Chapitre 3 : Simplification des Fonctions Logiques

III.1. Introduction

Puisque la réalisation d'une fonction logique même aussi simple qu'un ou exclusif n'est pas unique. Il est souvent souhaitable pour des raisons d'optimisation de disposer de sa forme minimale autrement dit de celle qui permet de l'implémenter avec un minimum de composants.

Il est donc nécessaire de disposer d'outils permettant de déterminer la forme minimale d'une fonction logique à partir de ses formes. On parlera dans ce cas de simplification ou minimisation de fonction.

Les fonctions logiques peuvent être simplifiées de deux façons différentes.

- La première fondée sur l'application des lois et des théorèmes de l'algèbre booléenne.
- La 2ème est une méthode graphique qui suit une démarche systématique.

III.2. Simplification algébrique

Cette technique de simplification repose sur l'utilisation desthéorèmes fondamentaux et des propriétés de l'algèbre de Boole. Après la recherche de l'expression algébrique de la fonction, l'étape suivante consiste à minimiser le nombre de termes dans une fonction afin d'obtenir un circuit plus petit donc plus facile à construire avec un coût plus réduit.

La simplification algébrique repose sur beaucoup d'astuce, quand la fonction est plus complexe (au-delà de trois variables) cette méthode de simplification devient peu évidente. On peut**[18]** :

- Supprimer les associations de termes multiples.
- Mettre en facteur des variablespour éliminerplusieurs termes.
- Mettre en facteur des variables pour faire apparaître des termes inclus.
- Ajouter unterme qui existe déjà à une expression logique.

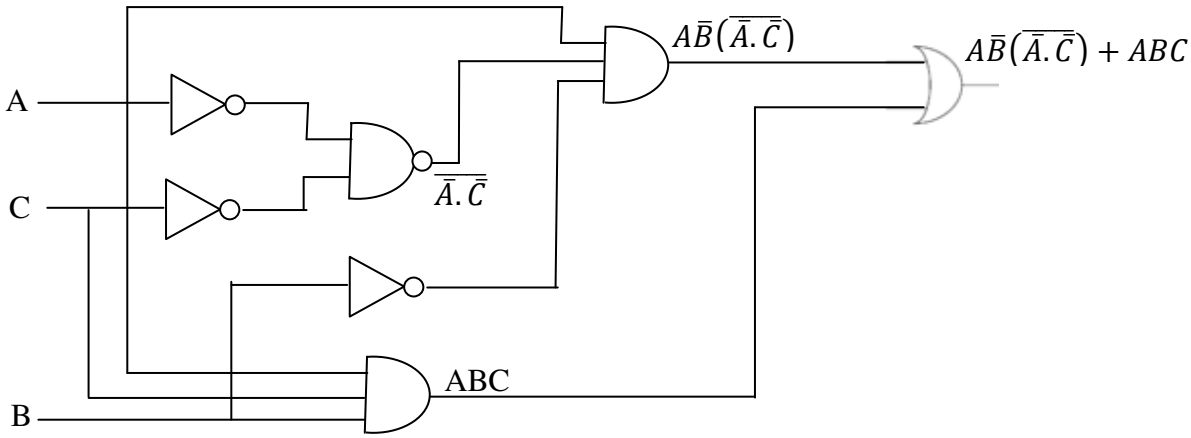
Technique de base :

1. La transformation **:** Par applications successives des théorèmes de Morgan et par multiplications des termes de l'expression pour obtenir une somme de produit.

( ) + ( ) + ( )

2. La vérification de chaque produit pour trouver les variables communes puis la mise en facteur de ses derniers, ceci permet d'éliminer plusieurs termes.

Exemple 1 : Simplifier par la méthode d'algébrique le circuit logique suivant :



1ère méthode : $f = ABC + A\bar{B}(\overline{\bar{A}.\bar{C}}) = ABC + A\bar{B}(A + C)$

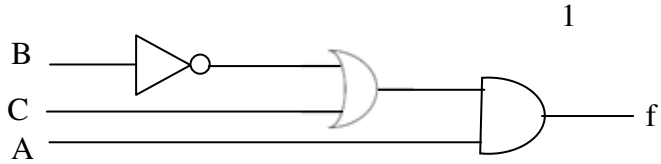
$= ABC + A\bar{B} + A\bar{B}C$

$= AC(\underbrace{B + \bar{B}}_{1}) + A\bar{B}$

$= AC + A\bar{B} = A(C + \bar{B})$

2ème méthode : $f = A(\underbrace{\bar{B} + BC}_{\bar{B} + C}) + A\bar{B}C$

$= A\bar{B} + AC + A\bar{B}C = A\bar{B}(1 + C) + \underbrace{AC}_{1} = A(\bar{B} + C)$



Exemple 2 : Simplifier l'expression suivante :

1ère méthode : $f = ABC + AB\bar{C} + A\bar{B}C = AB(\overbrace{C + \bar{C}}^{1}) + A\bar{B}C$

$= AB + A\bar{B}C$

$= A(B + \bar{B}C)$

$\underbrace{\quad}_{B + C}$

$= A(B + C)$

2ème méthode : $f = ABC + AB\bar{C} + ABC + A\bar{B}C$

$= AB(C + \bar{C}) + AC(B + \underbrace{\bar{B}}_{1})$

$= AB + AC$

$= A(B + C)$

III.3. Simplification par la méthode de Karnaugh

La méthode de Karnaugh consiste à présenter les combinaisons des variables d'entrées (les termes) d'une fonction logique : Chaque case du tableau de karnaugh correspond à une ligne de la table de vérité.

Le tableau de Karnaugh est en fait un tableau circulaire à double entrées qui utilise le code GRAY. La première entrée horizontale est en fait la suite de la dernière entrée verticale.

Il est possible de simplifier l'expression de Fen regroupant selon des règles précises les cases adjacentesdu tableau de Karnaugh qui comportent des 1.On donne à ce processus de combinaisons le nom de *réunion*[**19**].

III.3.1. Réunion de pair (doublet)

Le regroupement de 2 cases adjacentes contenant des '1' (doublet) dansuntableaudeKarnaughéliminelavariablequiestàlafoiscomplémentéeetnoncomplémentée.

Soit les tableaux de Karnaugh suivants :

C \ BA	**00**	**01**	**11**	**10**
0	0	1	1	0
1	0	0	0	0

(a)

C \ BA	**00**	**01**	**11**	**10**
0	0	**1**	0	0
1	0	**1**	0	0

(b)

C \ BA	**00**	**01**	**11**	**10**
0	0	0	0	0
1	1	0	0	1

(c)

Figure III.1 : Tableaux de Karnaugh avec une réunion d'un doublet de 1 adjacents.

(a) : $A\bar{B}\bar{C} + AB\bar{C} = A\bar{C}(A + \bar{B}) = A\bar{C}$

Dans le tableau (a), il y a deux 1 qui sont voisin, le premier correspond à$A\bar{B}\bar{C}$ et le deuxième à $AB\bar{C} \rightarrow$ seule B change d'état. Ces deux termes peuvent être réunis, le résultat éliminé B.

(b) : $G = A\bar{B}$ $(A\bar{B}\bar{C} + A\bar{B}C)$

(c) : $f = \bar{A}\bar{B}C + \bar{A}BC = \bar{A}C$

Il existe deux « 1 » dans la colonne à droite et dans la colonne à gauche sur la même ligne,ces deux « 1 » sont également adjacents et leursréunionséliminent la variable B.

Exemple : Simplifier la fonction définie par le tableau de Karnaugk suivant :

DC \ BA	**00**	**01**	**11**	**10**
00	0	0	1	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

Simplification algébrique :

$$F = AB\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D$$
$$= B\bar{C}\bar{D}(A + \bar{A}) + \bar{A}\bar{C}D(B + \bar{B})$$
$$= B\bar{C}\bar{D} + \bar{A}\bar{C}D$$

Simplification par le tableau de karnaugh :

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

III.3.2. Réunion de quartet (groupe de 4)

Le regroupement de 4 cases adjacentes contenant des '1' (quartet) dansun tableau deKarnaughéliminelesdeuxvariablesquisontàlafoiscomplémentéesetnoncomplémentées.

Soit les tableaux de Karnaugk suivants :

C \ BA	00	01	11	10
0	1	1	1	1
1	0	0	0	0

(a)

DC \ BA	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	1	0	0
10	0	1	0	0

(b)

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

(c)

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	0	0	1
10	1	0	0	1

(d)

DC \ BA	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

(e)

Figure III.2 : Tableaux de Karnaugh avec une réunion d'un quartet de 1 adjacents.

(a):$F = \bar{C}$

(b):$F = A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}D$

$$= A\bar{B}\bar{D}(\bar{C} + C) + A\bar{B}D(C + \bar{C})$$
$$= A\bar{B}\bar{D} + A\bar{B}D = A\bar{B}(\bar{D} + D)$$
$$= A\bar{B}$$

(c): $F = AC$

(d): $F = \bar{A}D$

(e): $F = \bar{A}\bar{C}$

III.3.3. Réunion d'octet (groupe de 8)

Le regroupement de 8 cases adjacentes contenant des '1' (octet)dansun tableau deKarnaughéliminelestroisvariablesquisontàlafoiscomplémentéesetnoncomplémentées.

Soit les tableaux de Karnaugk suivants :

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

(a)

DC \ BA	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	1	0	0
10	1	1	0	0

(b)

DC \ BA	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

(c)

DC \ BA	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

(d)

Figure III.3 : Tableaux de Karnaugh avec une réunion d'un octet de 1 adjacents.

(a) :$F = C\bar{D} + CD = C(\bar{D} + D) = C$
(b): $F = \bar{A}\bar{B} + A\bar{B} = \bar{B}$
(c): $F = \bar{C}$
(d):$F = \bar{A}$

III.3.4. Processus de simplification au complet[12], [15], [18], [20]

- Le nombre de cases d'un groupement doit être égal à 1, 2, 4, …2n.
- faire le minimum des regroupements qui contient le maximum des cases
- Faire les regroupements de taille maximale, de manière à éliminer le plus grand nombre possible de variables dans les termes de l'expression
- Un bit à 1 peut appartenir à plusieurs groupements
- Dans chaque groupement on ne retient que les termes dont l'état des variables ne change pas à l'intérieur d'un regroupement.
- L'expression de la fonction logique simplifiée est égale à la somme logique (OU logique) de tousles termes trouvés dans chaque groupement(doublet, quartet ou octet).

Exemple: Soit les tableaux de Karnaugh suivants. Donner la fonction simplifiée de F à partir de ces tableaux.

DC \ BA	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	1	1	0	0
10	0	0	0	0

$$F = \bar{B}C + C\bar{D} + AB\bar{D}$$

DC \ BA	00	01	11	10
00	0	1	0	0
01	0	1	1	1
11	1	1	1	0
10	0	0	1	0

$$F = A\bar{B}\bar{D} + BC\bar{D} + \bar{B}CD + ABD$$

DC \ BA	00	01	11	10
00	0	0	0	1
01	1	1	1	1
11	0	0	0	0
10	0	1	0	0

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

DC \ BA	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	0	0	1
10	1	0	0	1

$$F = \bar{A}\bar{C} + \bar{A}B + AC\bar{D}$$

III.4. Conditions indifférentes

Parfois, la fonction à réaliser n'est pas complètement définie, c'est-à-dire que la valeur logique de la fonction pour certaines combinaisons des variables n'a pas d'importance. Ceci pour plusieurs raisons :

- Soit qu'on est indifférent à la valeur qu'elle peut prendre pour certaines combinaisons des variables.
- Soit que ces combinaisons de valeurs de variables sont physiquement où pratiquement impossibles.

Les cases du tableau de Karnaugh correspondantes à ces combinaisons sont appelées des cases ∅.

Pour trouver l'expression la plus simple de la fonction logique, on pourra utiliser ces cases ∅, selon le cas soit comme des cases 1, soit comme des cases 0. Il n'y a pas de règle permettant un choix judicieux, mais en général on attribuera à chaque case ∅ la valeur (0 ou 1) qui permet d'obtenir le plus petit nombre de groupements de plus grande taille possible qui englobe toutes les cases**[10]**.

Exemple 1 : Simplifier la fonction de F donnée par la T.V suivante.

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

Conditions indifférents

T.K :

C \ BA	**00**	**01**	**11**	**10**
0	0	0	1	ϕ
1	0	ϕ	1	1

L'expression de sortie la plus simple est : $F = B$

Exemple 2 : Simplifier la fonction de F :

DC \ BA	**00**	**01**	**11**	**10**
00	1	1	1	ϕ
01	1	1	0	0
11	0	0	ϕ	ϕ
10	1	ϕ	1	ϕ

$$F = \bar{C} + \bar{B}\bar{D}$$

Conclusion : La méthodede simplification graphique des fonctions logiques est basée sur les tableaux de Karnaugh. Cette méthode simple et efficace conduit à une solution minimale, mais elle devient inutilisable lorsque les dimensions de la table sont trop importantes (au-delà de 4 variables d'entrées). Dans ce cas, il faut utiliser des méthodes plus élaborées, comme celle de Quine Mc Cluskey.Programmable sur un certain nombre de logiciels spécialisés (Espresso, Mc Boole), elle est très efficace et peut proposer toutes les solutions minimales existantes. **[12]**.

III.5. Méthode de Quine-Mc Cluskey

Cette méthode est basée sur :

1. La recherche des intersections premières.
2. La recherche de la réunion minimale.

La méthode de Quine-Mc Cluskey est utilisée lorsque la fonction est représentée par la première forme canonique ou chaque intersection contient toutes les variables de la fonction. Elle consiste tout d'abord à classer les intersections canoniques monômes selon leurs poids.

Définition 1 : On appelle poids d'un monôme complet, le poids de son équivalent binaire.

Définition 2 : Equivalent binaire : le nombre binaire obtenuen remplaçant toutes les variables $\bar{x_i}\ par\ 0\ et\ x_i\ par\ 1$.

Définition 3 : On appelle poids d'un nombre binaire le nombre de 1 contenu dans ce nombre où le nombre de lettre non barrée dans ce monôme.

Exemple : monôme complet **:** $DC\bar{B}\bar{A}$
Équivalent binaire 1100
Poids : 2

Remarque : le poids maximum = nombre de variable de la fonction.

Une fois que le classement fait, on compare chaque terme de chaque groupe avec chaque terme de groupe de poids immédiatement supérieur.En utilisant l'identité suivante :$A\bar{x_i} + Ax_i = A$

A : représente l'ensemble des variables qui ne changent pas d'état.

xi : la variable qui change état. On obtiendra ainsi une 1ère forme réduite. Que l'on réduira de nouveau jusqu'à que ne soit plus possible de faire des réductions.

Exemple : Utiliser la méthode de Quine-Mc Cluskeypour simplifier la fonction suivante : $f(d,c,b,a) = \Sigma(0,1,3,5,7,8,10,12,13,14,15)$.

$f = \bar{a}\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d} + abc\bar{d} + a\bar{b}c\bar{d} + abc\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}b\bar{c}d + \bar{a}\bar{b}cd + a\bar{b}cd + \bar{a}bcd + abcd$

p=0	$\bar{a}\bar{b}\bar{c}\bar{d}$ (0)	$\bar{b}\bar{c}\bar{d}$ (0,1) $\bar{a}\bar{b}\bar{c}$	
p=1	$a\bar{b}\bar{c}\bar{d}$ (1) $\bar{a}\bar{b}\bar{c}d$ (8)	$a\bar{c}\bar{d}$ $a\bar{b}\bar{d}$ $\bar{a}\bar{c}d$ $\bar{a}\bar{b}d$	$a\bar{d}$ $\bar{a}d$

p=2	$ab\bar{c}\bar{d}$ $a\bar{b}c\bar{d}$ $\bar{a}b\bar{c}d$ $\bar{a}\bar{b}cd$	$ab\bar{d}$ $ac\bar{d}$ $a\bar{b}c$ $\bar{a}bd$ $\bar{b}c\bar{d}$ $\bar{a}cd$	ac cd
p=3	$abc\bar{d}$ $a\bar{b}cd$ $\bar{a}bcd$	abc acd bcd	
p=4	$abcd$		

Nous obtenons après avoir effectué toutes les comparaisons, la liste de l'intersection première. Nous allons maintenant former la grille des termes premiers.

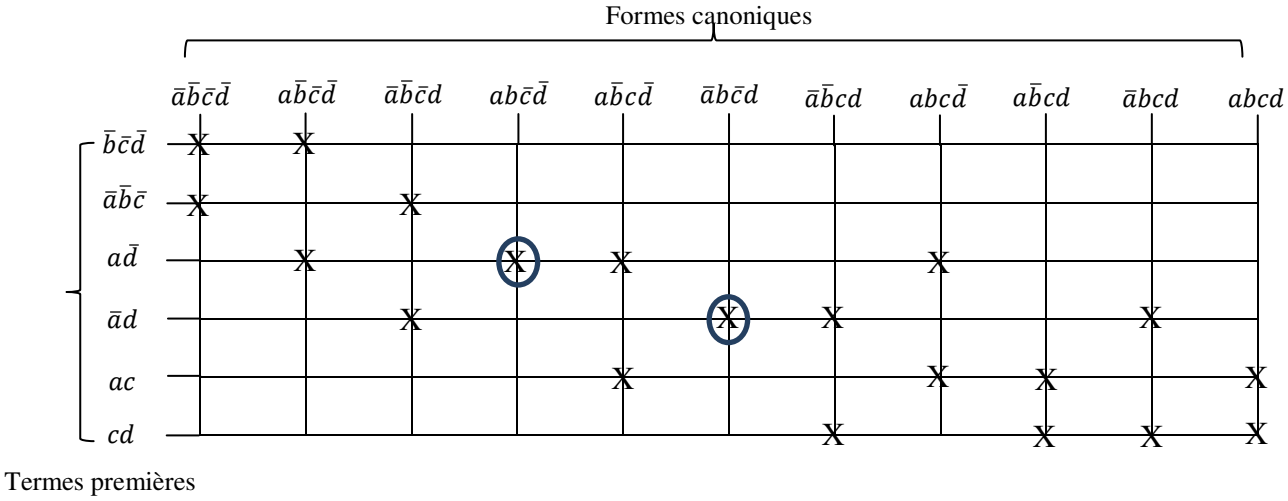


Figure III.4 : Grille des termes premiers.

$$f = a\bar{d} + \bar{a}\,d$$

L'expression de f n'est pas la forme minimale, ni l'expression algébrique de F. Elle ne contient que les termes essentiels.

Si le terme canonique figuré une seule fois dans une intersection première, cette dernière est un terme essentiel, et donc il doit apparaitre dans la forme simplifiée après avoir éliminé toutes les intersections canoniques qui sont inclues dans les termes essentiels. Trouver le reste constituera une nouvelle table s'appelle table de **choix**.

La table de choix est obtenue à partir de la grille des termes premiersoù le même terme canonique est figuré deux fois dans deux intersections premières successives.

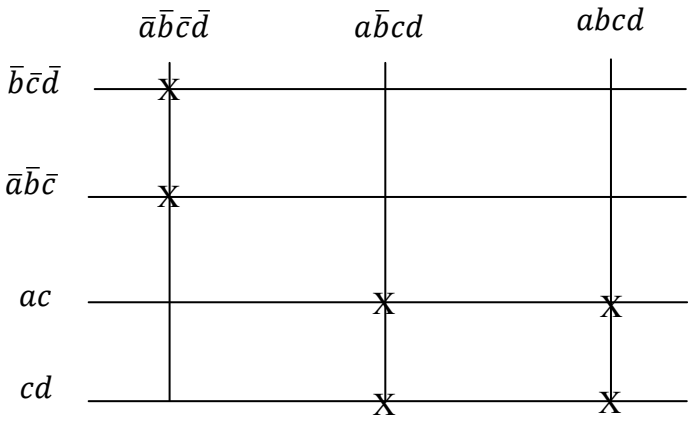


Figure III.5 : Table de choix.

La fonction simplifiée ou minimale de f comporte les termes essentiels et l'un des 4 termes donnés par la fonction de choix.Pour les couvrir, on a le choix entre $(\bar{b}\bar{c}\bar{d}, ac)\ ou\big(\bar{b}\bar{c}\bar{d},\ cd\big)\ ou\ \big(\bar{a}\bar{b}\bar{c},\ ac\big)\ ou\ \big(\bar{a}\bar{b}\bar{c},\ cd\big)$. Ce qui donne les quatre formes simplifiées minimales de f :

$$f = a\bar{d} + \bar{a}d + \bar{b}\bar{c}\bar{d} + ac$$

Oŭ $$= a\bar{d} + \bar{a}d + \bar{b}\bar{c}\bar{d} + cd$$

Oŭ $$= a\bar{d} + \bar{a}d + \bar{a}\bar{b}\bar{c} + ac$$

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

Avant-propos

Cet ouvrage traite la première partie qui consiste en l'étude des circuits logiques combinatoires. Il est destiné essentiellement aux étudiants ingénieurs, licenciés, masters et à tous les lecteurs qui veulent approfondir leurs connaissances dans le domaine de l'électronique numérique.

Dans le premier chapitre, on étudiera les différents systèmes de numération : systèmes Hexadécimal, décimal, octal et binaire ainsi que les processus de conversion d'un nombre écrit dans une base b quelconque àune autre base différente. Nous présenteronsensuite les codes numériques tels que les codes B.C.D, Code Excédent 3, Gray et ASCII. Nous terminons ce chapitre par l'étudedes codes détecteurs, correcteurs d'erreurs et la codification des nombres signés.

Le deuxième chapitre sera consacré à l'étude des opérateurs logiques et les lois fondamentales de l'algèbre de Boole ainsi que les lois de Morgan. Nous allons traiter aussi les fonctions logiques et la représentation géographique de ces dernières par le tableau de Karnaugh.

Le troisième chapitre présentera deux techniques de simplification des fonctions logiques, la simplification algébrique qui est basée sur les lois de l'algèbre booléenneet la simplification par la méthode de Karnaugh qui suive des règles précises pour minimiser la fonction logique.

Dans le dernier chapitre, nous allons étudier des systèmes combinatoires qui réalisent des fonctions particulières telles que : le décodeur, le codeur, le multiplexeur et le démultiplexeur.

Y.BELHADEF

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université ABOU BEKR BELKAID TLEMCEN
Faculté de Technologie

Département de Télécommunications
Laboratoire de Télécommunications TTL

Licence en Circuitset Systèmesde Télécommunications : CST

COURS : ELECTRONIQUE NUMERIQUE CONBINATOIRE

Présentée par

M^elle^ BELHADEF Yamina

Année universitaire 2016-2017

Sommaire

Chapitre 1 : Systèmes de Numération et Codage

Chapitre 2 : Algèbre de Boole et Fonction Logique

Chapitre 3 : Simplification des Fonctions Logiques

Chapitre 4 : Systèmes Logiques Combinatoires

Chapitre 1 : Systèmes de Numération et Codage

I.1. Introduction

Le système décimal est utilisé principalementpour représenter les chiffres, mais il existe d'autres systèmes de numération qui sont généralement rencontrés dans les circuits digitaux et les systèmes à base de micro processeurs.Dans ce chapitre, nous nous intéressons aux systèmes de numération suivants : binaire, octal, décimalet hexadécimal ainsi que les méthodes de conversion entre ces systèmes de numération. Nous allons traiter aussi plusieurs codes numériques tels que les codes BCD, Code Excédent 3, Gray et ASCII.

I.2. Base d'un système de numération

La base b du système de numération définie le nombre de symboles différents utilisés pour représenter des nombres quelconques dans un système de numération de base b.

Les bases de numérations les plus utilisées sont les suivantes :

- Base 2 ou système de numération binaire.
- Base 8 ou système de numération octal.
- Base 10 ou système de numération décimal.
- Base 16 ou système de numération hexadécimal.

L'écriture simplifiée d'un nombre N de base b peut être représentéde la façon suivante :

$$(N)_b = (a_n \dots \dots \dots a_0)_b \qquad (1)$$

Où $a_n \dots \dots \dots a_0$: représentent une séquence de symboles différents s'appelles chiffres ou digits.

Le nombre N de base b peut s'écrire sous la forme polynomiale suivante :

$$(N)_b = a_0 b^0 + a_1 b^1 + \cdots + a_n b^{\ n} \qquad avec \;\; 0 \leq a_i \leq b-1 \qquad (2)$$

Avec :

b :labase du système de numération.En décimal, on ne note pas l'indice$()_b$ pour le nombre N.
a_i: un chiffre (ou digit), et i : poids (Rang) du chiffre a_i**[1]**.

Par exemple, dans le système décimal, on écrit 4785 pour représenter le nombre :

$$4785 = 4.10^3 + 7.10^2 + 8.10^1 + 5.10^0$$

Rangs	3	2	1	0
Poids des rangs	10^3	10^2	10^1	10^0
Chiffre de poids le plus	fort			faible

I.3. Systèmes de numération et leurs bases

I.3.1. Système binaire(b=2)

C'est une des bases les plus utilisées en électronique. Ce système comporte deux symboles {0, 1} qui sont souvent appelés bits (binary digit)**[2]**.

Le nombre binaire N s'écrit sous la forme suivante :

$$N = (a_n \dots\dots\dots\dots a_0)_2 \qquad o\check{u} \quad a_i \in \{0,1\} \tag{3}$$

- a_n : le bit de poids le plus fort est appelé MSB (mostsignificant bit).
- a_0 : le bit de poids le plus faible est appelé LSB (lesssignificant bit).

Exemple : $N = (10111)_2$

Nous pouvons écrire le nombre N selon la forme polynomiale suivante :

$$N = \sum_{i=0}^{n} a_i 2^i, \quad o\check{u} \quad a_i \in \{0,1\} \tag{4}$$

* $(10111)_2 = 1 \times 2^0 + 1 \times 2^1 + 1 \times 2^2 + 0 \times 2^3 + 1 \times 2^4$
$= 1+2+4+0+16$
$= (23)_{10}$

I.3.2. Système décimal(b=10)

Dans unsystème décimal de base 10, il y a *dix digits*: 0,1,2,3,4,5,6,7,8 et 9 appelés *chiffres***[3]**.

Nous pouvons écrire le nombre N de la façon suivante :

$$N = \sum_{i=0}^{n} a_i 10^i, \quad o\check{u} \quad a_i \in \{0,1,2,3,4,5,6,7,8,9\} \tag{5}$$

Exemple :

* $(1234)_{10} = 4x10^0 + 3x10^1 + 2x10^2 + 1x10^3$
$=4 + 30 + 200 + 1000$
$= (1234)_{10}$

I.3.3. Système octal (b=8)

C’est un système à base 8, comprend 8 chiffres qui sont {0, 1, 2, 3, 4, 5, 6, 7}. Il n'y a pas de chiffres 8 et 9 **[4]**.

Selon la forme polynomiale, nous pouvons écrire :

$$N = \sum_{i=0}^{n} a_i 8^i, \quad o\check{u} \ a_i \in \{0,1,2,3,4,5,6,7\} \qquad (6)$$

Exemple :

* $(2145)_8 = 5 \times 8^0 + 4 \times 8^1 + 1 \times 8^2 + 2 \times 8^3$
  $= 5 + 32 + 64 + 1024$
  $= (1125)_{10}$

I.3.4. Système Hexadécimal (b=16)

C’est un système hexadécimal ou à base 16, contient 16 symboles qui sont $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$. Les lettres majuscules : A, B, C, D, E et F correspondent aux valeurs allant de 10 à 15.

Soit un nombre hexadécimal N s’écrit sous la forme suivante :

$$N = \sum_{i=0}^{n} a_i 16^i, \quad o\check{u} \ a_i \in \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\} \qquad (7)$$

Exemple :

$(B54)_{16} = 4.\ 16^0 + 5.16 + 11.16^2$
$= 4 + 80 + 2816$
$= (2900)_{10}$

I.4. Transcodage

I.4.1. Conversion d'un nombre écrit en base quelconque vers le décimal

La conversion d’un nombre N écrit dans une base b quelconque vers le décimal s'obtient par application directe de la forme polynomiale :

$$N = \sum_i a_i b^i \qquad (8)$$

Exemple: convertir les nombres suivants en leurs équivalents décimaux (la base est indiquée en indice) :

a. $(110101)_2$ **b.** $(2145)_8$ **c.** $(1FF)_{16}$

a. $(110101)_2 = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 0 \times 2^3 + 1 \times 2^4 + 1 \times 2^5$
$= 1 \times 1 + 0 \times 2 + 1 \times 4 + 0 \times 8 + 1 \times 16 + 1 \times 32$

= 1 + 4 +16 + 32
= $(53)_{10}$

b. $(512)_8 = 2 \times 8^0 + 1 \times 8^1 + 5 \times 8^2$
= 2 x 1 + 1 x 8 + 5 x 64
= 2 + 8 + 320
= $(330)_{10}$

c. $(1FF)_{16} = F \times 16^0 + F \times 16^1 + 1 \times 16^2$
= 15 x 1 + 15 x 16 + 1 x 256
= 15 + 240 + 256
= $(511)_{10}$

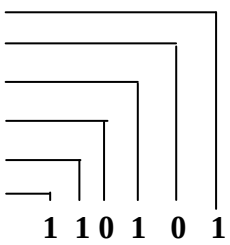
I.4.2. Conversions binaires

I.4.2.1. Décimal → binaire par division successive

On divise le nombre à convertir puis des quotients par la base 2 tout en conservant les restes de ces divisions jusqu'à ne plus pouvoir diviser par 2 (où jusqu'à l'obtention d'un quotient nul).Pour obtenir le nombre binaire, on écrit tous les restes de ces divisions en partant de la dernière division vers la première (sens de lecture vers le haut)**[5]**.

Exemple : $(53)_{10}$ à convertir en base 2

N= $(53)_{10}$ ÷ 2 = 26 reste 1
26 ÷ 2 =13 reste 0
13 ÷ 2 = 6 reste 1
6 ÷ 2 = 3 reste 0
3 ÷ 2 = 1 reste 1
1 ÷ 2 = 0 reste1



Le résultat est donc $(53)_{10} \rightarrow (110101)_2$

I.4.2.2. Décimal → binaire par soustraction

Cette méthode consiste à retrancher du nombre N_{10} la plus grande puissance entière de 2 possible, et ainsi de suite dans l'ordre décroissant des puissances. Si on peut retirer la puissance de 2 concernée, on note (1) sinon on note (0) et on continue de la même manière jusqu'à la plus petite puissance de 2 possible (2^0 pour les entiers).

Exemple : $(230)_{10}$ à convertir en base 2

De	230	on peut retirer	128 reste 102	(1)
De	102	on peut retirer	64 reste 38	(1)
De	38	on peut retirer	32 reste 6	(1)
De	6	on ne peut pas retirer	16 reste 6	(0)
De	6	on ne peut pas retirer	8 reste 6	(0)
De	6	on peut retirer	4 reste 2	(1)
De	2	on peut retirer	2 reste 0	(1)
De	0	on ne peut pas retirer	1 reste 0	(0)

↓ Sens de lecture

Le résultat est donc : $(230)_{10} \rightarrow (11100110)_2$

Cette technique implique de connaître les valeurs décimales associées aux puissances de 2 :

Puissance	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
résultat	1024	512	256	128	64	32	16	8	4	2	1

I.4.2.3. Conversion d'un nombre décimal fractionnaire → Binaire

On multiplie le nombre fractionnaire par 2, la partie entière ainsi obtenue représentant le poids binaire (1 ou 0). La partie fractionnaire restante est à nouveau multipliée par 2 et ainsi de suite jusqu'à ce que la partie fractionnaire soit nulle ou que la précision obtenue soit atteinte.

Exemple 1 : soit à convertir le nombre $(0.625)_{10}$

0.625x 2 = 1.250
0.250 x 2 = 0.500 ↓ Sens de lecture
0.500 x 2 = 1.000

Quand il ne reste plus de partie fractionnaire, on s'arrête. Ainsi $(0.625)_{10}$ devient $(0.101)_2$

Exemple 2 : soit à convertir le nombre $(0.583)_{10}$

0.583 x 2 = 1.166
0.166 x 2 = 0.332
0.332 x 2 = 0.664 Sens de lecture
0.664 x 2 = 1.328 $(0.583)_{10} = (0.1001)_2$

Remarque : Pour convertir un nombre décimal fractionnaire aux systèmes numériques à base b, on multiplie la partie fractionnaire par la base b jusqu'à l'obtention d'un nombre entier inférieur à la base ou la précision désirée.

I.4.2.4. Conversion d'un nombre binaire fractionnaire → décimal

Le principe est similaire :

- On utilise les puissances de 2 négatives en commençant à 2^{-1}
- On parcourt les bits de la gauche vers la droite.

Exemple : Convertir $(0.111)_2$ en base 10

$(0.111)_2 = (0 \times 2^0 , 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3})_{10}$

$= (0, 0.5 + 0.25 + 0.125)_{10}$

$= (0{,}875)_{10}$

I.4.3. Conversion binaire-octal et octal-binaire

I.4.3.1. Conversion binaire vers octal

A partir de la virgule, décomposer le nombre binaire par ensembles de 3 bits en allant vers la droite pour la partie fractionnaire et vers la gauche pour la partie entière. Convertir ensuite chaque groupe séparément en base 8 (octal) selon le code binaire naturel [6]. En effet, un mot binaire de 3 bits permet de coder les nombres entiers décimaux compris entre 0 et 7.

Le tableau ci-dessousexpose la correspondance entre les chiffres décimaux allant de 0 à 7 et leurs équivalents binaires sur 3 bits :

Décimal	Binaire
0	$(000)_2$
1	$(001)_2$
2	$(010)_2$
3	$(011)_2$
4	$(100)_2$
5	$(101)_2$
6	$(110)_2$
7	$(111)_2$

Tableau I.1 : Conversion des chiffres décimaux en binaire.

Exemple: Soit àConvertir en octal les nombres binaires suivants:

a. $(100\ 111\ 010\ 001)_2 = (4721)_8$

b. $(101\ 110\ 011\ 101)_2 = (5635)_8$

c. $(1110111)_2 = ?$

Il faut décomposer ce mot en 3 bits à partir de la droite : 1 110 111. On complète ce nombre binairepar des zéros à gauche pour avoir 3 bits : 001 110 111

Ce qui donne en tenant compte de la pondération 4-2-1 des bits respectifs : d'où le résultat : $(1110111)_2 = (167)_8$

I.4.3.2. Conversion octal vers binaire

Pour cela, il suffit d'écrire chaque digit d'un nombre exprimé en octal en son équivalent binaire sur 3 bits**[1]**.

Exemple : Convertir en binaire le nombre en octal $N=(7510.246)_8$

$N= (7510,246)_8$

$N= (111\ 101\ 001\ 000, 010\ 100\ 110)_2$

I.4.4. Conversion binaire-hexadécimal et hexadécimal-binaire

I.4.4.1. Conversion binaire vers hexadécimal

La conversion binaire hexadécimal, se fait en décomposantles bits du nombre binaire par ensembles de 4 bits en allant vers la droite pour la partie fractionnaire, et vers la gauche à partir de la virgule pour la partie entière, puis en remplace chaquegroupement par le chiffre hexadécimal correspondant **[4]**.

Le tableau suivant donne la correspondance entre les chiffres hexadécimaux allant de 0 à F et leurs équivalents binaires sur 4 bits :

Base 10	Base 16	Base 2
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
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Tableau I.2 : Conversion des nombres hexadécimaux en binaire.

Exemple : Convertir en hexadécimal le nombre binairesuivant : $(1110011101,01110001)_2$

Il faut regrouper ce nombre binaireen 4 bits à partir de la droite : $(11\ 1001\ 1101,0111\ 0001)_2$. On complète ce mot par des zéros à gauche pour avoir 4bits : $(\mathbf{00}11\ 1001\ 1101,0111\ 0001)_2$

D'où le résultat : $(0011\ 1001\ 1101,0111\ 0001)_2 = (3\ 9\ D,\ 71)_{16}$

I.4.4.2. Conversion hexadécimal vers binaire

La conversion hexadécimal –binaire s'obtient en remplaçant chaque chiffre du nombre hexadécimal par son équivalent binaire sur quatre bits.

Exemple : Soit à convertir en binaire le nombre en hexadécimal $N = (7A1F,B46)_{16}$

$N = (7A1F,B46)_{16}$
$N = (0111\ 1010\ 0001\ 1111,\ 1011\ 0100\ 0110)_2$

I.5. Codage

Le codage est une opération qui établit une correspondance entre les éléments de deux ensembles. A un nombre donné correspond une combinaison binaire appelée mot-code. L'ensemble des mots-code forme le code.

Dans tout système électronique, quel qu'il soit, le traitement des informations ne peut se faire sans avoir préalablement codé celles-ci. Suivant les types d'applications, plusieurs codes sont utilisés.

I.5.1. Codification des nombres naturels

Quand on fait correspondre à un nombre décimal son équivalent binaire on parle de codage binaire pur ou naturel.

I.5.1.1. Code DCB (BCD: Binary Coded Decimal)

Le code DCB (Décimal Codé Binaire) contient 10 mots-code, appelés tétrades ou quartets. Chaque chiffre du système décimal '0' à '9' est codé individuellement en binaire naturel sur 4 bits (quartet). Chaque élément binaire d'un mot code a un poids comme en binaire. Le bit le plus à gauche a le poids 8, le suivant 4, puis 2, puis 1. **[2]**.

Le code BCD est souvent utilisé pour le codage des nombres affichés sur une calculatrice.

Chiffres décimaux	Code BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Tableau I.3 : Conversion des chiffres décimaux en code BCD.

Exemple : Exprimons le nombre $(974)_{10}$ en code BCD

Résultat : $(974)_{10} = (1001\ 0111\ 0100)_{BCD}$

I.5.1.2.Code Excédent 3

Le code excédent ou bien le code majoré de 3 (excess 3) à une certaine parenté avec le code BCD. Le code excédent 3 d'un nombre décimal est déterminé de la même manière du code BCD sauf quand ajoute 3 à chaque chiffre décimal avant d'effectuer la conversion.

Exemple : $(1\ \ 3)_{10}$
+3 +3
4 6
(0100 0110)

Le tableau ci-dessous donne le code excédent 3 des 10 chiffres décimaux.

Chiffres décimaux	Code BCD	Excédent 3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100
10	1010	
11	1011	
12	1100	
13	1101	
14	1110	
15	1111	

Tableau I.4 : Conversion des chiffres décimaux en code Excédent 3.

Remarque : Les groupes non valides sont : 0000, 0001, 0010, 1101, 1110 et 1111.

I.5.1.3.Code Gray (code binaire réfléchi)

Dans ce code, il n'y a qu'une variable qui change d'état à la fois entre deux valeurs consécutives. Où bien deux représentations codées successives ne différent que d'un seul bit.II est nommé réfléchi car il faut recopier les valeurs comme si elles étaient réfléchies dans un miroir.

- On commence par tous les bits à 0 [0000].

- Pour obtenir la valeur suivante on change seulement le bit de poids le plus faible qui ne ramène pas à une valeur antérieure.

0000 →0
000**1** → 1
00**1**1 → 2
001**0** → 3
0**1**10 → 4

Le tableau suivant expose la correspondance entre le code décimal, le code binaire et le code binaire réfléchi sur 4 bits**[1]**,**[2]**:

Décimal	**Binaire**	**Binaire réfléchi**
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Tableau I.5 : Conversion des chiffres décimaux en binaire puis en code Gray.

Le code binaire réfléchi est un code non pondéré ; il n'obéit pas à la forme polynomiale. Ce code est d'une grande utilité dans les conversions d'une grandeur analogique en une grandeur numérique.

I.5.2.Code ASCII

Le code ASCII signifie (American Standard Code for Information Interchange), ilconstitue une norme universelle pour l'échange d'information entre le micro-ordinateur et ses périphériques (imprimante, clavier, écran, etc.). C'est un code qui permet la représentation des caractères alphanumériques d'un micro-ordinateur (les chiffres, les lettres majuscules et les

lettres minuscules ainsi que des caractères spéciaux (? ! + - : / # @ & …), Le code ASCII de base représentait les caractères sur 7 bits (c'est à- dire 128 caractères possibles, de 0 à 127)**[7]**.

I.5.3. Quelques codes particuliers

Le codage binaire est très courant pour une utilisation dans des circuits électroniques tels qu'un ordinateur, dans lesquels le signal électrique expédié sur la ligne estencodéau niveau de la réception (ordinateur) à l'autre extrémité du support. Lors du transport des données qui circulent souvent sur un long trajet (réseau par exemple), le signal électrique peut subir des perturbations et le récepteur ne reçoit pas l'information envoyée par l'émetteur. C'est pour cela, il existe des codes particuliers pour la détection d'erreurs et même pour la correction de ces erreurs.

I.5.3.1. Codes détecteurs d'erreurs

Les codes correcteurs d'erreurs sont employés pour éliminer les effets du bruitlors de la transmission des données. La méthode la plus répondue pour la détection des erreurs est la méthode debitde parité.

* ***Contrôle de parité***

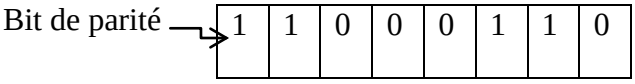
L'un des systèmes de contrôle les plus simples est lecode de parité (appelé parfois VRC : *Vertical Redundancy Check*).Lecode à contrôle de parité pair consiste à ajouter un « 1 » (appelé bit de parité)si le nombre total de bits à 1 du mot de code (généralement 7 bits, pour former un octet avec le bit de parité) soit un nombre impair, « 0 » dans le cas contraire**[8]**.

La méthode de contrôle de paritéimpair est une technique similaire à la précédente sauf que dans ce cas, le nombre total de bits à 1 du mot de code soit un nombre impair.

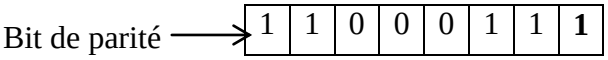
Selon l'exemple présenté ci-dessous, les bits de données étant en nombre pair, le bit de parité est donc positionné à 0 si on utilise le code de parité pair pour protéger l'information.

Bit de parité →								
0	1	1	0	0	1	1	0

Par contre, dans l'exemple suivant, le nombre de bits de données à 1 est impair, le bit de parité est placé à 1 :

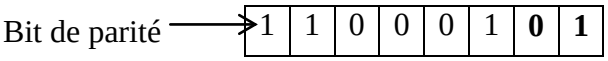


Imaginons dorénavant qu'après transmission de l'octet précédent le bit situé à droite soit erroné :



Suivant le bit de paritéutilisé pourprotéger l'octet transmis: une erreur est détectée.

Par contre, il impossible de détecter les erreurs de la séquence de bits transmissi deux bits (ou un nombre pair de bits) venaient à se modifier en même temps.



Le codeà contrôle de parité permet de détecter les erreurs en nombre impair, il nedétecte donc que 50 % des erreurs.

L'inconvénient majeur observépour ce système de détection d'erreursest qu'il nepermet pasde corriger les erreurs détectées (la seule solution est d'exiger la retransmission de l'octet inexact…).C'est pour cette raison, ils existent d'autres systèmes de détection d'erreurs plus améliorésqui ont été mis au point, les codes les plus utilisablessont nommés :

- Codes autocorrecteurs
- Codes autovérificateurs

I.5.3.2. Codes correcteurs d'erreurs

La technique de codage en appliquant le code correcteur d'erreurs est basée sur la redondance. Elle est suffisante pour corriger les erreurs de transmission d'une information et de retrouver le message initial sur un support de communication moins fiable.

Parmi ces codes nous pouvons mentionneruncode correcteur linéaire ou précisément le code de Hamming. Ce code permet la détection et la correction automatique d'une erreur en précisant sa positionsur une seule lettre du message.

Suivant la longueur du mot code, il n'existe pas d'autre code plus denseayant la même efficacité de correction que le code de Hamming. Donc c'est un code parfaitou son rendement est plus important.

I.5.4. Codification des nombres signés

I.5.4.1. Représentation d'un entier relatif

Le codage d'un nombre entier relatif consiste à savoir s'il s'agit d'un nombre positif ou d'un nombre négatif. De plus, il faut conserver les règles d'addition pour calculer ces nombres signés. Dans ce cas, on utilise un type de codage que l'on appelle complément à deux et qui permet d'exécuter les opérations arithmétiques employées naturellement**[9]**.

- **Un entier relatif positif** ou nul sera représenté en binaire (base 2) comme un entier naturel, à la seule différence que le bit le plus significatif (le bit de poids fort)représente le signe. Les autres bits codent la valeur absolue du nombre. On aura toujours un bit utilisé pour préciser le signe du nombre (0 correspond à un signe positif, 1 à un signe négatif). Par exemple, si on code lenombre entier naturel 7 (en base décimale) sur 4 bits, le nombre le plus grand sera 0111 **[9]**.
  - Sur 1 octet (8 bits), l'intervalle de codage est [-128, 127].
  - Sur 2 octets (16 bits), l'intervalle de codage est [-32768, 32767].
  - Sur 4 octets (32 bits), l'intervalle de codage est [-2147483648, 2147483647].

  D'une façon générale un codage sur n bits de la plus grand entier relatif positif sera 2^{n-1}-1.

- **Un entier relatif négatif** sera représenté grâce au codage en complément à deux.

I.5.4.2. Principe du complément à deux

1. La méthode la plus simple consiste à écrire la valeur absolue du nombre en binaire pure.Le bit de poids fort doitprendre la valeur 0.
2. On inverse les bits de l'écriture binaire de la valeur absolue (les 1 deviennent des 0 et vice versa). On fait ce qu'on appelle le complément à un.
3. On ajoute 1 au résultat (les dépassements sont ignorés).

Cette opération correspond au calcul de 2^n-$|x|$, où n est la longueur de la représentation et $|x|$ la valeur absolue du nombre à coder.

Ainsi -1 s'écrit comme 256-1=255=$(11111111)_2$, pour les nombres sur 8 bits.

Exemple :

On désire coder la valeur –35 sur 8 bits. Il suffit :

1. d'écrire 35 en binaire : 00100011
2. d'écrire son complément à 1 : 11011100
3. et d'ajouter 1 : 11011101

La représentation binaire de -35 sur 8 bits est donc 11011101.

On remarquera qu'en additionnant un nombre et son complément à deux on obtient 0. En effet,

00100011 + 11011101 = 00000000 (avec une retenue de 1 qui est éliminée).

Chapitre 2 : Algèbre de Boole et Fonction Logique

II.1. Introduction

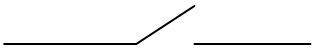
L'algèbre de BOOLE ne traite que de la logique combinatoire, c'est à dire des circuits numériques dont la sortie ne dépend que de l'état présent des entrées (sans mémoire des états passés).Dans ce chapitre, on étudiera les opérateurs logiques qui constituent les blocs élémentaires des circuits logiques et nous verrons comment il est possible de décrire leur fonctionnement grâce à l'algèbre de Boole. Nous verrons également comment on construit des logigrammes en associant des opérateurs logiquesbooléen (NON, ET, OU, NON ET, NON OU, OU exclusif, NON OU exclusif).

II.2. Variable logique

Dans un système binaire lavariable ne peut prendre que deux valeurs distinctes un 0 et 1, elle est appelée**Variable Logique.** En électronique ces deux valeurscorrespondes à deux états passant-bloqué ou à deux niveaux haut (H) et bas (L).

Exemple :

1. Soit un interrupteur à deux positions et soit « a »la variable logique qui définit son état : on associe à chacune des positions un état de la variable.



a = 0 interrupteur (ouvert)

a = 1 interrupteur (fermé)

2. Soit une ampoule électrique et soit X la variable logique qui définit son état :
   - Lampe éteinte→ Etat de la variable X = 0
   - Lampe allumée→ Etat de la variable X = 1

II.3. Algèbre de Boole et Opérateur logique

L'algèbre de Boole est l'outil mathématique qui permet d'établir la relation entre les sorties (fonctions logiques) et les entrées (les variables logiques) d'un système logique (synthèse du système). Réciproquement, cet outil nous permet de déterminer les règles de fonctionnement d'un système logique existant (analyse de système).

L'algèbre de Boole est définie sur l'ensemble E constitué des éléments {0,1} et muni de trois opérateurs de base.

- 2 lois de composition :
  - OU / OR notée (a + b) : Loi d'addition logique, Retourne 1 si a ou b est 1, sinon retourne 0.
  - ET / AND notée (a . b ou ab) : Loi de multiplication logique, Retourne 1 si a et b sont à 1, sinon retourne 0.
- Une loi de complément négation
  - NON / NOT (a) : inverse / complémente la valeur de la variable a.

II.4. Opérateurs Logiques

Les fonctions logiques sont conçues à partir d'un groupe d'opérateurs élémentaires appelés « portes ». Chaque opérateur est représenté par un symbole et sa fonction est définie par une table de vérité**[10]**.

II.4.1. Porte NON (NOT), porte inverseur

Si A est la variable d'entrée, S la variable de sortie vaut : $S = \bar{A}$ (on prononce A barre).La fonction NON est un opérateur qui affecte à la variable de sortie l'état complémentaire de la variable d'entrée.L'action de cet opérateur d'inversion est résumée dans le tableau suivant**[11]**.

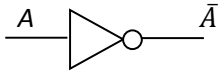
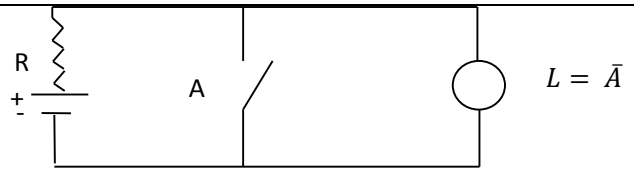
Symbole	Table de vérité	Montage
	A | $S = \bar{A}$ 0 | 1 1 | 0	 A = 0 → L =1 (la lampe s'allume) A = 1 → L = 0 donc $L = \bar{A}$

Tableau II.1 : Propriétés de la porte inverseur NOT.

Exemple :

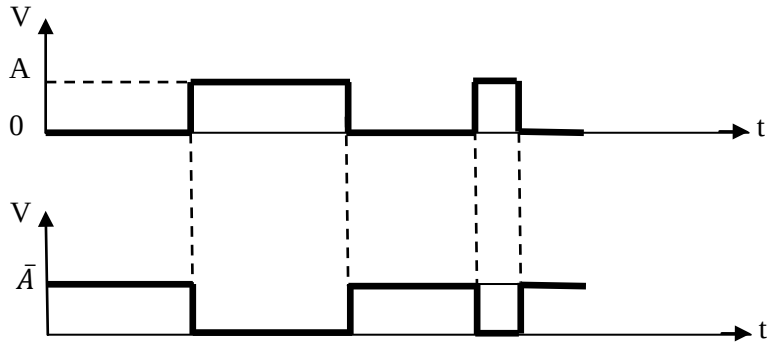
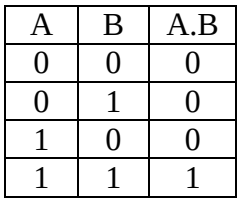
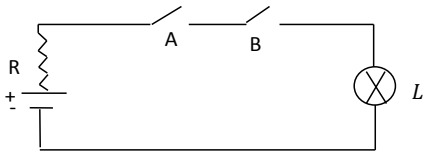


Figure II.1 : Chronogrammes des différentes entrées/sortie d'une porte NOT.

II.4.2.Porte ET (AND)

Si A et B sont les variables d'entrées de l'opérateur AND (ET), alors la sortie S = A.B. La sortie est à l'état 1 si les deux entrées sont simultanément à l'état1. La fonction AND est symbolisée par l'opérateur mathématique « **.** ». L'action de cet opérateur est résumée dans le tableau suivant :

Symbole	Table de vérité	Montage
		 L s'allume si A =1 et B = 1 donc L= AB

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Tableau II.2 : Propriétés de la porte ET.

Exemple :

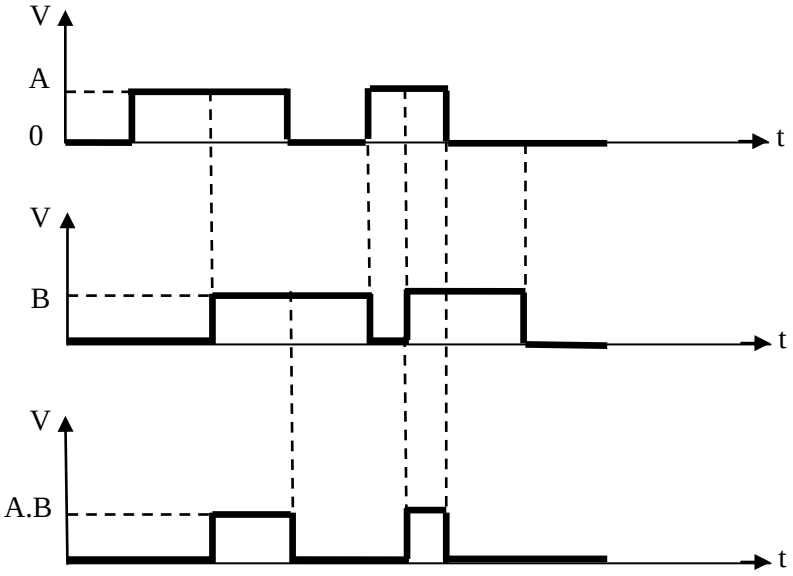
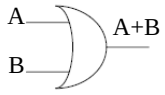
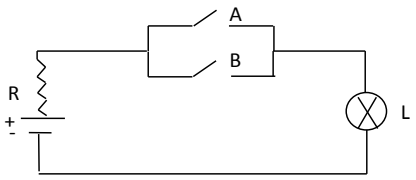


Figure II.2 : Chronogrammes des différentes entrées/sortie d'une porte ET.

II.4.3. Porte OU (OR)

Si A et B sont les variables d'entrées de l'opérateur OR (OU), alors la sortie S = A+B.La sortie est à l'état 1 si au moins une des entrées est à l'état 1. La fonction OR est symbolisée par l'opérateur mathématique« + ». L'action de cet opérateur est résuméedans le tableau suivant :

<table>
<tr><th>Symbole</th><th>Table de vérité</th><th>Montage</th></tr>
<tr>
<td> 
</td>
<td>
<table>
<tr><th>A</th><th>B</th><th>A+B</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
<td> 

L s'allume si A =1 ou B = 1 donc L= A+B</td>
</tr>
</table>

Tableau II.3 : Propriétés de la porte OU.

Exemple :

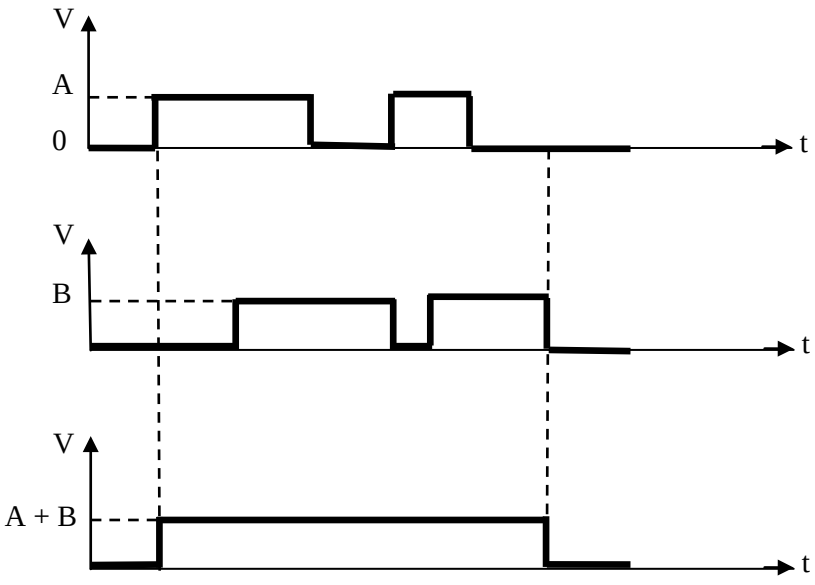


Figure II. 3 : Chronogrammes des différentes entrées/sortie d'une porte OU.

II.5. Lois fondamentales de l'algèbre de Boole[12]

* **Loi de commutativité**
  - A + B = B + A
  - A . B = B . A

* **Loi d'associativité**
  - A + (B + C) = (A + B) + C

- A . (B . C) = (A .B) . C

* **Éléments neutres**
  - A + 0 = A
  - A . 1 = A

* **Éléments absorbants**
  - A +1 = 1
  - A . 0 =0

* **Loi de complémentarité (Eléments symétriques)**
  - $A + \bar{A} = 1$
  - $A \,.\, \bar{A} = 0$

* **Loi de l'idempotence**
  - A + A = A
  - A . A = A

* **Loi de l'involution** $\bar{\bar{A}} = A$
* **Loi de l'Absorption**
  - A + (B.A) = A
  - A.(B + A) = A

* **Loi de l'inclusion** $(A.B) + (A.\bar{B}) = A$

* **Loi de distributivité**
  - AND est distributive sur OR : A . (B + C) = (A . B) + (A . C)
  - OR est distributive sur AND : A + (B . C) = (A + B) . (A + C)

(A + B) . (A + C) = A . A + A . C + A . B + B . C
=A + A . C + A . B + B . C
= A (1 + C + B) + $\underbrace{B \,.\, C}_{=1}$
= A . 1 + B . C
= A + B . C

II.5.1. Théorèmes de Morgan

Premier théorème : Le complément d'une somme logique est égalau produit logique des termes de cette somme, eux-mêmes complémentés. Le théorème s'applique quelque soit le nombre de termes de la somme**[13]**.

$\overline{a + b} = \bar{a} \cdot \bar{b}$ Expression généralisée : $\overline{\sum_{i=1}^{n} a_i} = \prod_{i=1}^{n} \bar{a}_i$

Deuxième théorème : Le complément d'un produit logique est égal à la somme logique des termes de ce produit, eux-mêmes complémentés. Le théorème s'applique quelque soit le nombre de termes du produit.

$\overline{a \cdot b} = \bar{a} + \bar{b}$ Expression généralisée : $\overline{\prod_{i=1}^{n} a_i} = \sum_{i=1}^{n} \bar{a_i}$

II.5.2. D'autres théorèmes

Premier théorème :Loi d'absorption

$$A + A.B = A$$

$$A \underbrace{(1 + B)}_{=1} = A$$

Deuxième théorème :$A + \bar{A}.B = A + B$?

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

Troisième théorème :$AB + \bar{A}C + BC = AB + \bar{A}C$?

$= AB + \bar{A}C + BC\ (A + \bar{A})$

$= AB + \bar{A}C + ABC + \bar{A}BC$

$= AB + \bar{A}C$

II.6. Opérateurs logiques universels

En technique numérique on retrouve souvent deux autres types de portes logiques qui sont : les portes NON OU (**NOR**) et NON ET (**NAND**).

II.6.1.NAND (NON ET)

Si A et B sont les variables d'entrées de l'opérateur NAND (NON ET), alors la sortie$S = \overline{A.B}$.La sortie donneunrésultatvraidèsquel'unedesentréesestfausse. La fonction NAND est l'inverse de la fonction AND. Le symbole de la porte NAND est le symbole duAND suivi d'une bulle qui matérialise l'inversion. L'action de cet opérateur est résumée dans le tableau suivant**[11]**.

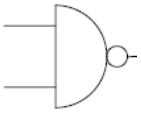
Symbole	Table de vérité
	A | B | $\overline{A.B}$ 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0

Tableau II.4 : Propriétés de la porte NON ET.

Exemple : déterminer la forme d'onde d'une sortie de porte NAND pour les entrées suivantes.

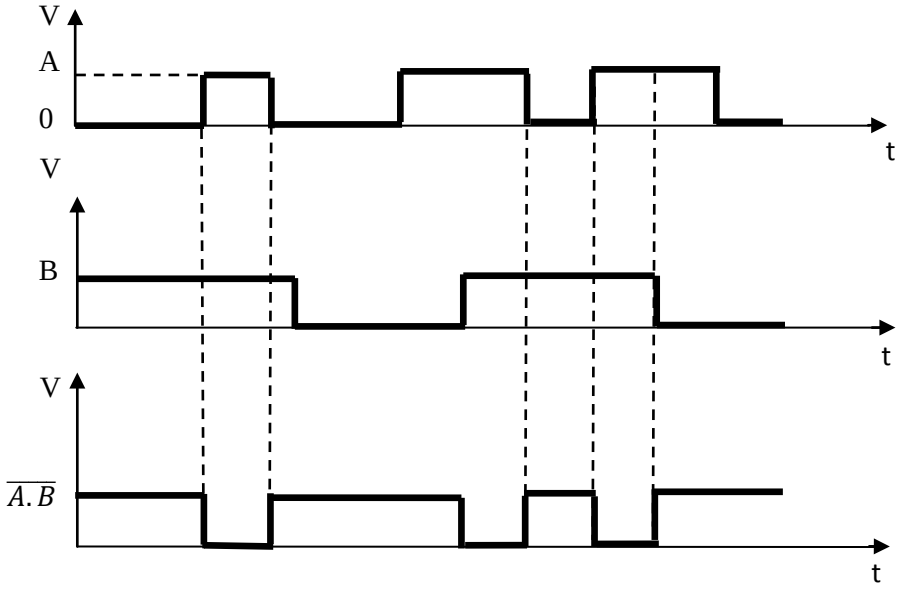


Figure II.4 : Chronogrammes des différentes entrées/sortie d'une porte NAND.

II.6.2. Porte NOR (NOT OR)

Si A et B sont les variables d'entrées de l'opérateur NOR (NON OU), alors la sortie $S = \overline{A + B}$. La sortie est à un niveau haut seulement si les deux entrées ont un niveau bas. La fonction NOR est l'inverse de la fonction OR. Le symbole de la porte NOR est le symbole du OR suivi d'une bulle qui matérialise l'inversion. L'action de cet opérateur est résumée dans le tableau suivant :

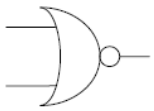
Symbole	Table de vérité
	A B $\overline{A+B}$ 0 0 1 0 1 0 1 0 0 1 1 0

Tableau II.5 : Propriétés de la porte NON OU.

Exemple : déterminer la forme d'onde de sortie d'une porte NOR pour les entrées suivantes :

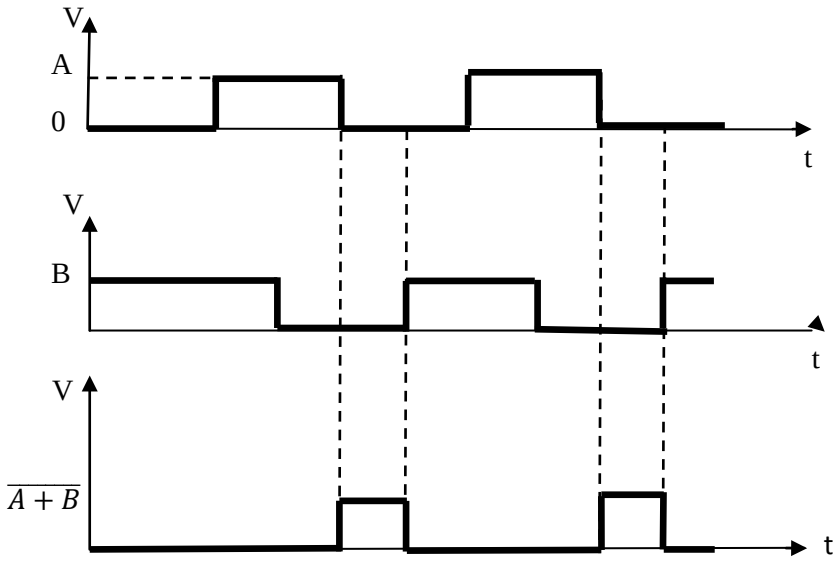


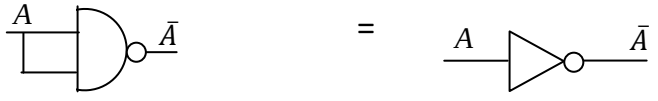
Figure II.5 : Chronogrammes des différentes entrées/sortie d'une porte NOR.

II.7. Réalisation des opérateurs logiques ET, OU, NON en fonction de NAND et NOR

Il est possible d'écrire les trois opérateurs de base ET, OU, NON à partir de l'opérateur NOR, ou de l'opérateur NAND.

II.7.1. OpérateursET, OU, NON en logiqueNAND

* **La porte inverseur NON:** $\bar{A} = \overline{A \cdot A}$



* **La porte AND :** $A \cdot B = \overline{\overline{A \cdot B}} = \overline{\overline{A \cdot B}\overline{A \cdot B}}$

* **La porte OU :** $A + B = \overline{\overline{A + B}} = \overline{\bar{A} . \bar{B}}$

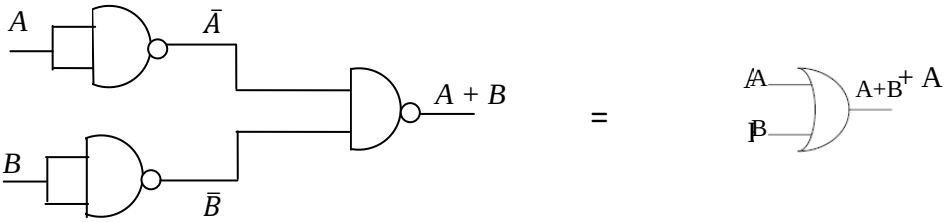
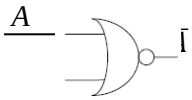


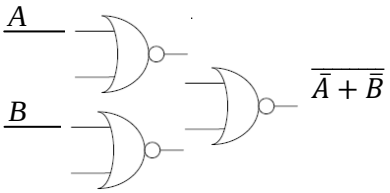
Figure II.6 : Réalisation des portes ET, OU, NON à l'aide des portes NAND.

II.7.2. OpérateursET, OU, NON en logiqueNOR

* **La porte inverseur NON:** $\bar{A} = \overline{A + A}$



* **La porte AND :** $A \cdot B = \overline{\overline{A \cdot B}} = \overline{\bar{A} + \bar{B}}$



* **La porte OU :** $A + B = \overline{\overline{A + B}} = \overline{\overline{A + B} + \overline{A + B}}$

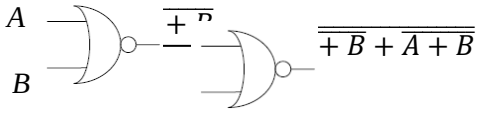


Figure II.7 : Réalisation des portes ET, OU, NON à l'aide des portes NOR.

<u>Exemple</u> : Réaliser le circuit logique de la fonction F en utilisant des portes NAND puis des portes NOR

$$F = \bar{a}bc + ab\bar{c} + a\bar{b}c$$

* **Réalisation de F à l'aide de portes NAND**

$$F = \bar{\bar{F}} = \overline{\overline{\bar{a}bc + ab\bar{c} + a\bar{b}c}} = \overline{\overline{\bar{a}bc} \cdot \overline{ab\bar{c}} \cdot \overline{a\bar{b}c}}$$

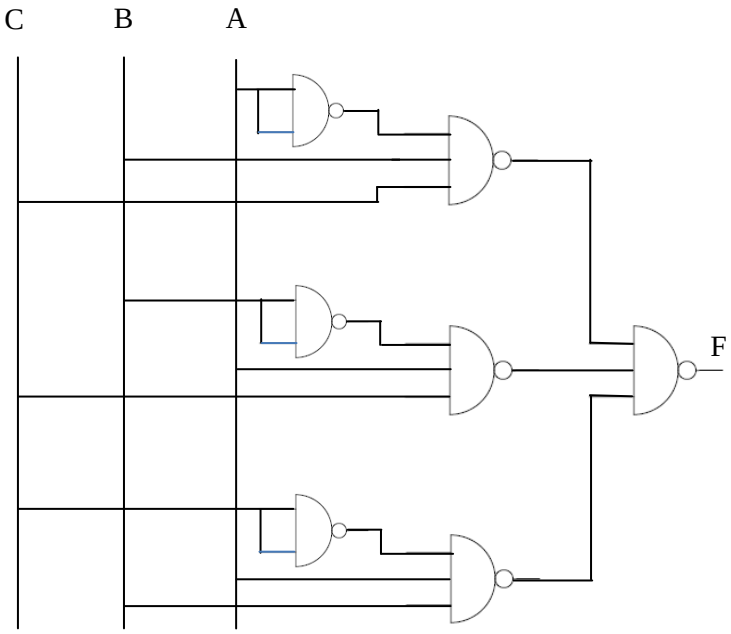


Figure II.8 : Réalisation de la fonction F à l'aide de portes NAND.

* **Réalisation de F à l'aide de portes NOR**

$$F = \bar{a}bc + ab\bar{c} + a\bar{b}c = \overline{\overline{\overline{\bar{a}bc} + \overline{ab\bar{c}} + \overline{a\bar{b}c}}}$$

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

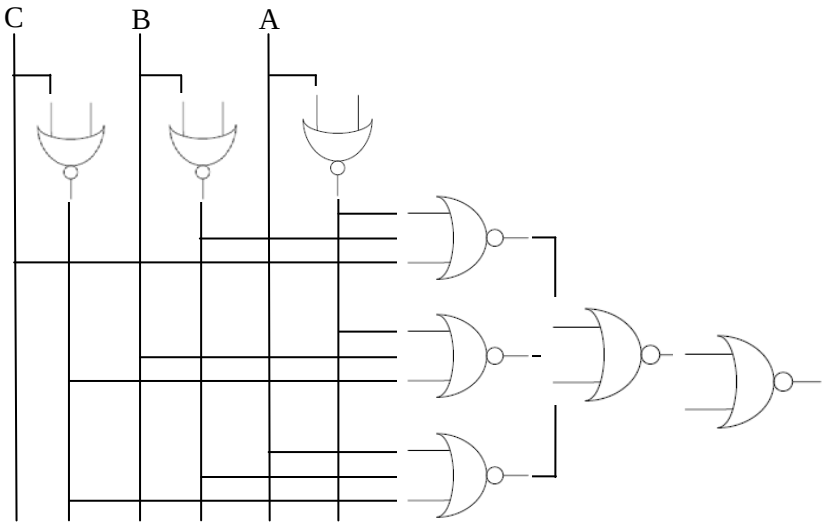


Figure II.9 : Réalisation de la fonction F à l'aide de portes NOR.

II.8. Opération OU exclusif (XOR), et identité logique $(\overline{XOR})$

II.8.1. OU exclusif

La porte XOR (OU exclusif) ne porte que deux entrées. Si A et B sont les variables d'entrées de l'opérateur XOR, alors la sortie$S = A \oplus B = A \cdot \bar{B} + \bar{A} \cdot B$. S vaut 1 si une seule (exclusivement une) de ses entrées vaut 1. La fonction XOR est symbolisée par $\oplus$ (un + entouré d'un cercle) car il réalise l'addition en binaire, mais modulo 2. Le OU normal (inclusif) prend l'état 1 si les deux entrées sont à l'état 1 (1+1=1). Le OU exclusif exclut ce cas (d'où son nom). L'action de cet opérateur est résumée dans le tableau suivant**[11]**.

Symbole	Table de vérité

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Tableau II.6 : Propriétés de la porte XOR.

* **Réalisation de OU exclusif à l'aide de portes NAND**

1ère méthode : $S = A \oplus B = \overline{\overline{A\bar{B} + \bar{A}B}} = \overline{\overline{A\bar{B}}.\overline{\bar{A}B}}$

2ème méthode :

$S = A\bar{B} + \bar{A}B + \bar{A}A + \bar{B}B$ avec

$\bar{A}A = 0$

$\bar{B}B = 0$

$S = \bar{A}(A + B) + \bar{B}(A + B) = (A + B)(\bar{A} + \bar{B})$

$= A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B})$

$= A.\overline{AB} + B.\overline{AB} = \overline{\overline{A.\overline{AB} + B.\overline{AB}}}$

$S = \overline{\overline{A.\overline{AB}}.\overline{B.\overline{AB}}}$ → **NAND**

II.8.2. Opération identité logique $(\overline{XOR})$

L'opérateur XNOR (NON OU exclusif) porte sur deux variables d'entrées. Si A et B sont les variables d'entrées, alors la sortie$S = \overline{A \oplus B} = \bar{A} \cdot \bar{B} + A \cdot B$. S vaut 1 si les deux entrées sont identiques. La fonction XNOR est l'inverse de la fonction XOR. Le symbole de

la porte XNOR est le symbole du XOR suivi d'une inversion. L'action de cet opérateur est résumée dans le tableau suivant :

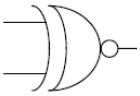
Symbole	Table de vérité
	A | B | $A \odot B$ 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1

Tableau II.7 : Propriétés de la porte$\overline{XOR}$.

II.9. Fonctions logiques

La logique combinatoire est la fonction logique dont la valeur de sortie ne dépend que de l'état des entrées (pas de mémorisation). La fonction logique est une fonction qui relie N variables logiques avec un ensemble d'opérateurs logiques de base **[14]**.

- Une fonction logique possède une ou plusieurs variables logiques d'entrées et une variable logique de sortie.
- La valeur de sortie (0 ou 1) de cette fonction dépend de la valeur des variables d'entrées.

Exemple :$f(a,b,c,d) = a + \overline{(b+\bar{c}).a} + \bar{d}.(\bar{b}+c)$

Les fonctions logiques peuvent être représentées par des Tables de vérités.

II.9.1.Table de Vérité

Si une fonction logique possède n variables logiques on aura donc 2^n combinaisons. Les 2^n combinaisons sont représentées dans une table qui s'appelle table de vérité**[10]**.

En peut représenter la fonction logique de n variables$f = (x_1, \ldots\ldots, x_n)$ par un tableau (table de vérité de 2^n lignes et (n+1) colonnes).

- n pour les variables.
- 1 pour les fonctions.

L'ordre des combinaisonssera croissant de haut vers le bas et elles seront écrites dans le code binaire naturel.

x_n	x_{n-1}		x_1	F
0	0	…….. ……	0	0
0	0	……..	1	0
.	.		.	1
.	.		.	.
1	1			.
1	.		1	

2^n ligne

n+1 Colonne

Tableau II.8 : Table de vérité de 2^n lignes et (n+1) colonnes.

Exemple :

$$F = \bar{A}B + B\bar{C} + ABC = f(A, B, C)$$

Chiffres décimaux	C	B	A	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

F=1 si ABC =1

ou$B\bar{C} = 1$

ou$\bar{A}B = 1$

II.9.2. Formes canoniques

On appelleforme canonique (ou normale) d'une fonction la forme ou chaque terme contienttoutes les variables. L'écriture sous forme canonique est unique.

Exemple : $F(A, B, C) = \bar{A}BC + AC\bar{B} + AB\bar{C} + ACB$

Il existe plusieurs formes canoniques : les plus utilisées sont la première et la deuxième forme.

II.9.2.1.1ère forme canonique (somme canonique)

On appelle « minterme » de N variables, un produit de ces N variables ou de leurs complémentaires.

La1ère forme canonique (forme canonique disjonctive : FCD) est la somme demintermesou la somme de produits algébriques pour lesquels la fonction vaut 1.**[15]**.

Exemple: Soit A, B, C et Dquatre variables booléennes ;

Somme canonique de F :$F = \bar{A}B\bar{C} + AB\bar{C} + \bar{A}BC + ABC$

$\bar{A}B\bar{C}, AB\bar{C}, \bar{A}BC, ABC$: sont des mintermes.

II.9.2.2.2ème forme canonique : (produit canonique)

On appelle « maxterme » de N variables, une somme logique de ces N variables ou de leurs complémentaires.

La 2ème forme canonique (forme canonique conjonctive : FCC) est le produit des maxtermesou le produit de sommes algébriques pour lesquels la fonction vaut 0**[15]**.

Exemple: On cherche la 2ère forme canonique de F

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

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

$$= \left(\overline{\bar{A}\bar{B}\bar{C}}\right).\left(\overline{A\bar{B}\bar{C}}\right).\left(\overline{\bar{A}\bar{B}C}\right).\left(\overline{A\bar{B}C}\right)$$

$F = (A + B + C).(\bar{A} + B + C).(A + B + \bar{C}).(\bar{A} + B + \bar{C})$→ 2ème forme canonique de F

II.9.3. Image d'une fonction

L'image d'une fonction logique de F notée ÷ F est la valeur binaire {0, 1} de F pour tous les combinaisons des variables de la fonction F.

÷ F = 0 0 1 1 0 0 1 1 (Image de F)

0 1 2 3 4 5 6 7

L'image d'une somme ou d'un produit des fonctions logiques est la somme ou le produit des images de ses fonctions respectivement.

Exemple :÷ F_1= 0 0 1 1 0 0 0 1

÷ F_2= 0 1 0 1 0 0 1 1

÷ (F_1 + F_2) = 0 1 1 1 0 0 1 1

÷ (F_1 . F_2) = 0 0 0 1 0 0 0 1

II.9.4. Expression numérique

En peut représenter une fonction logique par la valeur décimal de chaque combinaison des variables, elle peut être définie comme **[16]**:

- Une somme des états (mintermes) pour lesquels la fonction logique vaut 1. On note $F = \sum(N_1, \dots N_p)$.
- Un produit desétats (maxtermes)pour lesquelsla fonction logique vaut 0. On note$F = \prod(N_1, \dots N_q)$.

Exemple :

Soit une fonction logique F(A, B, C) de 3 variables A, B et C.

On pose la variable « A » poids fort : $N = A \,.\, 2^2 + B \,.\, 2^1 + C \,.\, 2^0$

N	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

- La première forme canonique de F (somme de produits) :

$$F = \sum(1, 2, 4, 7) = \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot \bar{C} + A \cdot B \cdot C$$

- La deuxième forme canonique de F (produit de sommes) :

$$F = \prod(0, 3, 5, \quad 6)$$

$$= (A + B + C) \cdot (A + \bar{B} + \bar{C}) \cdot (\bar{A} + B + \bar{C}) \cdot (\bar{A} + \bar{B} + C)$$

II.10. Tableau de Karnaugh

Le tableau de Karnaugh est un outil graphique qui permet de simplifier facilement et méthodiquement des expressions booléennes de 3 à 5 variables**[17]**.

Pour une fonction logique F de N variables, le tableau est constitué de la façon suivante :

- Chaque case du tableau de karnaugh correspond à une rangée de la table de vérité.
- C'est un tableau de 2^N cases, N étant le nombre de variables de la fonction logique.
- Chaque case représente une combinaison des entrées.
- Sur les lignes et les colonnes du tableau, on représente l'état des variables d'entrées codées en binaire réfléchi (code Gray). Chaque fois que l'on passe d'une case à l'autre, une seule variable change d'état (cases adjacentes).
- On recherche les cases adjacentes qui ont pour valeur 1 et on les regroupe en 8, 4 et 2.

Lestableaux de Karnaugh en fonction du nombre d'entrées sont présentés ci-dessous.

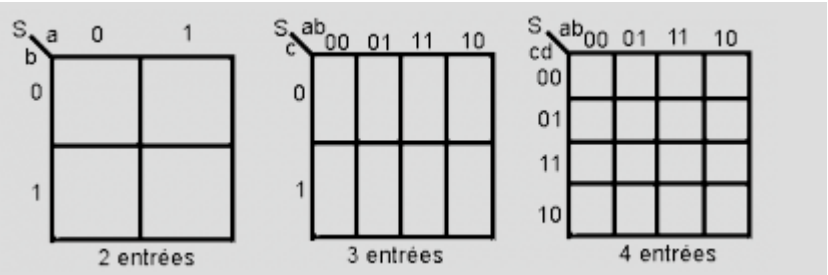


Figure II.10 : Tableaux de Karnaugh en fonction du nombre d'entrées.

Valeurs décimal	**C**	**B**	**A**	**F**
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0

Karnaughà fonction F à 3 variables.

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 1:Présenter le tableau de partir de la table de vérité de la

Le tableau de karnaugh de la fonction F :

C \ BA	**00**	**01**	**11**	**10**
0	0	0	1	0
1	0	1	1	1

Exemple 2 : Soit la fonction : $F = ab\bar{c}\bar{d} + \bar{a}b\bar{c}\bar{d} + \bar{a}\,\bar{b}\bar{c}d + \bar{a}b\bar{c}d$. Donner le tableau de karnaugh de la fonction F.

Le tableau de Karnaugh d'une matrice de 4 variables : F (a,b,c,d)

dc \ ba	**00**	**01**	**11**	**10**
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

d c b a

$2^3\ 2^2\ 2^1\ 2^0$

Le tableau de Karnaugh de la fonction F :

dc \ ba	**00**	**01**	**11**	**10**
00	0	0	**1**	**1**
01	0	0	0	0
11	0	0	0	0
10	**1**	0	0	1

Remarque : les tableaux de Karnaugh sont plus d'une grande utilité quand le nombre de variables dépasse à 6.

Chapitre 4 : Systèmes Logiques Combinatoires

IV.1. Système logique combinatoire

Un système logique combinatoire est un système comprenant plusieurs fonctions combinatoires. Ces fonctions de sorties s'expriment selon des expressions logiques des seules variables d'entrées. Elles peuvent se présenter sous forme de **[16]**:

- Table de vérité,
- Représentation décimale (somme des états ou produit des états)
- Expression canonique,
- Tableau de KARNAUGH,
- Schéma logique (logigramme)….

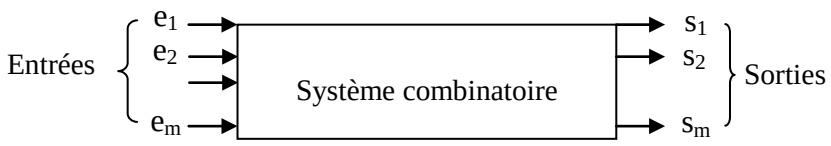


Figure IV.1 : Symbole d'un système logique combinatoire.

$S_1 = f_1(e_1 , e_2 , \ldots, e_m)$

$S_2 = f_2(e_1 , e_2 , \ldots, e_m)$

….

$S_m = f_m(e_1 , e_2 , \ldots, e_m)$

Les sorties S_i ne subissent pas de changement tant que les entrées ne changent pas.

IV.2. Synthèses des systèmes combinatoires

Exemple 1 : un moteur électrique à 2 vitesses peut être miseen marche par deux boutons poussoirs **a** et **b.**

1. au repos **a** et **b** ne sont pas actionnés
2. l'action en **a,** le moteur tourne lentement.
3. Action simultanée sur **a** et **b**, le moteur tourne rapidement.
4. **a** relâché, **b** actionné, le moteur tourne rapidement.
5. **b** relâché, le moteur s'arrête. Etablir le schéma de l'installation.

1/ Déterminer les entrées et les sorties.

2/ La T.V.

3/ Simplification des équations logiques et réalisation.

Solution :

1/ Les entrées du système: a, b.

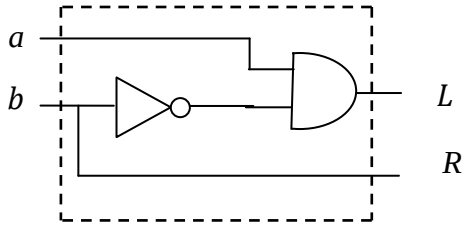
Les sorties: L et R.

Les fonctions logiques : $L = a\bar{b}$, $R = \bar{a}b + ab = b$

2/T.V :

a	b	L	R
0	**0**	0	0
0	**1**	0	1
1	**0**	1	0
1	**1**	0	1

3/ Réalisation



Exemple 2 : Deux pompes alimentent un même réservoir, 2 contacts x et y liés à des flotteurs sont actionnés, lorsque le niveau d'eau est inférieur à celui des contacts, le contact x est au-dessus de y.

Le fonctionnement dépond du niveau d'eau dans le réservoir :

- Réservoir plein x=0, y=0, aucune pompe ne fonctionne.
- Réservoir à moitié plein x=1, y=0, une seule pompe fonctionne.
- Réservoir vide x=1, y=1, les deux pompes fonctionnent en même temps, un contact « **c »** à 2 positions précise laquelle des 2 pompes doit fonctionner c=0 (pompe P_1), c=1 (pompeP_2).

1. Etablir les expressions et les schémas correspondant ou fonctionnement de chacune des deux pompes.

Solution :

Entrées : x, y, c

Sorties : P_1, P_2

c	x	y	P_1	P_2
0	0	0	0	0
0	0	1	∅	∅
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	∅	∅
1	1	0	0	1
1	1	1	1	1

c \ xy	**00**	**01**	**11**	**10**
0	0	∅	1	1
1	0	∅	1	0

c \ xy	**00**	**01**	**11**	**10**
0	0	∅	1	0
1	0	∅	1	1

$P_1 = y + \bar{c}x$ $P_2 = y + c\,x$

Nous remarquons d'après les deux exemples traités,que les systèmes combinatoires sont basés sur la théorie de l'Algèbre de Boole et les fonctions logiques. Donc un système combinatoire est défini par une ou plusieurs fonctions logiques où la valeur des sorties est définie en fonction des entrées du système combinatoire.

IV.3. Circuits combinatoires

Nous nous proposons d'étudier plusieurs circuits logiques combinatoires couramment employés dans les systèmes numériques. Parmi les fonctions combinatoires, nous étudierons les éléments suivants **[22]**:

- Codeurs.
- Décodeurs.
- Multiplexeurs.
- Démultiplexeurs.

Toutes ses opérations ainsi que d'autres sont faciles à matérialiser grâce ou circuit intégré MSI (Medium ScaleIntegration).

IV.3.1. Décodeurs

Le décodeur est un circuit logique qui établit la correspondance entre un code d'entrée binaire de n bits et mlignes de sorties pour chacune des combinaisons possibles des entrées, 1 **seule** ligne de sortie est **validée [23].**

Figure IV.2 : Symbole d'un décodeur vrai au niveau haut.

Pour chacune de 2^n combinaisons d'entrées possibles une seule de m sorties passe au niveau **haut** toutes les autres restent au niveau bas.

Remarque : Certains décodeurs sont conçus pour avoir leurs sorties activées au niveau bas (sortie désactivée = niveau haut)**[24]**. Si on applique cette convention, on trouve toujours sur les lignes de sorties des petits ronds.

Figure IV.3 : Symbole d'un décodeur vrai au niveau bas.

IV.3.1.1. Décodeur 3 entrées/8 sorties (1 parmi 8)

C'est un circuit à 3 voies d'entrées et 8 voies desorties, on peut aussi dire que c'est un décodeur ou convertisseur binaire-octal. Parce qu'il établit la correspondance entre un code d'entrée binaire de 3 bits et une sortie parmi 8**[25-26]**.

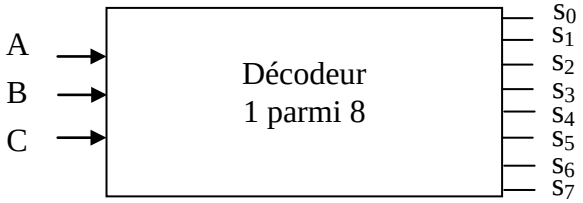


Figure IV.4 : Symbole d'un décodeur de 3 entrées/ 8sorties.

1. **Table de vérité et les équations logiques :**

C	B	A	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

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

2. Logigramme :

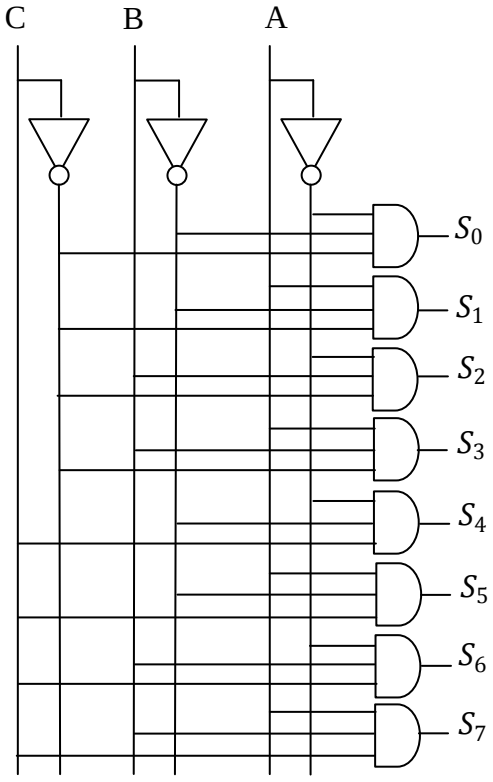


Figure IV.5 : Circuit logique d'un décodeur 1 parmi 8.

VI.3.1.1. Entrées de validation

Les décodeurs sont fréquemment dotés d'une ou plusieurs entrées de validation ENABLE « E » qui servent à valider son fonctionnement**[22]**.

Soit le décodeur de la figure suivante : imaginant qu'une ligne commune raccordée à la 3ème entrée de chaque porte AND. Quand cette ligne est gardée au niveau haut, le décodeur fonctionne normalement et le code d'entrée AB détermine quelle sortie passe au niveau haut. Cette ligne entrée est appelée **entrée de validation**

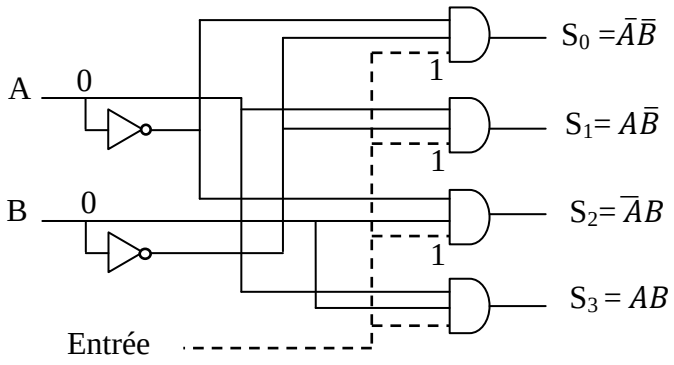


Figure IV.6 : Circuit logique d'un décodeur 1 parmi 4 avec une entrée de validation.

Donc ce décodeur est validé seulement si le signal de validation est au niveau haut, et reste bloqué si ce signal est au niveau bas.

VI.3.1.2. Décodeur en circuit intégré : 74LS138

Ce C.I 74LS138 est appelé décodeur/démultiplexeur3 vers 8qui a trois voies d'entrées (A, B, C) et 8 = 2^3 voies de sorties (décodeur 1 parmi 8). La validité du circuit se fait par les entrées de sélection notées$\bar{G}_1, \bar{G}_2$et G_3. Quand les entrées de validation $\bar{G}_1$ et $\bar{G}_2$ sont à la fois à l'état bas et G_3 à l'état haut, une seule sortie est vraie au niveau Bas (toutes les autres sont à l'état haut) selon le code d'entrée donné. Le décodeur n'est pas validési au moins une des trois entrées de sélection n'est pas active, et toutes les sorties sont aux niveaux haut quel que soit les entrées A, B et C.

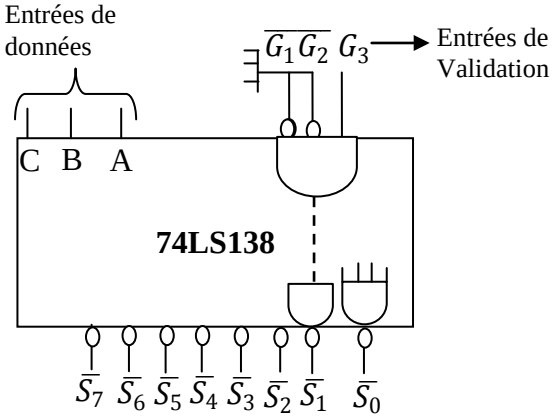


Figure IV.7 : Circuit intégré d'un décodeur 1 parmi 8 : 74LS138.

La table de vérité du décodeur 74LS138 est donnée par le tableau suivant**[22]**, **[27]**.

Entrées						Sorties							
G_3	$\overline{G_2}$	$\overline{G_1}$	C	B	A	$\overline{S_7}$	$\overline{S_6}$	$\overline{S_5}$	$\overline{S_4}$	$\overline{S_3}$	$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$
x	**x**	**1**	x	x	x	1	1	1	1	1	1	1	1
x	**1**	**x**	x	x	x	1	1	1	1	1	1	1	1
0	**x**	**x**	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	**0**
1	0	0	0	0	1	1	1	1	1	1	1	**0**	1
1	0	0	0	1	0	1	1	1	1	1	**0**	1	1
1	0	0	0	1	1	1	1	1	1	**0**	1	1	1
1	0	0	1	0	0	1	1	1	**0**	1	1	1	1
1	0	0	1	0	1	1	1	**0**	1	1	1	1	1
1	0	0	1	1	0	1	**0**	1	1	1	1	1	0
1	0	0	1	1	1	**0**	1	1	1	1	1	1	1

Tableau IV.1 : Table de vérité du décodeur 74LS138.

VI.3.1.3. Décodeur BCD-Décimal

Le décodeur BCD-décimal 74LS42 comporte quatre entrées A, B, C et D (pour coder 0 à 9) et 10 sorties$\overline{S_0}$,$\overline{S_1}$,…,$\overline{S_9}$ validées à l'état bas.

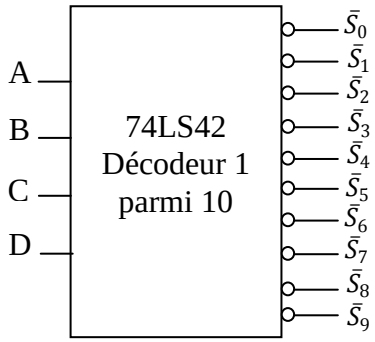


Figure IV.8 : Symbole d'un décodeur BCD-décimal : 74LS42.

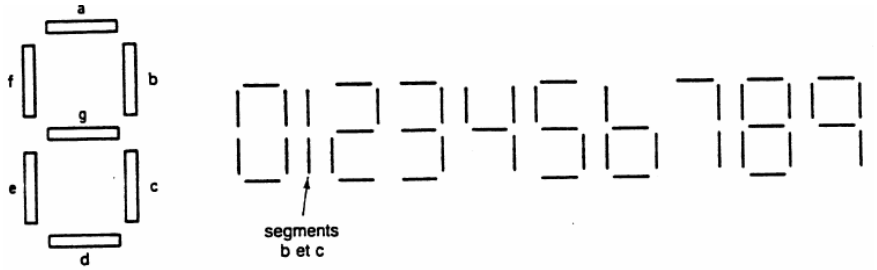
Ce décodeur est conçu de façon que les codes inutilisés (1010,1011, 1100, 1101, 1110, 1111) n'activent aucune des sorties lorsque se trouvent appliquer sur l'entrée. La table de vérité de ce décodeur est donnée par le tableau suivant**[24]** :

DCBA	Sortie active =0
0000	$\bar{s}_0$
0001	$\bar{s}_1$
0010	$\bar{s}_2$
0011	$\bar{s}_3$
0100	$\bar{s}_4$
0101	$\bar{s}_5$
0110	$\bar{s}_6$
0111	$\bar{s}_7$
1000	$\bar{s}_8$
1001	$\bar{s}_9$
1010	Aucune
1011	Aucune
1100	Aucune
1101	Aucune
1110	Aucune
1111	Aucune

Tableau IV.2 : Table de vérité d'un décodeur BCD-décimal : 74LS42.

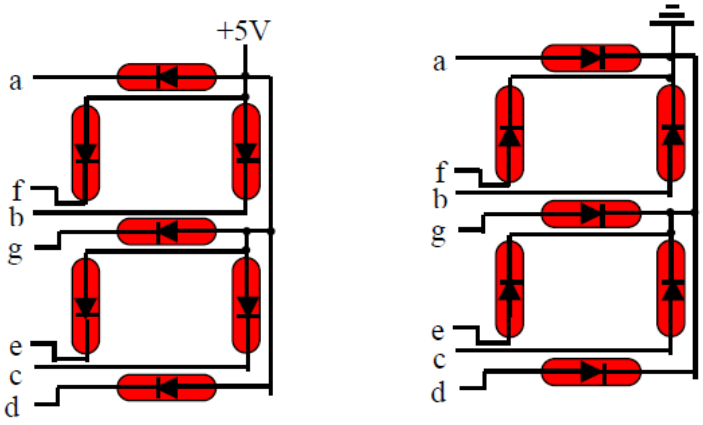
VI.3.1.4. Décodeurs BCD 7 Segments

Un domaine d'application considérable des décodeurs est celui de la conversion de données binaires en une forme se prêtant à un affichage numérique. Un afficheur 7 segments permet de visualiser les chiffres de 0 à 9. Cet afficheur est composé de 7 diodes électroluminescentes(L.E.D) a, b, c, d, e, f et gqui émettent de la lumière quand elles sont traversées par un courant et qu'ils nécessitent en fonction du type d'afficheur (anode commune ou cathode commune) une polarisation spécifique**[28]**.



Dans le cas de l'afficheur à anodes communes, toutes les anodes sont reliées et connectées au potentiel haut (5V), de façon que pour allumer un segment, il faut lui appliquer sur sacathode un niveau bas

Pour l'afficheur à cathodes communes,toutes les cathodes sont reliées et connectées au potentiel bas (masse), de façon à rendre lumineux le segment qui présenteun niveau haut sur son anode**[22]**.



Afficheur à anodes communes Afficheur à cathodes communes

Figure IV.9 : Afficheur 7 segments à anodes et à cathodes communes.

Un décodeur BCD 7 segments accepte en entrée les 4 bits BCD et rend active les sortiesqui vont permettre de faire passer un courant dans les segments qui forment le chiffre décimal correspondant.

Le schéma ci-dessous représente l'association d'un décodeur et d'un afficheur 7 segments. Les anodes sont connectées à travers les résistances pour limiter le courant, les sorties appropriées du décodeur sont les sorties qui sont vraies au niveau bas.

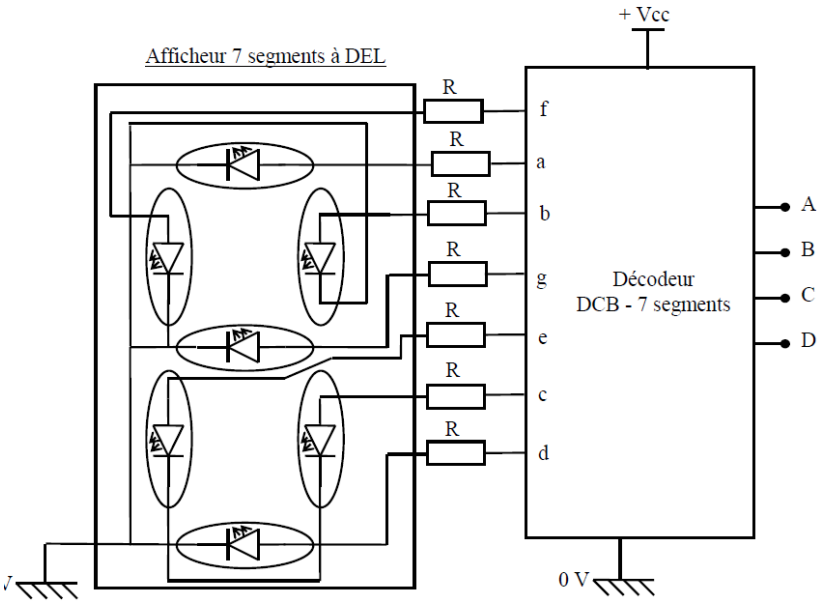


Figure IV.10 : Décodeur BCD 7 Segments.

Le tableau ci-dessous donne la table de vérité détaillant le fonctionnement du décodeur BCD-7 segment permettant l'affichage des différents chiffres décimaux. Les variables d'entréessont codées en BCD sur 4 bits A, B, C, D, la sortie fournit 7 états correspondants à chaque segment d'un afficheur nommés a, b, c, d, e, f et g. Une sortie à 0 représente son extinction, une sortie à 1 représente l'allumage de la LED.

Chiffres	D C B A	a	b	c	d	e	f	g
0	0 0 0 0	1	1	1	1	1	1	0
1	0 0 0 1	0	1	1	0	0	0	0
2	0 0 1 0	1	1	0	1	1	0	1
3	0 0 1 1	1	1	1	1	0	0	1
4	0 1 0 0	0	1	1	0	0	1	1
5	0 1 0 1	1	0	1	1	0	1	1
6	0 1 1 0	0	0	1	1	1	1	1
7	0 1 1 1	1	1	1	0	0	0	0
8	1 0 0 0	1	1	1	1	1	1	1
9	1 0 0 1	1	1	1	0	0	1	1

Tableau IV.3 : Table de vérité d'un décodeur BCD 7 Segments.

VI.3.1.5. Afficheur à cristaux liquide (ACL)

Les afficheurs à cristaux liquide fonctionnent avec des tensions de 3 à 15 V (tensions alternatives) à bas fréquencevarie entre (25Hz -60Hz). Et consomment très peu de courant. La tension alternative nécessaire pour allumer un segment est appliquée entre ce dernier et la plaque arrière commune à tous les segments. Les segments et la plaque arrière forme un condensateur**[28]**.

En pratique on applique les signaux carrés déphasés entre les segments et la plaque ce qui montre la figure suivante :

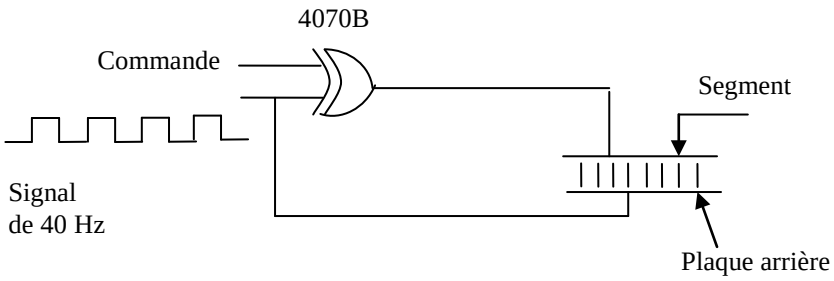


Figure IV.11 : Principe de fonctionnement d'un afficheur à cristaux liquide (ACL).

Une onde carrée de 40 Hz est appliquée à la plaque arrière, de même à l'entrée de la porte OU exclusif de type CMOS, à l'autre entrée de la porte OU exclusif on applique un signal de commande qui donne l'ordre d'allumer ou d'éteindrece segment.

* **Commande = 0**

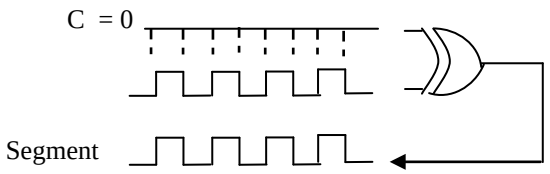


Figure IV.12 : Sortie de la porte OU exclusif si la commande = 0.

La sortie de la porte OU exclusif est une réplique de l'onde carrée 40 Hz ; de sorte que les signaux appliqués à la plaque arrière et le segment sont les mêmes. Donc, il ny a pas une différence de tension et le segment reste éteint.

* **Commande= 1**

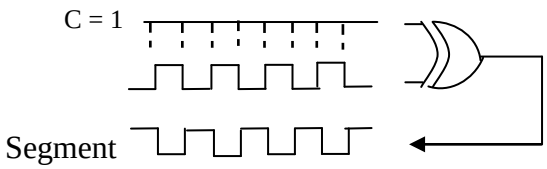


Figure IV.13 : Sortie de la porte OU exclusif si la commande = 1.

Quand l'entréede la commande « C » est haute, la sortie du porte OU exclusif est l'inverse de l'onde carrée de 40Hz et le signal appliqué au segment est en inverse de phase par rapport au signal appliqué à la plaque. Par conséquents, la tension de segment sera alternativement à +5V et -5V par rapport à la plaque, cette tension alternative allume le segment.

Ce principe de base peut être appliqué à une rangée d'afficheur ACL de 7 segments.

Afficheur à ACL à partir d'un décodeur 7 segment 4511 :

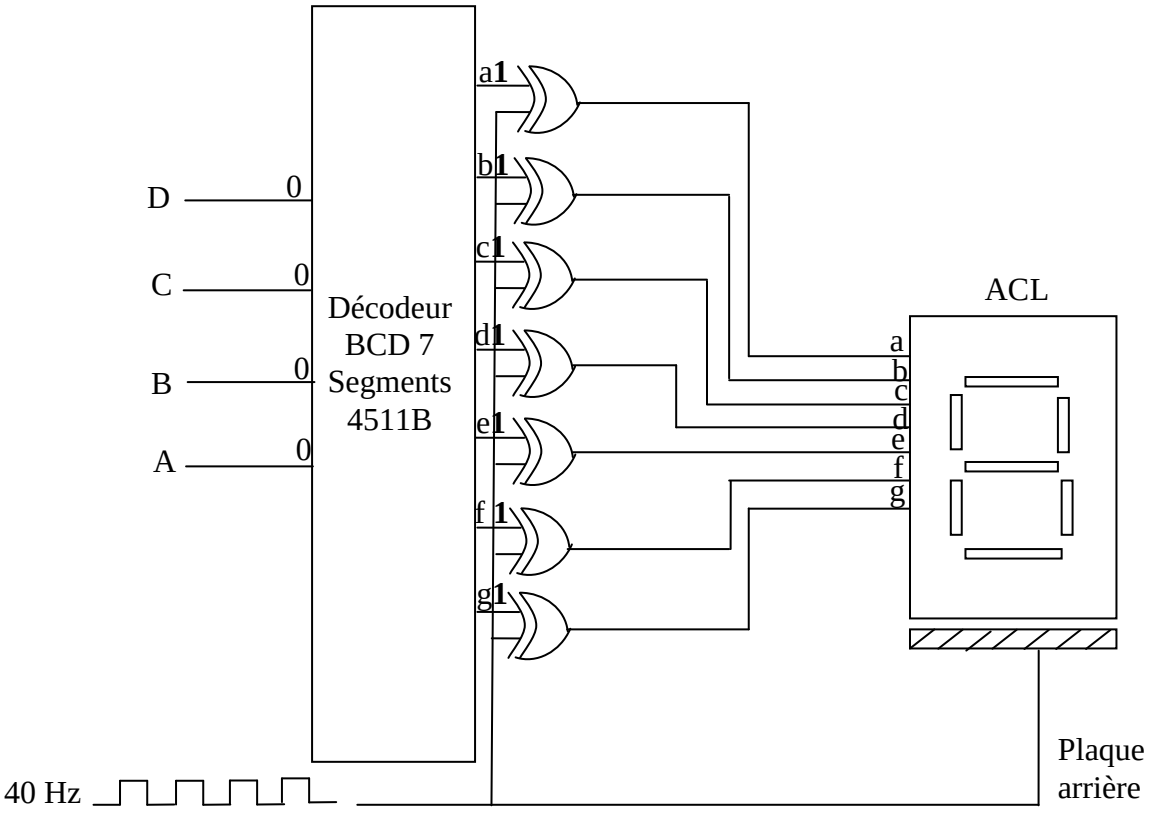


Figure IV.14 : Afficheur à ACL à partir d'un décodeur 7 segment : 4511.

Le décodeur BCD 7 segments CMOS 4511fournit les signaux de commande ou 7 portes de OU exclusif qui excitent 7 segments.Les sorties de 4511 sont vraies aux niveaux hauts puisqu'ilfaut un niveau haut pour allumer un segment.

IV.3.2. Codeurs

Le **codeur** (ou encodeur) possède 2^n entrées et N sorties, dont une seule entrée est activée à la fois parmi 2^n entrées. Il fournit en sortie le numéro de l'indice de l'entrée active codé en binaire sur n bits**[29]**.

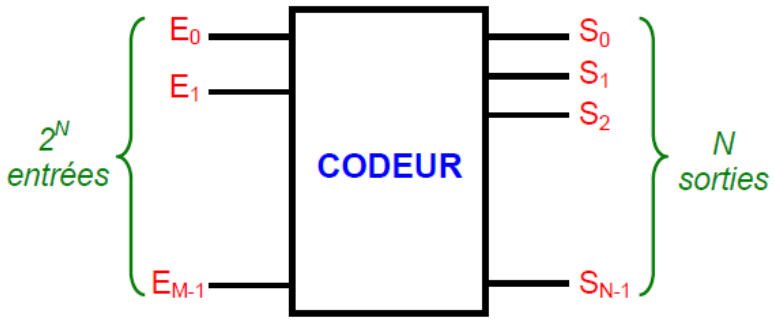


Figure IV.15 : Schéma fonctionnel d'un codeur.

IV.3.2.1. Codeur octal binaire

Un codeur octal binaire à 8 voies d'entrées est produit une représentation de sortie binaire de 3 bits (vraies au niveau bas).

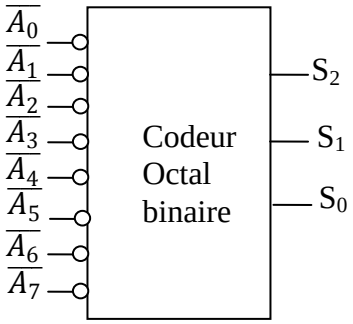


Figure IV.15 : Symbole d'uncodeur Octal binaire.

Table de vérité :

$\overline{A_0}$	$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	$S_2S_1S_0$
Ø	1	1	1	1	1	1	1	000
Ø	0	1	1	1	1	1	1	001
Ø	1	0	1	1	1	1	1	010
Ø	1	1	0	1	1	1	1	011
Ø	1	1	1	0	1	1	1	100
Ø	1	1	1	1	0	1	1	101
Ø	1	1	1	1	1	0	1	110
Ø	1	1	1	1	1	1	0	111

Tableau IV.4 : Table de vérité d'uncodeur Octal binaire.

A partir de cette table de vérité on calcule S_0, S_1, S_2 :

$$S_0 = \overline{\overline{A_1}} + \overline{\overline{A_3}} + \overline{\overline{A_5}} + \overline{\overline{A_7}}$$
$$S_1 = \overline{\overline{A_2}} + \overline{\overline{A_3}} + \overline{\overline{A_6}} + \overline{\overline{A_7}}$$
$$S_2 = \overline{\overline{A_4}} + \overline{\overline{A_5}} + \overline{\overline{A_6}} + \overline{\overline{A_7}}$$

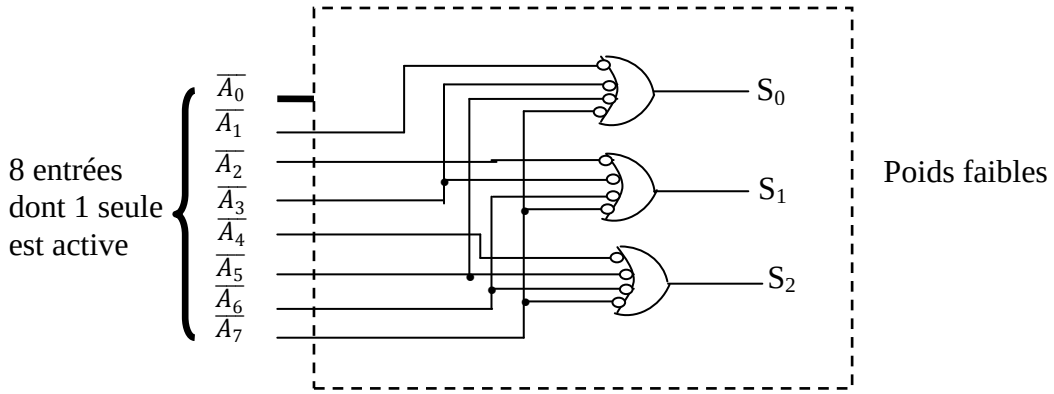


Figure IV.16 : Circuit logique d'un codeur octal/binaire.

Remarque

On remarque que $\overline{A_0}$ n'est plus connecté à une porte logique puisque les sorties de codeur sont normalement à 000 quand aucun des entrées de $\bar{A}_1$ à $\overline{A_7}$ sont aux niveaux bas.

IV.3.2.2. Codeur de priorité

Le codeur de la figure précédente produit des résultats erronés si aux moins deux entrées sont rendues actives simultanément, un codeur de priorité est une version modifiée de codeur élémentaire.Lorsqu'une ou plusieurs lignes d'entrées sont activées, l'encodeur fournit en sortie une valeur binaire correspondant à l'indice de l'entrée active de rang le plus élevé**[30]**. Par exemple, quand les deux entrées $\bar{A}_5$ et $\bar{A}_3$ sont actives en même temps, la réponse donnée en sortie est $S_2S_1S_0$ (101).

IV.3.2.2.1. Codeur de priorité décimal BCD (74147)

Ce circuit possède 9 entrées et 4 sorties donnant les 10 combinaisons binaires (0000 à 1010). Les entrées et les sorties de ce circuit sont actives au niveau bas (L)**[31-32]**. La

présentation de codeur de priorité décimal BCD et sa table de vérité sont données par les figures suivantes :

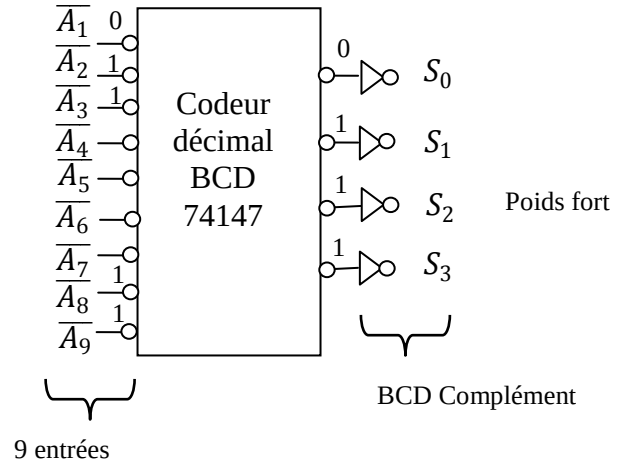


Figure IV.17: Symbole decodeur de priorité décimal BCD (74147).

Table de vérité :

$\overline{A_1}$	$\overline{A_2}$	$\overline{A_3}$	$\overline{A_4}$	$\overline{A_5}$	$\overline{A_6}$	$\overline{A_7}$	$\overline{A_8}$	$\overline{A_9}$	$\overline{S_3}\overline{S_2}\overline{S_1}\overline{S_0}$
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1 1 1 1**
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	**0**	**0 1 1 0**
Ø	Ø	Ø	Ø	Ø	Ø	Ø	**0**	**1**	**0 1 1 1**
Ø	Ø	Ø	Ø	Ø	Ø	**0**	**1**	**1**	**1 0 0 0**
Ø	Ø	Ø	Ø	Ø	**0**	**1**	**1**	**1**	**1 0 0 1**
Ø	Ø	Ø	Ø	**0**	**1**	**1**	**1**	**1**	**1 0 1 0**
Ø	Ø	Ø	**0**	**1**	**1**	**1**	**1**	**1**	**1 0 1 1**
Ø	Ø	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1 1 0 0**
Ø	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1 1 0 1**
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1 1 1 0**

Tableau IV.5 : Table de vérité d'un codeur de priorité décimal BCD (74147).

Les sorties de 74147 sont normalement à 1 quand aucune des entrées ni à son niveau vraie (bas), ceci correspond à la condition d'entrée du chiffre décimal 0. Pour obtenir le code BCD naturel à partir des sorties de 74147, il faut ajouter un inverseur à chacune de sortie.

IV.3.2.2.2. Application : codeur d'interrupteur

Les 10 interrupteurs peuvent être ceux d'un clavier de calculatrice représentant les chiffres "0" à "9". L'ouverture d'un interrupteur entraîne l'application d'un niveau haut sur l'entrée de décodeur par l'intermédiaire d'une résistance de tirage. Lorsqu'un interrupteurest fermé, l'entréecorrespondante est au niveau bas, donc son activation**[33]**.

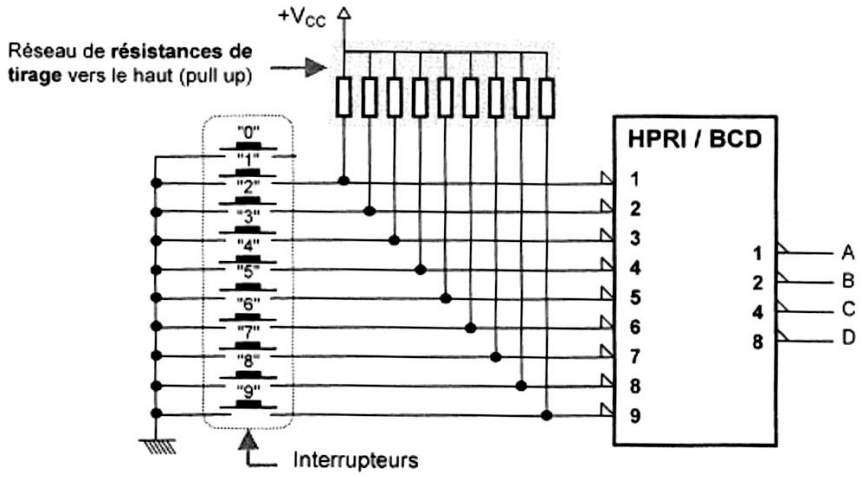


Figure IV.18 : Exemple d'un codeur d'interrupteur à 9 entrées et 4 sorties.

Si maladroitement plusieurs interrupteurs sont enfoncés simultanément, le codeur classique donne un résultat erroné car il ne sait plus quel numéro doit être codé. On utilisera alors dans ce cas un codeur prioritaire.

Un codeur prioritaireest un dispositif réalisant le codage du numéro le plus élevédans le cas où plusieurs interrupteurs seraient actionnés. Si une seule commande est envoyée sur le codeur prioritaire, celui-ci fonctionne comme un codeur classique**[23]**.

IV.3.3. Multiplexeurs (Sélecteurs des données)

Un multiplexeur ou sélecteur de données, appelé encore commutateur numérique commandé **[31]**, est un circuit logique qui possède 2^N entrées d'informations (I_i), N entrées de sélection (S_i) et une sortie unique Z.

.

Sa fonction consiste à pouvoir sélectionner une de ses 2^n entrées d'information et la mettre en communication avec sa sortie à l'aide du code d'adresse de n bits appliqué sur les entrées de sélection.

On pourra de plus trouver une entrée de validation E. Si cette broche est validée, le multiplexeur délivre sur sa sortie Z les informations présentes sur l'entrée de donnée qui est sélectionnée avec les entrées de sélection. Par contre quand cette broche n'est pas validée, la sortie Z est égale à 1 (ou 0), et ceci quelle que soit l'adresse appliquée et le niveau des entrées I_i**[22]**.

La représentation fonctionnelle du multiplexeur est alors donnée par la figure ci-dessous :

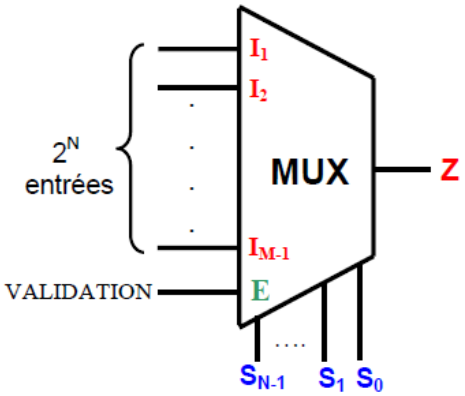


Figure IV.19 :Schéma fonctionnel d'un multiplexeur.

IV.3.3.1. Multiplexeur à 2 entrées

C'est un multiplexeur à 2 entrées logiques A et B, et une sortie logique Z. En fonction d'une variable de sélection notée E, une des 2 entrées se retrouvera à la sortie du multiplexeur **[27]**.

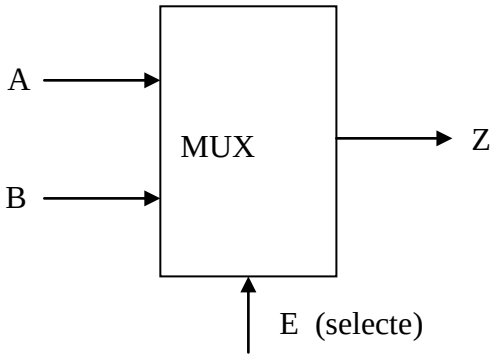


Figure IV.20 :Schéma fonctionnel d'un multiplexeur à 2 entrées.

Table de vérité :

E	A	B	Z	
0	0	0	0	
0	0	1	0	
0	1	0	1	A
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	0	B
1	1	1	1	

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

$$Z = A\bar{E} + BE$$

Tableau IV.6 : Table de vérité et le tableau de Karnaugh d'un multiplexeur à 2 entrées.

MUX à 2 entrées

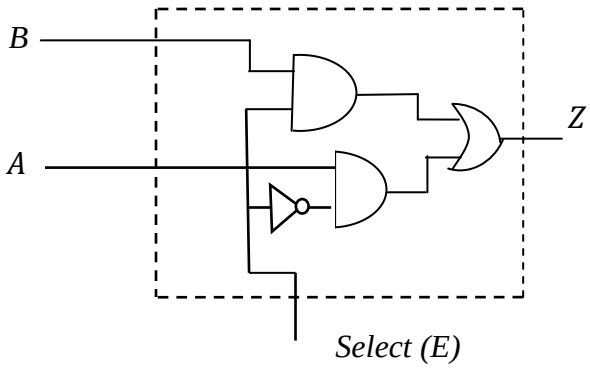


Figure IV.21 :Circuit logique d'un multiplexeur à 2 entrées.

IV.3.3.2.Multiplexeur à 4 entrées

C'est un multiplexeur à 4 entrées logiques (A, B, C, D) et une sortie logique Z. En fonction des variables de sélection S_0S_1, une des 4 entrées se retrouvera à la sortie du multiplexeur.

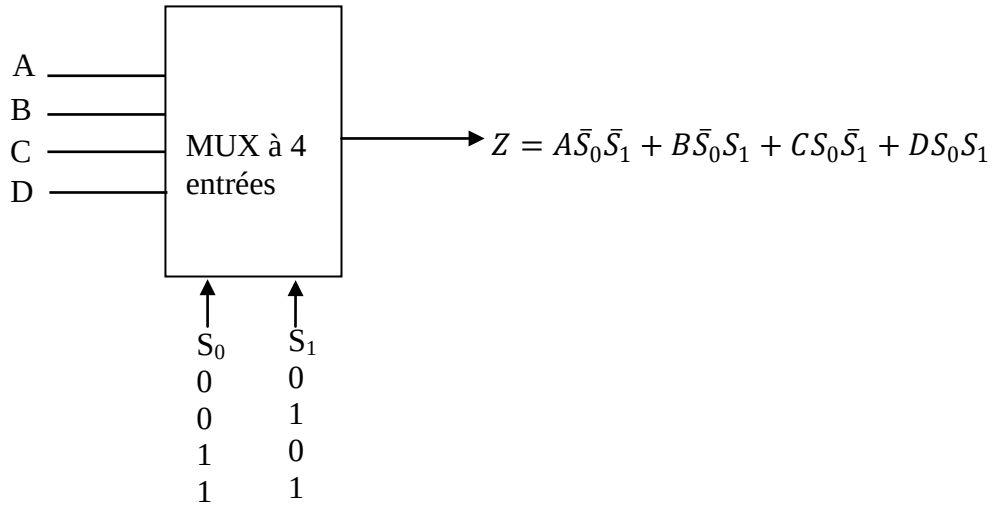


Figure IV.22 :Schéma fonctionnel d'un multiplexeur à 4 entrées.

TV :

S1	S_0	Z
0	0	A
0	1	B
1	0	C
1	1	D

Tableau IV.7 : Table de vérité d'un multiplexeur à 4 entrées.

IV.3.3.3.Multiplexeur 74LS151

Le circuit référencé 74LS151 est un multiplexeur 8 vers 1. Il comporte 8 entrées de données (D_0 à D_7)pouvant être aiguillées vers la sortie Y(ou Y=W) grâce aux entrées de sélection A, B et C. Il englobe aussi d'une entrée de validation notée G permettant d'autoriser le fonctionnement du circuit (elle est vraie au niveau logique bas) **[24]**, **[34]**.

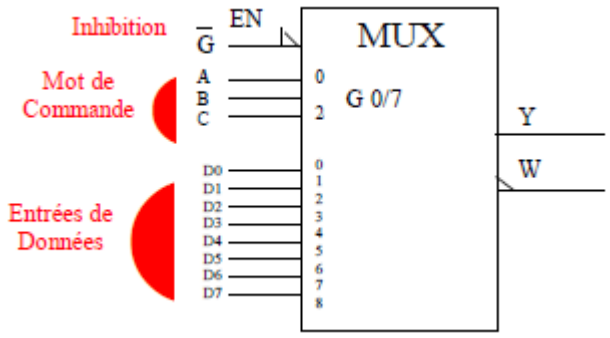


Figure IV.23 :Schéma fonctionnel du multiplexeur74LS151.

Table de vérité :

$\bar{G}$	C	B	A	Y	W
1	∅	∅	∅	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	0	1	D_1	$\overline{D_1}$
0	0	1	0	D_2	$\overline{D_2}$
0	0	1	1	D_3	$\overline{D_3}$
0	1	0	0	D_4	$\overline{D_4}$
0	1	0	1	D_5	$\overline{D_5}$
0	1	1	0	D_6	$\overline{D_6}$
0	1	1	1	D_7	$\overline{D_7}$

Tableau IV.8 : Table de vérité du multiplexeur 74LS151.

Quand $\bar{G} = 0$, les entrées de sélection A B Cchoisissent une entrée des données (D_0 à D_7) dans les valeurs se retrouvent sur la sortie Y.

Quand $\overline{G} = 1$, le multiplexeur est invalidé de sorte que Y =0 quel que soit le code d'entrée de sélection.

$$Y = D_0\bar{C}\bar{B}\bar{A}G + D_1\bar{C}\bar{B}AG + D_2\bar{C}B\bar{A}G + D_3\bar{C}BAG + D_4C\bar{B}\bar{A}G + D_5C\bar{B}AG + D_6CB\bar{A}G + D_7CBAG$$

IV.3.4. Démultiplexeurs

Le démultiplexeur réalise l'opération inversedumultiplexeur : il aiguille vers une parmi plusieurs sorties, les informations provenant d'une même source.

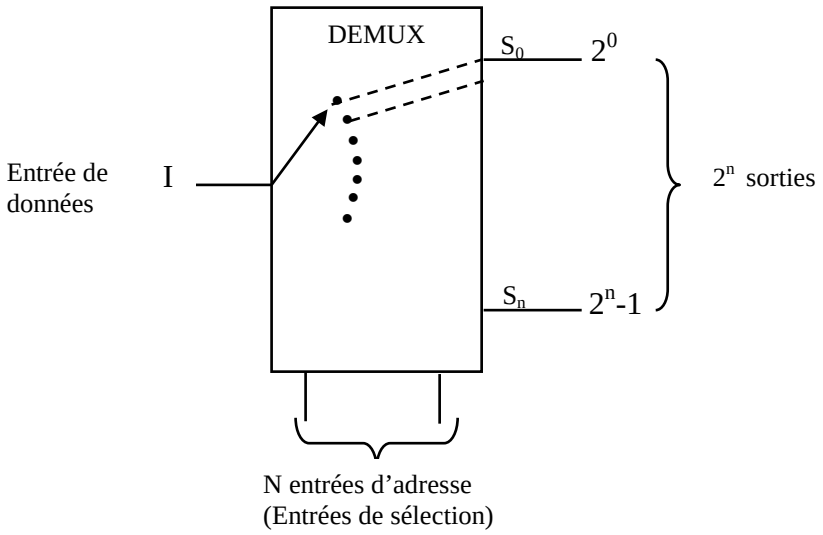


Figure IV.24 :Schéma fonctionnel d'undémultiplexeur.

Le démultiplexeur dispose d'une seule entrée de donnée I, de n entrées d'adresse et de 2^n sorties, où une seule sortie est active à la fois. Lorsqu'on applique sur les lignes d'adresse le code binaire i, l'entrée I est reliée à la sortie S_i **[35]**.

IV.3.4.1.Démultiplexeur à 8 sorties

La table de vérité d'un démultiplexeur 1→8 est présentée dans le Tableau ci-dessous dans le cas où les sorties non actives sont à 0.

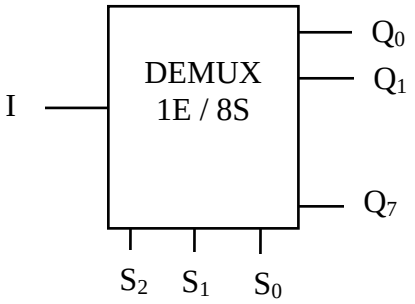


Figure IV.25 :Schéma fonctionnel d'undémultiplexeur à 8 sorties.

Tablede vérité :

S_2	S_1	S_0	Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

Tableau IV.9 : Table de vérité d'undémultiplexeur à 8 sorties.

Le principe du démultiplexeur est assez proche de celui d'un décodeur ayant une entrée de validation. En effet, l'entrée de validation du décodeur peut servir d'entrée de données I pour le démultiplexeur, et les entrées A, B, C jouent le rôle d'entrées d'adresse ceux qui permettent alors de transformer le décodeur en démultiplexeur. C'est pour cette raison les fabricants de circuits intégrés donnent souvent le nom de décodeur/démultiplexeur à ces dispositifs **[22]**.

Décodeur + entrée de validation = Démux (I l'entrée de données)

La figure ci-dessous nous montre comment nous pouvons utiliser le C.I 74138 (c'est un décodeur un parmi huit) comme démultiplexeur.

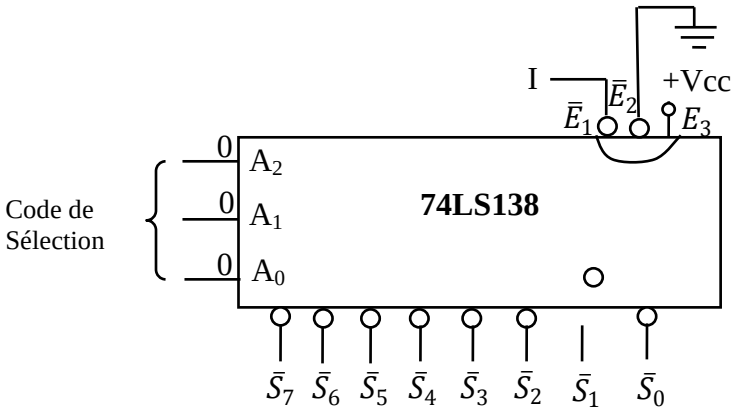
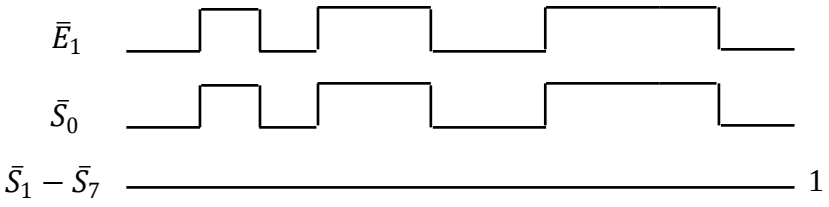


Figure IV.26 :Circuit intégré d'un décodeur/ Démultiplexeur 74138.

Dans le circuit de la figure ci-dessus,les entrées A_0, A_1, A_2 jouent le rôle d'entrées d'adresse, tandis que les lignes d'entrées $\overline{E_2}$et E_3 sont validées. L'entrée de validation $\overline{E_1}$ est utilisée comme l'entrée de donnée I.

Par exemple, les formes d'ondes pour $A_2A_1A_0 = 000$:



Références

[1] KARIM Mohammed, « Chapitre 1: Systèmes de Numération et Codes », Faculté des Sciences de Fès.

[2] « Numération et codage et l'information », 1ère STI, Electronique, Ecole Modèle d'Electronique.

[3] Mr KHATORY, «Initiation informatique I, Système de numération », Ecole Supérieure deTechnologie de Fès, Filière Génie Industriel et Maintenance.

[4] TRABELSI Hichem, «Circuits logiques combinatoires, Systèmes de numération et codes », Université Virtuelle de Tunis.

[5] BTS IG 1ère année AMSI, « Les systèmes de numération- Cours et TD », page 1- 8.

[6] Mr M. HALAILI– INSFP /SBA, «Support de cours : Systèmes de numération», Techniques numériques, Février 2006.

[7] M. BERNARD, «Systèmes de numération and codage », édité le 03/09/2008, page 1-11.

[8] http://www.commentcamarche.net/contents/97-controle-d-erreur-crc.

[9] Didier Müller, «Chapitre 3 : Codage de l'information»,août 2013.

[10]M . HALAILI-INSFP /SBA,«Algèbre de Boole et portes logiques », Support de cours : Techniques numériques, Février 2012.

[11] C.ALEXANDRE, « Circuits numériques : 1ère partie »,Polycopié de cours : Electronique A4, lundi 19 janvier 2004.

[12] Daniel Etiemble, «Algèbre de Boole et fonctions booléennes», S4-CLM/Notes de cours.

[13] De Morgan Augustus (1806–1871): Logicien et Mathématicien Anglais (FormalLogic 1847; Trigonometry and Double Algebra 1849), «Systèmes logiques-Logique combinatoire ».

[14] http://www.fsr.um5a.ac.ma/cours/informatique/elbenani/algebre_boole.pdf

[15] Semchedine Moussa, «Chapitre 3 : L'algèbre de Boole », 2012.

[16] D. DUBOIS, « La fonction logique combinatoire », http://dado59.free.fr/www2/cours/automatique/fonctionlogcombi.pdf

[17]http://www.lifl.fr/~dekeyser/S3AE/support%20cours/karnaugh.pdf

[18] TS CRSA, Les automatismes, Algèbre logique, Lycée L.RASCOL 10, Rue de la RépubliqueBP 218. 81012 ALBI CEDEX.

[19] Laurent MURA, «La Logique Combinatoire », IUT de Colmar -Département GTR -1ière année.

[20]http://bannaladi.fr/cours/Traitement/5_KARNAUGH.pdf.

[21] Serge Iovleff , «Mathématiques Pour l'Informatique I : Algèbre de Boole et Logique», 14 janvier 2005.

[22] TRABELSI Hichem, «Circuits logiques combinatoires, Fonctions combinatoires», Université Virtuelle de Tunis.

[23] www.gecif.net,« Les décodeurs – Les transcodeurs », Génie Electrique.

[24] http://ensa-mecatronique.e-monsite.com/medias/files/cours-elec-num-3.pdf.

[25] http://genie-indus.e-monsite.com/medias/files/tp2.pdf

[26] David Bouchet, « Chapitre 1 : Les circuits logiques», Architecture des ordinateurs – MLI336 – Paris 5 – 2013/2014, Version du 27/09/2013.

[27] M.TAYARI Lassaad, « Support de cours systèmes logiques», Ministère de l'enseignement supérieur Institut Supérieur des Etudes Technologiques de Gabès, Technologue à ISET GABES, Année Universitaire 2013-2014.

[28] http://www2.ac-lyon.fr/lyc01/cotiere/IMG/pdf/CirNum_Prof.pdf

[29] Michel Jézéquel, « ELP 304 : Cours 2 Circuits combinatoires», Département Electronique, Telecom Bretagne. http://public.enst-bretagne.fr/~douillar/ELP304/Cours2.pdf.

[30] Michel Krammer, « Logique combinatoire», ENSL1 : Electronique numérique et synthèse logique, Dépt : GEII, 2010/2011.

[31] Professeur MTIBAA Abdellatif, «Electronique Numérique», Génie Electrique, Année Universitaire 2011/2012.

http://fr.slideshare.net/abdellatifmtibaa/cours-mtibaa-electroniquenumerique2012.

[32] EPMI Cergy 1AING, «Electronique numérique, Logique combinatoire et multiplexage»,

http://karlaoui.free.fr/Site%20Epmi/Electronique_num%C3%A9rique/Cours/3.Logique_combinatoire_et_multiplexage.pdf

[33] CPGE TSI, «Le comportement des systèmes logiques combinatoires»,Centre d'Intérêt 3 : TRAITER l'information, Sciences Industrielles pour l'Ingénieur, Lycée P.-P. Riquet – St-Orens de Gameville.

[34]https://pigeon.users.greyc.fr/Enseignements/doc_pedagogique/logique_combinatoire/circuit_logiques.pdf

[35] Ecole polytechnique, CiP1, «Travaux pratiques informatique industrielle, Logique combinatoire », Université de Nice-Sophia Antipolis, Département Electronique.

[36] « Cours logique séquentielle : Les bascules », Lycée Jules Ferry – Versailles – CRDEMA, 2007-2008.

[37] F DAUCHY/S GARCIA/F MANDIN, «Les mémoires élémentaires: les bascules RS, D et JK», Lycée Mireille GRENET-COMPIEGNE.

[38] El-M. HARKAT, «Les bascules et leurs applications»,Ecoles d'Ingénieurs et des Départements Universitaires de technologie, Edition 2011.

[39] Etienne Messerli, Yves Meyer, « Electronique numérique 2ème tome Systèmes séquentiels », Haute Ecole Spécialisée de Suisse occidentale, Version 1, Décembre 2014.

[40] S. Bolay, «Les Circuits Bistables (flip-flop) », Electro/Info 1, CFPs-EMVs, 2009.

[41] http://genelec.santonum.free.fr/_fichiers/s1-Electronique/s13-Logique-cablee/Cours-Seq-Bascules.i1351.pdf.

[42] Pascal Bordellon, « Logique Séquentielle (prof)», Lycée clément ADER (77) –Fichier téléchargé sur : www.lelectronique.com.