	R S
0 0	x 0
0 1	0 1
1 0	1 0
1 1	0 x

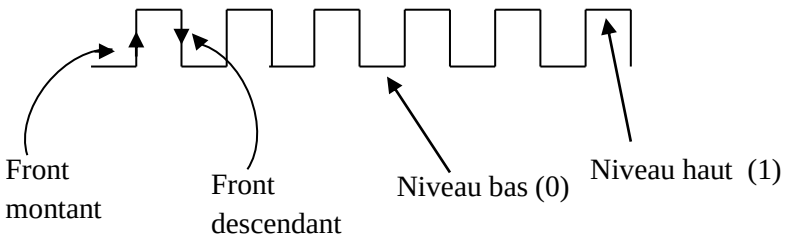
Où x, peut prend les valeurs : 0 ou 1.

2.2. La bascule RSH :

C'est une bascule RS synchronisé par un signal d'horloge H (niveau haut en général).

a. Système synchrone (notion d'horloge) :

Une horloge est une variable logique qui passe successivement de 0 à 1 et de 1 à 0 d'une façon périodique voir la figure suivante :



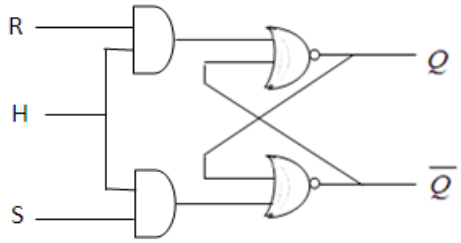
Cette variable (l'horloge) est utilisée souvent comme une entrée des circuits séquentiels, dans ce cas-là le circuit est dite synchrone.

- Lorsque un circuit séquentiel n'a pas d'horloge comme variable d'entrée ou si le circuit fonctionne indépendamment de cette horloge alors ce circuit est dite asynchrone.

- l'horloge est notée par 'h', 'H' ou 'CLK' (Clook)

b. La bascule RSH en niveau haut (H =1) :

b.1. Logigramme :



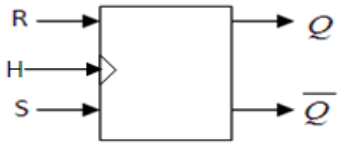
- lorsque H = 0, la bascule est dans l'état mémoire.

- lorsque H = 1, la bascule fonctionne comme une bascule RS.

b.2. Table de vérité :

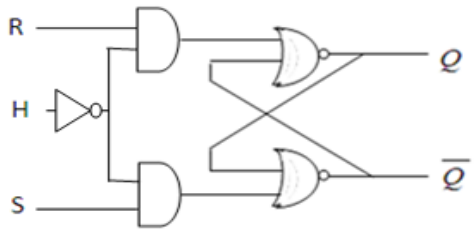
	H	R S	Q_+ $\overline{Q_+}$	La fonction
Mémoire	0	x x	Q $\overline{Q}$	Mémorisation (sorties inchangées)
Bascule RS	1	0 0	Q $\overline{Q}$	Mémorisation
	1	0 1	1 0	Set : mise à 1
	1	1 0	0 1	Reset : remise à 0
	1	1 1	* *	Interdite (indéterminé)

b.3. Symbole :



c. La bascule RSH en niveau bas (H = 0) :

c.1. logigramme :



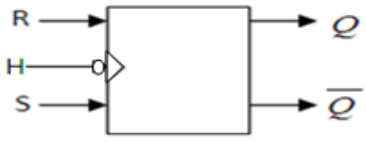
- lorsque H = 1, la bascule est dans l'état mémoire.

- lorsque H = 0, la bascule fonctionne comme une bascule RS.

c.2. Table de vérité :

	H	R S	Q_+ $\overline{Q_+}$	La fonction
Mémoire	1	x x	Q $\overline{Q}$	Mémorisation (sorties inchangées)
Bascule RS	0	0 0	Q $\overline{Q}$	Mémorisation
	0	0 1	1 0	Set : mise à 1
	0	1 0	0 1	Reset : remise à 0
	0	1 1	* *	Interdite (indéterminé)

c.3. Symbole :



Solution de TD N: 3

Exercice 2 :

On donne la fonction logique suivante :

$$f = \bar{a}bc + a\bar{c} + abc$$

Réaliser cette fonction à l'aide d'un multiplexeur MUX 8 à 1 (c'est-à-dire un MUX à 3 de sélection a, b et c)

Solution :

f est la sortie du multiplexeur (MUX 8 à 1) et a,b et c sont les entrées de sélection (ou les entrées d'adresses).

D'où la fonction générale de f :

$$f = \bar{c}\bar{b}\bar{a}x_0 + \bar{c}\bar{b}ax_1 + \bar{c}b\bar{a}x_2 + \bar{c}bax_3 + c\bar{b}\bar{a}x_4 + c\bar{b}ax_5 + cb\bar{a}x_6 + cbax_7 \quad (1)$$

Pour avoir : $f = \bar{a}bc + a\bar{c} + abc$

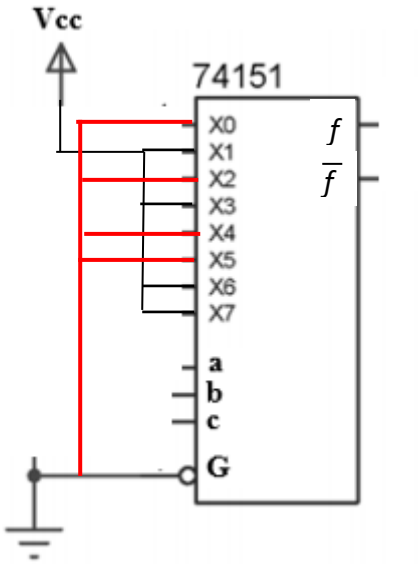
Il faut tout d'abord l'écrire sous sa forme canonique :

$$f = \bar{a}bc + a\bar{c}(b + \bar{b}) + abc = \bar{a}bc + ab\bar{c} + a\bar{b}\bar{c} + abc \quad (2)$$

Par identification entres les équations (1) et (2), on conclut que :

X_7, X_1, X_3, X_6 Doivent être à 1 (+Vcc) et X_0, X_2, X_4, X_5 doivent être à 0 (GND ou la masse).

Finalement, on obtient la solution (le schéma de MUX) suivante :



CHAPITRE III :

LES COMPTEURS

I. Introduction

Dans de nombreuses applications on est amené à faire des comptages d'impulsions dans un temps donné pour la mesure de fréquences (par exemple) ou tout simplement compter le nombre de fois où l'on opérera une certaine instruction . Dans certains cas il est nécessaire de compter dans d'autres il faut décompter à partir de zéro ou d'un nombre donné .on peut classer les compteurs suivant leur principe comme suit :

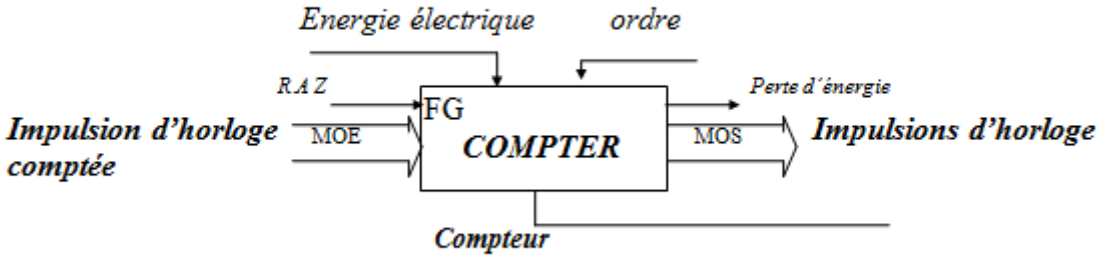
- Compteurs-décompteurs asynchrones
- Compteur-décompteurs synchrones

L'élément de base des compteurs est généralement une bascule à entrée d'horloge, soit de type bascule D, bascule JK ou bascule T

II. Généralités

C'est un dispositif destiné à *enregistrer* le résultat d'un *comptage d'impulsion*, soit pour lire directement ce résultat, soit pour *délivrer des signaux de commande* convenable.

Modélisation



III. Compteur asynchrone

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.

Avec n bascules on obtient 2^n combinaisons alors un compteur modulo 2^n.

1. Compteur modulo 16

Logigramme

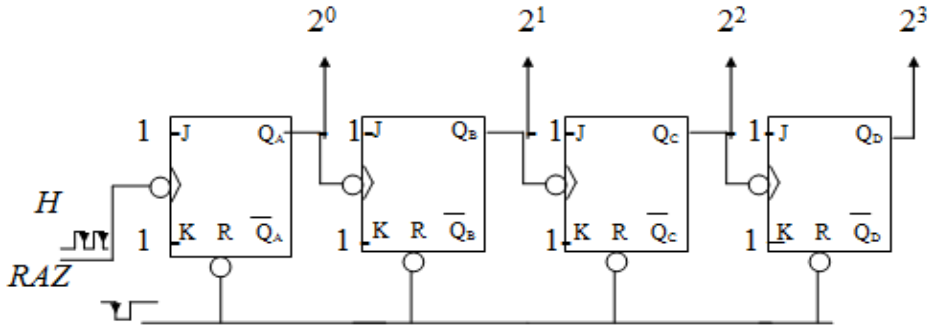
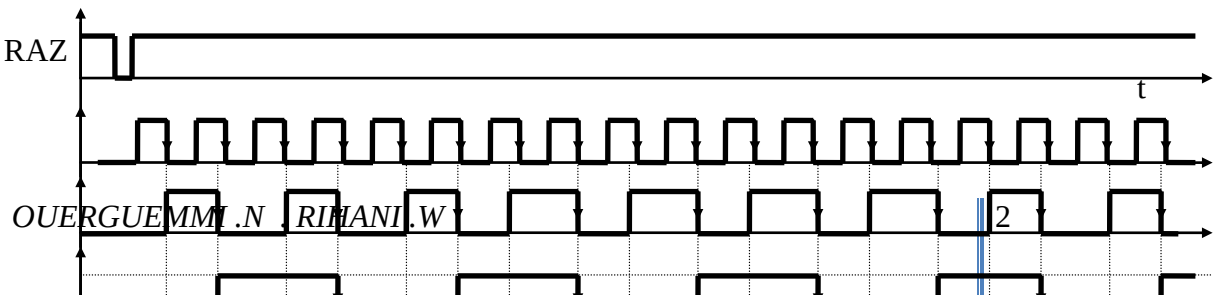


Figure1 : Compteur modulo 16

Table de séquences

N° IMPUL.	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Chronogramme



Q_A

t

Q_B

t

Q_C

Q_D

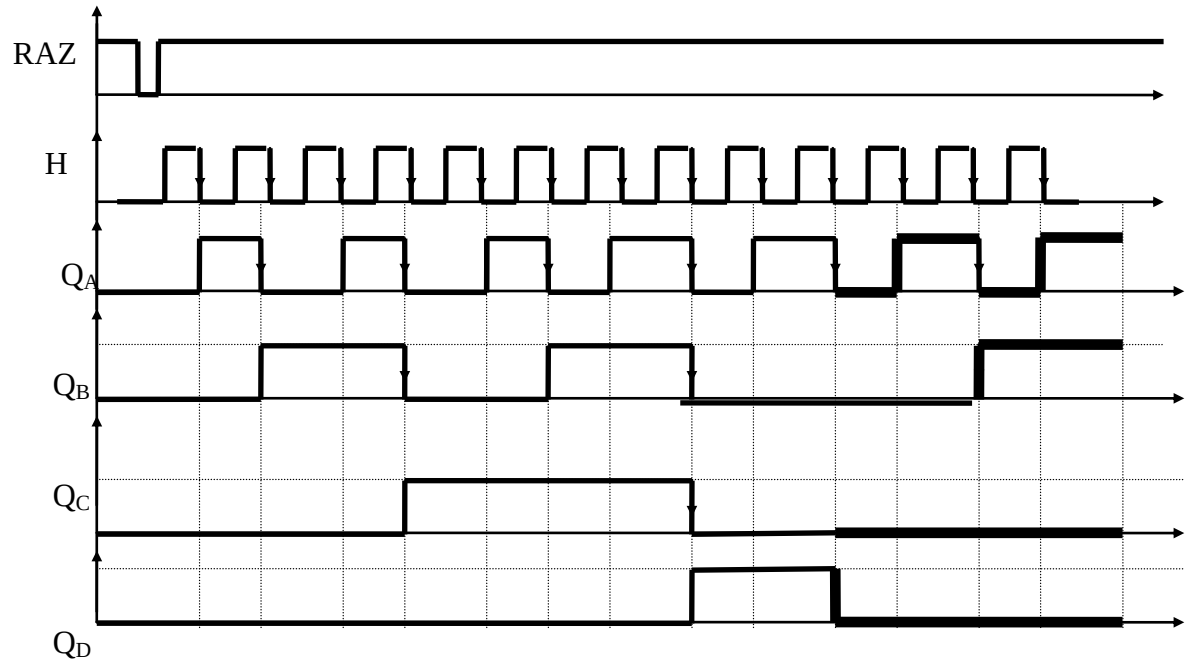
2. Compteur modulo 10 : (Avec *front descendant*)

On à $2^3 < 10 < 2^4$ donc il nous faut 4 bascules pour la réalisation de ce compteur modulo 10

Table de séquences

N° IMP	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Chronogramme



Logigramme

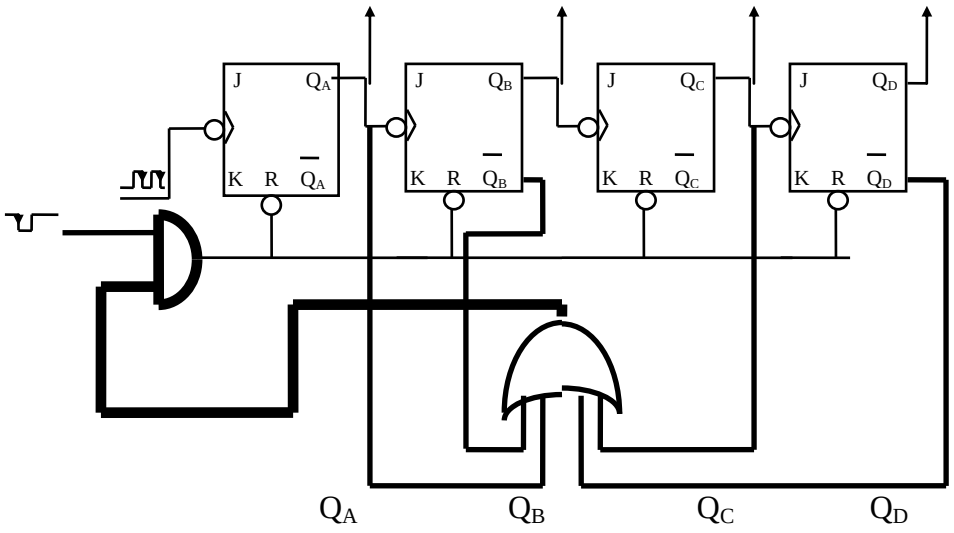


Figure2 : Compteur modulo 10

<u>Indication</u> :

La mise à zéro des bascules est obtenue :

- Soit par impulsion sur le RAZ.
- Soit lorsque le compteur devient égal à 9 c.-à-d. 1001 le passage suivant normalement est 10 c.à.d. 1010 alors il faut forcer cette dernière étape à zéro (**0000**) donc

$1 \rightarrow \mathbf{0}$ pour Q_D , $0 \rightarrow \mathbf{0}$ pour Q_C,

$1 \rightarrow \mathbf{0}$ pour Q_B S et $0 \rightarrow \mathbf{0}$ pour Q_A .

$\Rightarrow$ Grâce à une porte logique « OU » à 4 entrées lorsque :

$\overline{Q_D} + Q_C + \overline{Q_B} + Q_A = 0$ on aura obligatoirement la mise à zéro des 4 bascules donc la mise à zéro du compteur.

3. Décompteur asynchrone modulo 8 : (Avec *front montant*)

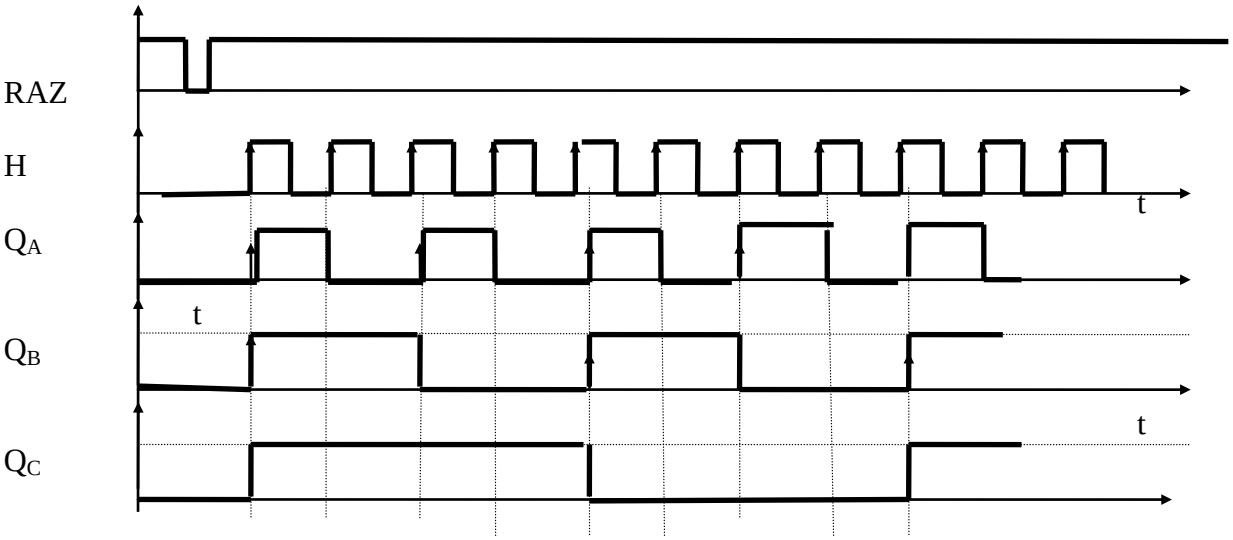
On à $2^3 = 8$ donc il nous faut 3 bascules pour la réalisation de ce décompteur asynchrone modulo 8.

Valeurs	Q_C	Q_B	Q_A
0	0	0	0
7	1	1	1

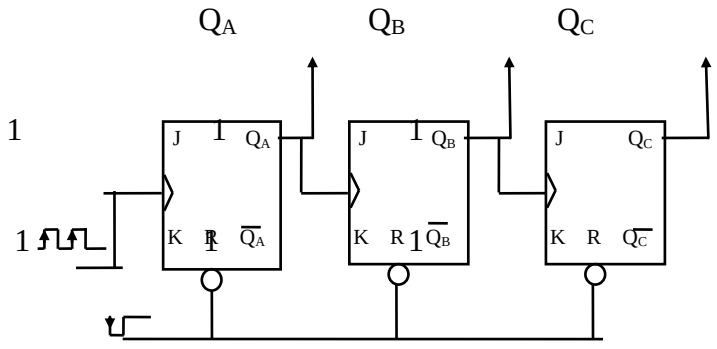
Table de séquences

6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0
7	1	1	1
6	1	1	0

Chronogramme



Logigramme



F*igure3:** ***Décompteur asynchrone modulo 8

Table de fonctionnement

Front	sortie de	Fonctionnement
↓	Q	Compteur
↓	$\overline{Q}$	Décompteur
↑	Q	Décompteur
↑	$\overline{Q}$	Compteur

IV. Compteur synchrone

Un compteur est dit synchrone lorsque les impulsions d'avancement sont envoyées simultanément sur les entrées d'horloge de toutes les bascules du compteur.

Toutes les bascules sont synchronisées sur le même signal d'horloge.

Un compteur synchrone modulo 2^n permet de compter de 0 à 2^n-1. Le nombre de bascule à utiliser est donc n.

Les bascules sont associées entre elles de telle manière que toutes les sorties Qi sont appliquées aux entrées J et K de la bascule i+1. (Une bascule doit avoir 1 sur les entrées J et K lorsque toutes les bascules précédentes prennent 1)

1. Compteur synchrone modulo 16

Logigramme

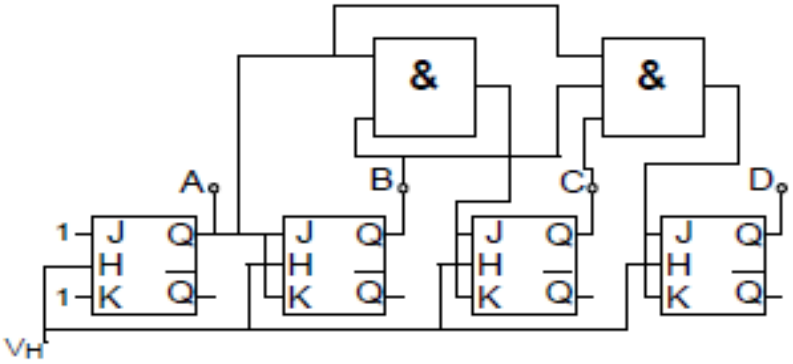
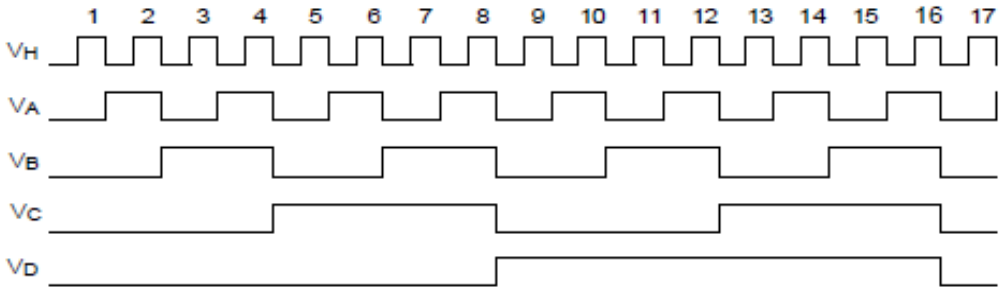


Figure4: Compteur synchrone modulo 16

Chronogramme



Etat initial : les sorties sont à zéro.

Après la seizième impulsion les sorties sont à nouveau à zéro.

2. Décompteur synchrone

Les bascules sont associées entre elles, de telle manière que toutes les sorties $\overline{Q_i}$ sont appliquées aux entrées J et K de la bascule i+1 (une bascule doit avoir 1 sur ces entrées J et K lorsque toutes les bascules précédentes prennent 0).

V. Synthèse des compteurs à l'aide des bascules

1. Tables de transitions des bascules

- Table des transitions réduite de bascule JK

	Qn	**Qn+1**	J	K
μ_0	0	0	0	ϕ
ε	0	1	1	ϕ
δ	1	0	ϕ	1
μ_1	1	1	ϕ	0

- Table des transitions réduite de bascule D

	Q_n	**Q_{n+1}**	D
μ_0	0	0	0
ε	0	1	1
δ	1	0	0
μ_1	1	1	1

2. Compteur synchrone modulo 5 avec bascule JK

- Table de comptage

	Etat n			**Etat n+1**		
	Q_C	**Q_B**	**Q_A**	**Q_C**	**Q_B**	**Q_A**
0	0	0	0	**0**	**0**	**1**
1	0	0	1	**0**	**1**	**0**
2	0	1	0	**0**	**1**	**1**
3	0	1	1		**0**	**0**
4	1	0	0	**0**	**0**	**0**

- Détermination des équations de J et K :

Pour Q_A				
Q_C \ Q_AQ_B	00	01	11	10
0	ε	ε	δ	δ
1	μ_0	-	-	-

Pour Q_B				
Q_C \ Q_AQ_B	00	01	11	10
0	μ_0	μ_1	δ	ε
1	μ_0	-	-	-

Pour Q_C				
Q_C \ Q_AQ_B	00	01	11	10
0	μ_0	μ_0	ε	μ_0
1	δ	-	-	-

J et K Pour Q_A

Q_C \ Q_AQ_B	00	01	11	10
0	1 ϕ	1 ϕ	ϕ 1	ϕ 1
1	0 ϕ	ϕ ϕ	ϕ ϕ	ϕ ϕ

J et K Pour Q_B

Q_C \ Q_AQ_B	00	01	11	10
0	0 ϕ	ϕ 0	ϕ 1	1 ϕ
1	0 ϕ	ϕ ϕ	ϕ ϕ	ϕ ϕ

J et K Pour Q_C

Q_C \ Q_AQ_B	00	01	11	10
0	0 ϕ	0 ϕ	1 ϕ	0 ϕ
1	ϕ 1	ϕ ϕ	ϕ ϕ	ϕ ϕ

$J_A = Q_C$ $K_A = 1$ $J_B = Q_A$ $K_B = Q_A$ $K_C = 1$ $J_C = Q_A\,Q_B$

- Schéma d'un compteur synchrone modulo 5 avec bascule JK :

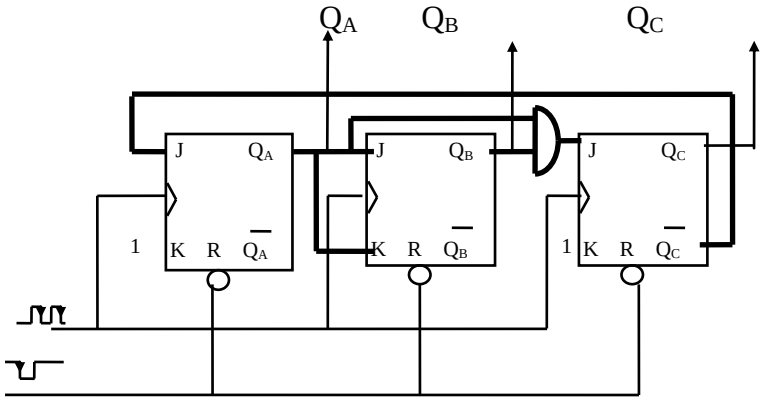


Figure5: Compteur synchrone modulo 5

• Indication :

Quand Q_C passe à 1 donc on aura $(100)_2 = (4)_{10}$ lorsque *le front arrive* il faut avoir (**000**) donc $Q_C = 0$ donc on met cette entrée directement à $J_A = 0$ et $K_A = 1$ est de cette façon on oblige Q_A à 0 ; $J_B = 0$ et $K_B = 0 \Rightarrow$ mémorisation à zéro et en même temps $J_C = 0$ avec $K_C = 1$ donc $Q_C = 0$, d'où on aura obligatoirement à la sortie la valeur (000) .

3. Décompteur synchrone modulo 7 avec bascule JK

- Table de comptage :

	état n			état n+1		
	Q_C	Q_B	Q_A	Q_C	Q_B	Q_A
6	1	1	0	**1**	**0**	**1**
5	1	0	1	**1**	**0**	**0**
4	1	0	0	**0**	**1**	**1**
3	0	1	1	**0**	**1**	**0**
2	0	1	0	**0**	**0**	**1**
1	0	0	1	**0**	**0**	**0**
0	0	0	0	**1**	**1**	**0**

- Détermination des équations de J et K :

Pour Q_A

Q_C \ Q_AQ_B	00	01	11	10
0	ε	ε	δ	δ
1	ε	ε	-	δ

Pour Q_B

Q_C \ Q_AQ_B	00	01	11	10
0	ε	δ	μ_1	μ_0
1	ε	δ		μ_0

Pour Q_C

Q_C \ Q_AQ_B	00	01	11	10
0	μ_0	μ_0	μ_0	μ_0
1	δ	μ_1		μ_1

J et K Pour Q_A

Q_C \ Q_AQ_B	00	01	11	10
0	1 ϕ	1 ϕ	ϕ 1	ϕ 1
1	1 ϕ	1 ϕ	ϕ 1	ϕ 1

J et K Pour Q_B

Q_C \ Q_AQ_B	00	01	11	10
0	1 ϕ	ϕ 1	ϕ 0	0 ϕ
1	1 ϕ	ϕ 1	ϕ ϕ	0 ϕ

J et K Pour Q_C

Q_C \ Q_AQ_B	00	01	11	10
0	0 ϕ	0 ϕ	0 ϕ	0 ϕ
1	ϕ 1	ϕ 0	ϕ ϕ	ϕ 0

$J_A = Q_B + Q_C$ $\quad$ $J_B = Q_A$ $\quad$ $J_C = Q_A\, Q_B$

$K_A = 1$ $\quad$ $K_B = Q_A$ $\quad$ $K_C = Q_A \,.\, Q_B$

VI. Contexte d'utilisation des compteurs

On tient compte des caractéristiques technologiques des bascules:

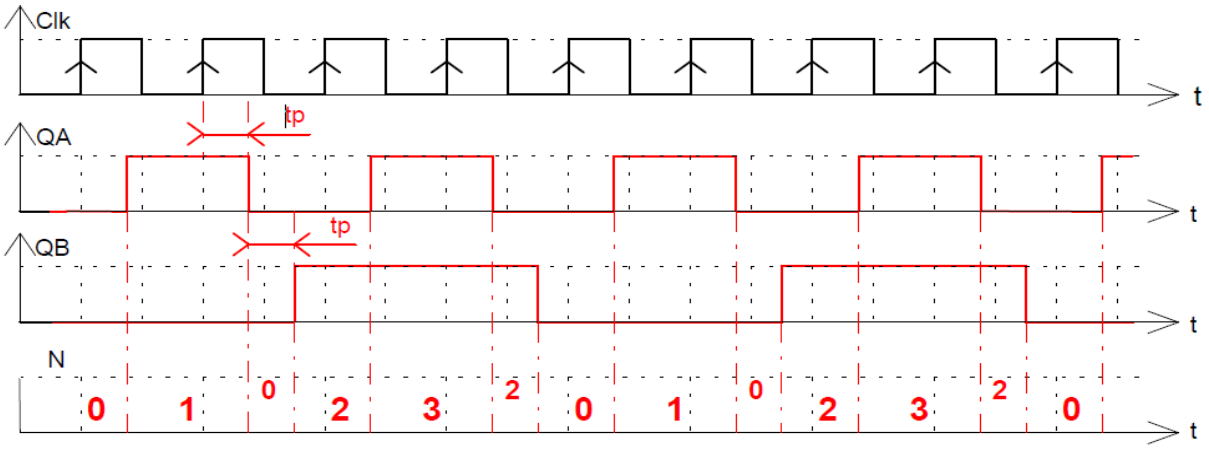
- La sortie Q de chaque bascule est mise à jour après un temps t_P après le front actif.

On suppose travailler à fréquence élevée, donc période petite, jusqu'à $T \gtrapprox t_P$

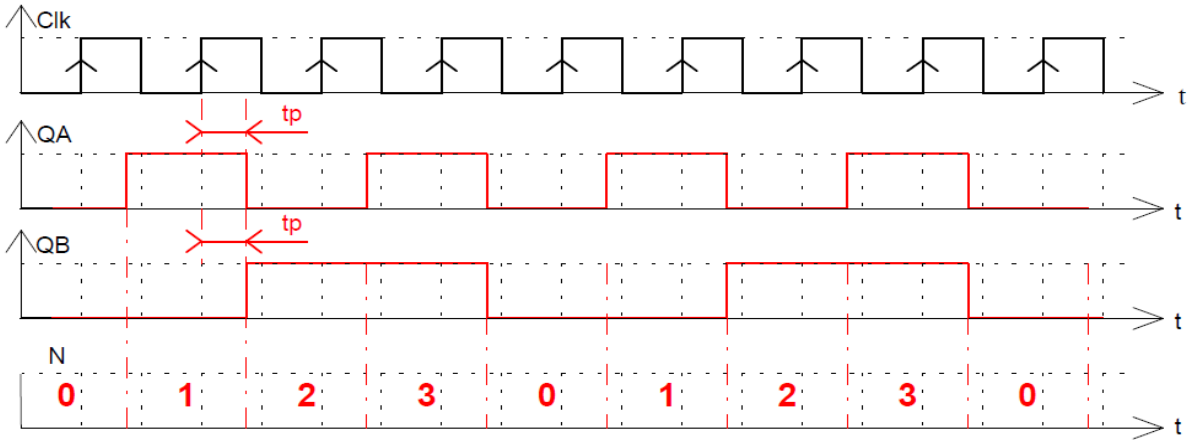
On prendra par exemple $T=8/3^* t_P$ donc $t_P = 3/8 *T$

(Sit_P = 300 ns, on choisit T = 800 ns donc f = 1,25 MHz pour mettre les défauts en évidence)

1. Compteur asynchrone



2. Compteur synchrone



3. Conclusion

Tableau : fonctions attribuées au composant Compteur

Utilisation	Compteur asynchrone	Compteur synchrone
Basse fréquence	Comptage Division de fréquence	Comptage Division de fréquence
Haute fréquence	Division de fréquence	Comptage Division de fréquence

Chapitre3

Les registres

Objectifs :

- ✓ **Connaitre les différents types de registres.**
- ✓ **Comprendre la méthode de synthèse d'un registre binaire.**
- ✓ **Apprendre à analyser le fonctionnement d'un circuit de registre.**

1. Introduction

Un registre est un ensemble de mémoires élémentaires (bascules), synchronisées par le même signal d'horloge.

Les registres sont largement utilisés dans les systèmes de traitement numérique (les ordinateurs par exemple) pour réaliser des opérations : de mémorisation provisoire (mémoire-tampon), de décalage, de rotation, …

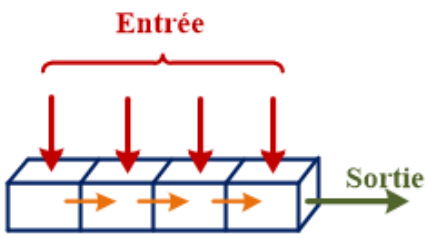
2. Caractéristiques d'un registre

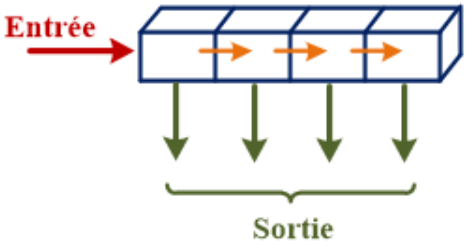
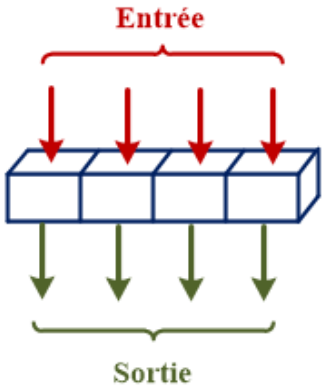
Tout registre est caractérisé par :

- **La capacité**: nombre de bits du mot binaire qu'il peut mémoriser.
- **Le mode d'écriture ou de chargement**: dépend du nombre d'entrées :
  - écriture série : génération bit par bit, avec transmission par un seul fil conducteur.
  - écriture parallèle : génération globale du mot de **n** bits, avec transmission par un bus de **n** bits (**n** fils conducteurs).
- **Le mode de lecture**:
  - lecture série : exploitation bit par bit du mot (une seule sortie).
  - lecture parallèle : exploitation globale du mot (**n** sorties).

3. Différents types de registre

Selon le mode d'accès en écriture (entrée) et en lecture (sortie), série ou parallèle, Il existe quatre types de registre :

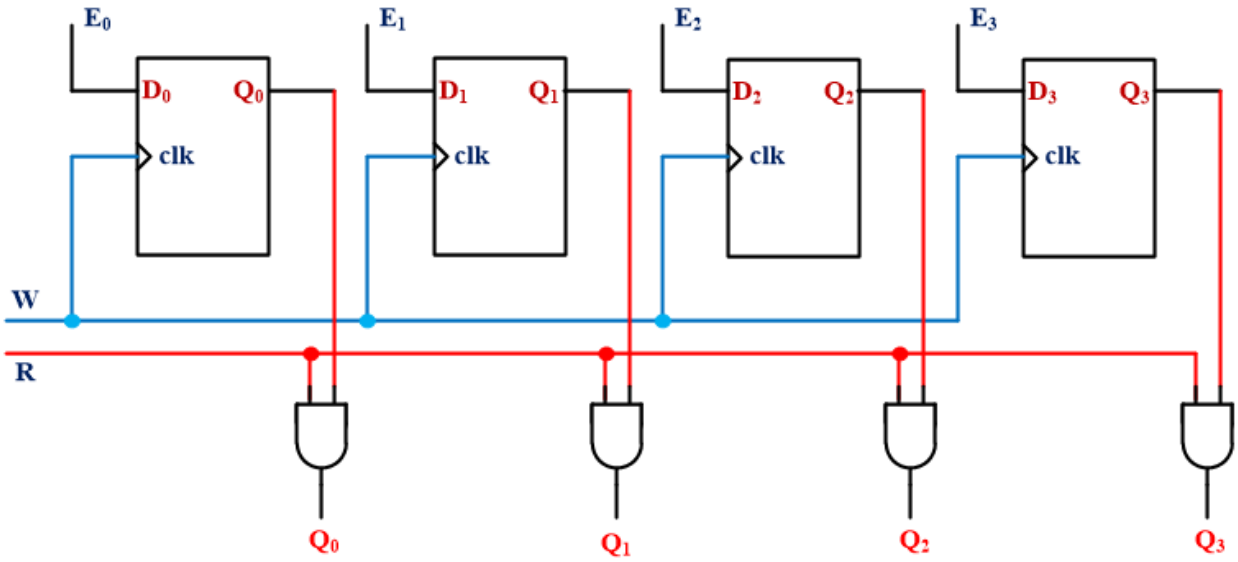
Entrée série / Sortie série **S**erial**IN** - **S**erial**OUT**(**SISO**)	**Entrée parallèle / Sortie série** **P**arallel**IN** - Serial **OUT** (**PISO**)
	

Entrée série / Sortie parallèle SerialIN - ParallelOUT (SIPO)	Entrée parallèle / Sortie parallèle ParallelIN - ParallelOUT (PIPO)
	

Ces quatre types peuvent être classés en deux catégories : *les* ***registres de mémorisation*** (tampon) et *les* ***registres à décalage***.

3.1 Registre de mémorisation

Ce registre permet la mémorisation de n bits. Il est donc constitué de n bascules, mémorisant chacune un bit. L'information sur n bits est chargée (écrite) au moyen d'un signal de commande (**W**) qui est le signal d'horloge des bascules puis elle est conservée et devient disponible en lecture. La figure suivante donne un exemple de registre de mémorisation 4 bits réalisé à base de bascules D.



Le registre mémorise les états des entrées E_0, E_1, E_2et E_3en synchronisme avec le signal d'écriture **W**. Ces états sont conservés jusqu'au prochain signal de commande **W**. Dans cet exemple les données mémorisés peuvent être lus sur les sorties Q_0, Q_1, Q_2etQ_3au moyen du signal de validation **R**. On remarque que ce registre est du type **PIPO**.

3.2 Registres à décalage

Dans un registre à décalage les bascules sont interconnectées de façon à ce que l'état logique de la sortie de la bascule de rang (i) puisse être transmis à la bascule de rang (i+1) lorsqu'un signal d'horloge est appliqué à l'ensemble des bascules. L'information peut être chargée en série (les n bits sont chargés l'un après l'autre) ou en parallèle (les n bits sont chargés simultanément).

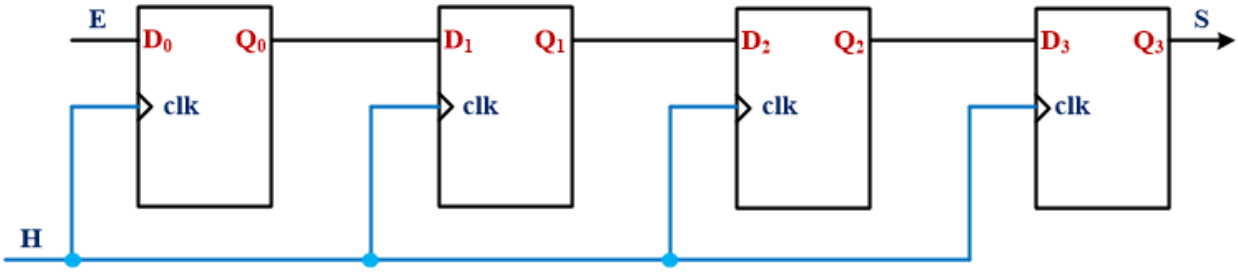
3.2.1 Registre à décalage entrée série -sortie série

Les bits d'information sont présentés séquentiellement bit après bit à l'entrée de la première bascule et se propagent à travers le registre à chaque impulsion du signal d'horloge, pour sortir par la dernière bascule.

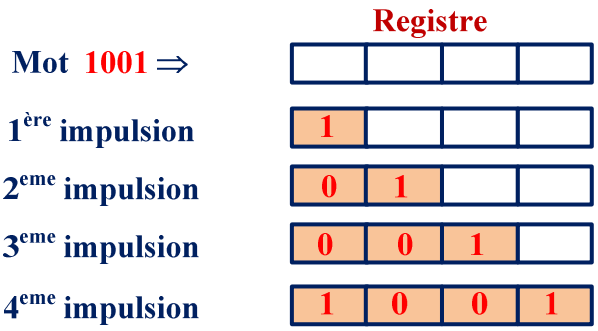
- **Décalage à droite**

Ci-dessous le circuit d'un registre à décalage à droite entrée série-sortie série 4 bits à base de bascules D.

$$D_0=ES\ ;\ D_i=Q_{i-1}\ (i \geq 1)$$



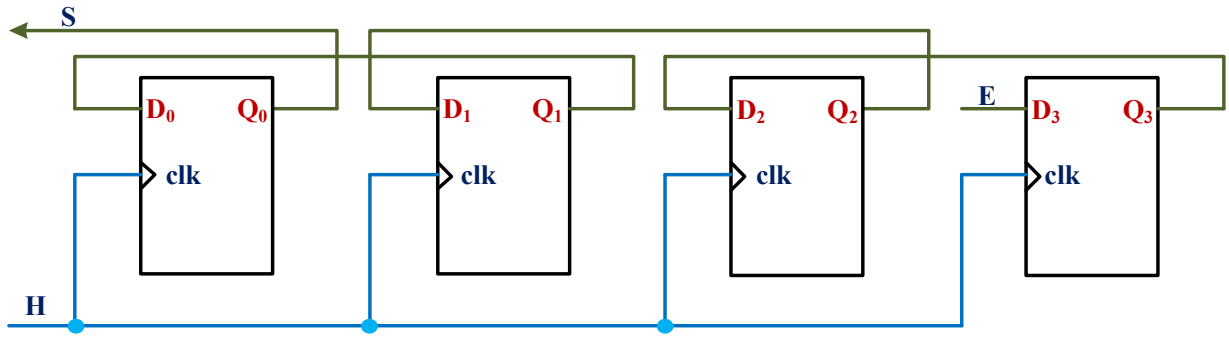
Exemple : Chargement du mot binaire 1001 :



- **Décalage à gauche**

Dans ce cas l'entrée de la bascule D de rang i doit être connectée à la sortie de la bascule de rang i+1.

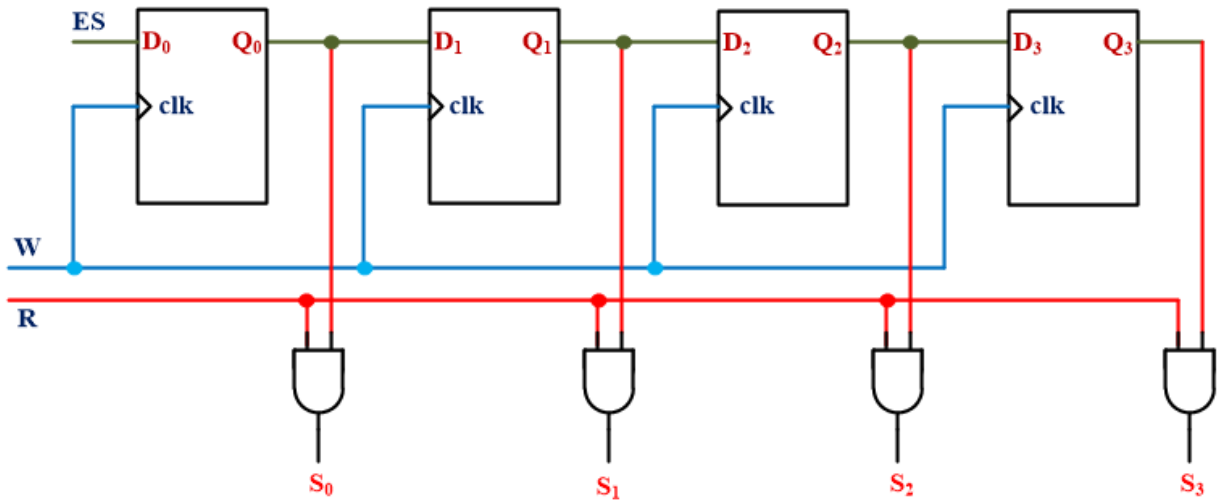
$$D_{n-1}=ES\ ;\ D_i=Q_{i+1}\ (i < n-1)$$



3.2.2 Registre à décalage entrée série - sortie parallèle

Ce type de registre permet de transformer un codage temporel (succession des bits dans le temps) en un codage spatial (information stockée en mémoire statique).

La figure suivante donne un exemple de registre de 4 bits à entrée série et sortie parallèle réalisé avec des bascules D. Le signal R (Read) n'est pas obligatoire, il permet juste de commander la lecture des sorties en mêmes temps, de façon à éviter la lecture au moment du chargement.



$$\begin{cases} D_0 = ES \\ D_i = Q_{i-1} \end{cases}$$

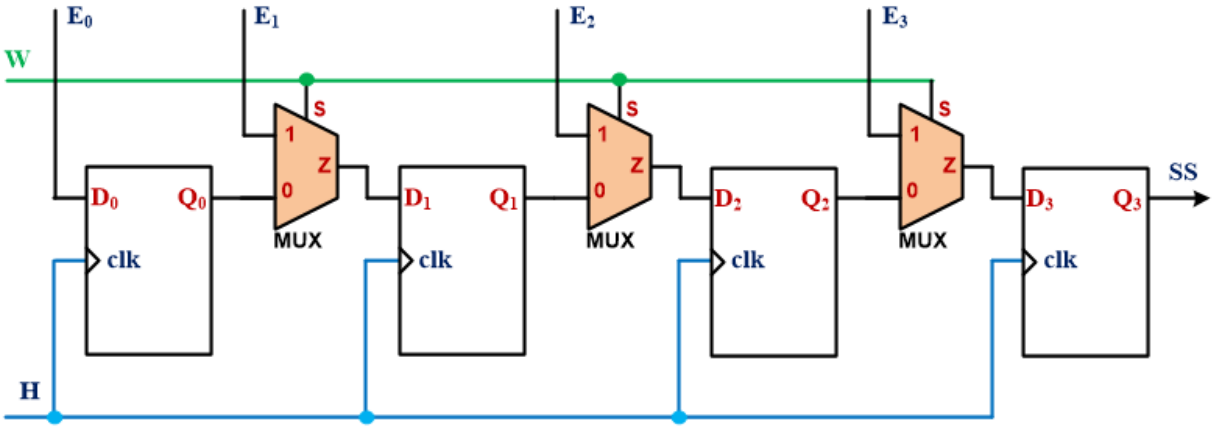
3.2.3 Registre à décalage entrée parallèle - sortie série

Le chargement parallèle des données peut s'effectuer de deux manière : synchrone ou asynchrone.

- **Chargement synchrone**

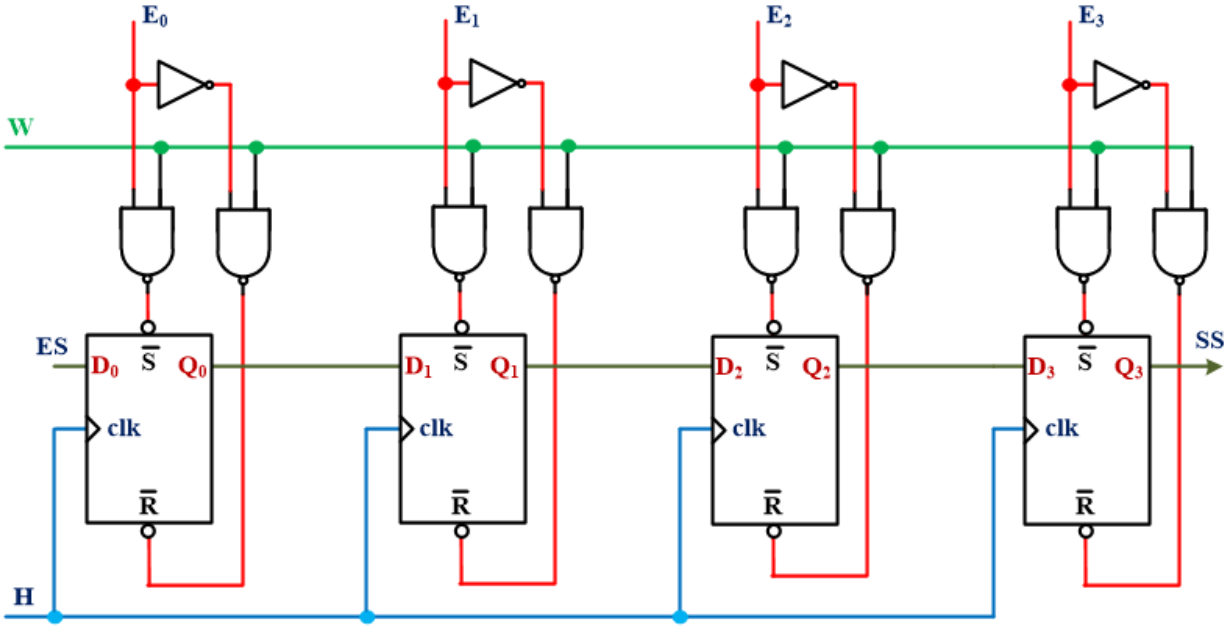
Dans ce cas il faut appliquer les données aux entrées synchrones Di. En fonction de l'ordre de chargement (écriture) W, chaque bascule recopie l'entrée Ei ou bien la sortie de la bascule i-1, à chaque front d'horloge. Il faut donc utiliser un multiplexeur 2 vers 1 à l'entrée Di de chacune des bascules 1 à n-1 comme le montre le circuit ci-dessous qui représente un registre 4 bits.

$$\begin{cases} D_0 = E_0 \\ D_i = Z_i = Q_{i-1}\overline{W} + E_i W \end{cases}$$



- **Chargement asynchrone**

On utilise ici les entrées asynchrones $\bar{R}$ et $\bar{S}$ pour forcer chaque bascule à **0** ou à **1** indépendamment du signal d'horloge. Les entrées synchrones sont utilisées pour la propagation des données à travers le registre comme l'indique la figure suivante.



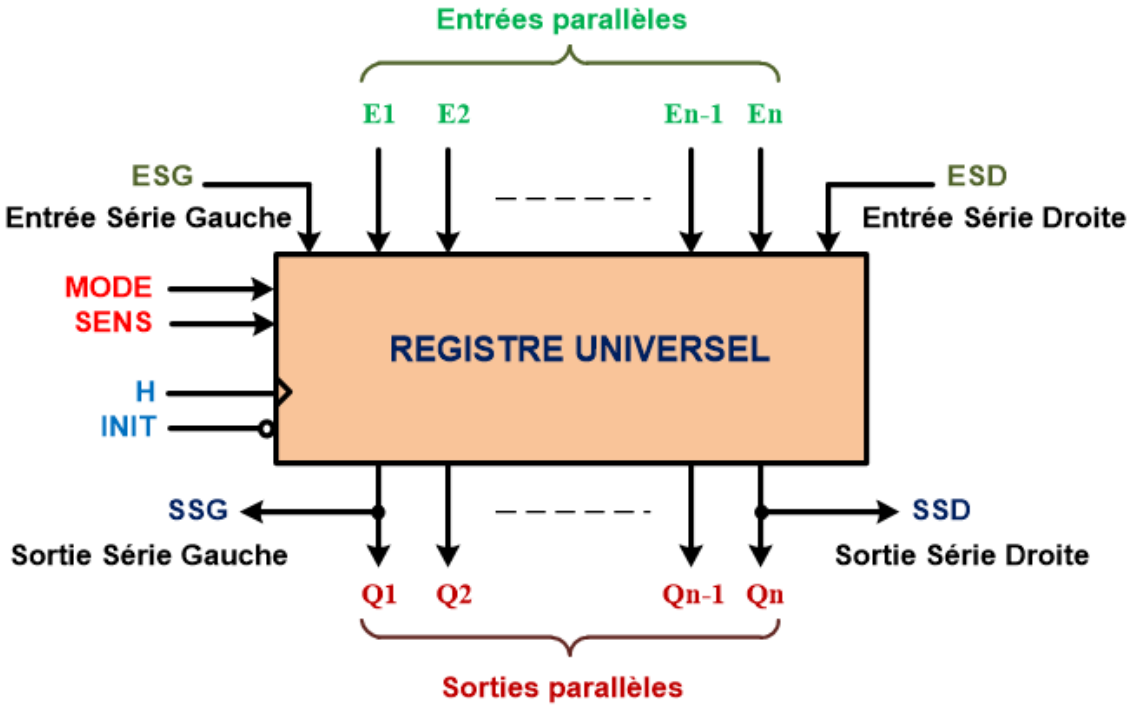
- Si **W=0** on a : $\bar{S}_i = \bar{R}_i = 1$: fonctionnement normal des bascules (décalage à droite des données à travers le registre).

- Si **W=1** on a :

$$\begin{cases} Ei = 0 \Rightarrow \bar{S}_i = 1 et \bar{R}_i = 0 \Rightarrow Qi = 0 \\ Ei = 1 \Rightarrow \bar{S}_i = 0 et \bar{R}_i = 1 \Rightarrow Qi = 1 \end{cases} \Rightarrow Qi = Ei$$

3.3 Registre universel

C'est un registre qui effectue un chargement des données série ou parallèle et un décalage à gauche ou droite avec une lecture série ou parallèle. Il dispose d'entrées de mode de fonctionnement qui définissent le type de chargement et le sens de décalage.La figue ci-dessous représente la configuration d'un tel registre.



- L'entrée **MODE** permet de choisir le mode de chargement série ou parallèle.
- L'entrée **SENS** permet de choisir le sens de décalage à gauche ou à droite.
- L'entrée **INIT** permet dinitialiser le registre.

Ce type de registre existe sous forme de circuit intégré qui assure toute les fonctions indiquées sur la figure ci-dessus (le circuit **74194** par exemple).

4. Applications des registres

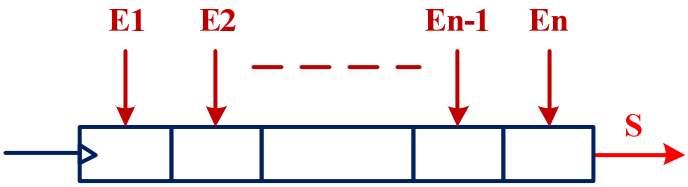
Dans ce qui suit nous citons quelques applications des registres.

4.1 Mémorisation temporaire d'une information

Les registres sont utilisés dans les microprocesseurs pour des mémorisations temporaires. En effet chaque registre mémorise temporairement un mot de **n** bits en attendant son traitement.

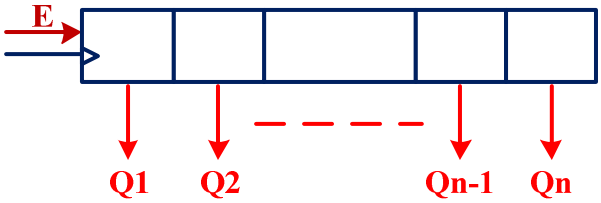
4.2 Conversion parallèle-série de mots binaires

Le mot binaire sur n bits est chargé en parallèle dans le registre puis des opérations de décalage successives permettent de le convertir en série.



4.3 Conversion série-parallèle d'untrain binaire

Un train binaire est lit en série et décalé puis récupéré sous forme binaire sur les sorties Q1 à Qn.



4.4 Ligne à retard numérique

Dans ce cas le registre permet de retarder un train binaire de n périodes du signal d'horloge.

4.5 Division et multiplication par 2^n

- ***Décalage à droite de n bits* : *division par 2^n***

	0	1	1	1	0	0	28 état initial
	↘	↘	↘	↘	↘		
0 →	0	0	1	1	1	0	14 après 1 front de H
	↘	↘	↘	↘	↘		
0 →	0	0	0	1	1	1	7 après 2 fronts de H

$\frac{28}{2^2} = 7$: division par **4** soit 2^2 après **2** impulsions d'horloge.

- ***Décalage à gauche de n bits : multiplication par 2^n***

0	0	0	1	1	1		7 état initial
0	0	1	1	1	0	←0	14 après 1 front de H
0	1	1	1	0	0	←0	28 après 2 fronts de H

$7 \times 2^2 = 28$: multiplication par **4** soit 2^2 après **2** impulsions d'horloge.

5. Exercice d'application

Réaliser, à base de bascules **D**, un registre à décalage à droite et à gauche **4** bits.