Ministère de l'enseignement supérieur et de la recherche scientifique

SYSTEMES DE NUMERATION ET CODAGE DES INFORMATIONS

Chapitre 1

SYSTEMES DE NUMERATION ET CODAGE DES INFORMATIONS

1. OBJECTIFS

- Traiter en détails les différents systèmes de numération : systèmes décimal, binaire, octal et hexadécimal ainsi que les méthodes de conversion entre les systèmes de numération.
- Etudier plusieurs codes numériques tels que les codes DCB, GRAY .

2. SYSTEMES DE NUMERATION

Pour qu'une information numérique soit traitée par un circuit, elle doit être mise sous forme adaptée à celui-ci. Pour cela Il faut choisir un système de numération de base B (B un nombre entier naturel ≥ 2)

De nombreux systèmes de numération sont utilisés en technologie numérique. Les plus utilisés sont les systèmes : Décimal (base 10), Binaire (base 2), , Octal (base 8) et Hexadécimal (base 16).

Le tableau ci-dessous représente un récapitulatif sur ces systèmes :

Décimal	Binaire	Octal	Hexadécimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

2.1 Représentation polynomiale(formule generale)

Tout nombre **N** peut se décomposer en fonction des puissances entières de la base de son système de numération. Cette décomposition s'appelle la forme polynomiale du nombre **N** et qui est donnée par :

$$N=a_nB^n + a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \ldots + a_2B^2 + a_1B^1 + a_0B^0$$

- **B** : Base du système de numération, elle représente le nombre des différents chiffres qu'utilise ce système de numération.
- a_i : un chiffre (ou digit) parmi les chiffres de la base du système de numération.
- **i** : rang du chiffre a_i.

2.2 Système décimal (base 10)

Le système décimal comprend 10 chiffres qui sont {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} c'est un système qui s'est imposé tout naturellement à l'homme qui possède 10 doigts.
Ecrivons quelques nombres décimaux sous la forme polynomiale :

Exemples :

$(5462)_{10}= 5*10^3 + 4*10^2 + 6*10^1 + 2*10^0$

$(239.537)_{10}= 2*10^2 + 3*10^1 + 9*10^0 + 5*10^{-1} + 3*10^{-2} + 7*10^{-3}$

2.3 Système binaire (base 2)

Dans ce système de numération il n'y a que deux chiffres possibles {0, 1} qui sont souvent appelés bits « binary digit ». Comme le montre les exemples suivants, un nombre binaire peut s'écrire sous la forme polynomiale.

Exemples :

$(111011)_2= 1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

$(10011.1101)_2= 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4}$

2.4 Système tétral (base 4)

Ce système appelé aussi base 4 comprend quatre chiffres possibles {0, 1, 2, 3}. Un nombre tétral peut s'écrire sous la forme polynomiale comme le montre les exemples suivant :

Exemples :

$(2331)_4= 2*4^3 + 3*4^2 + 3*4^1 + 1*4^0$

$(130.21)_4= 1*4^2 + 3*4^1 + 1*4^0 + 2*4^{-1} + 1*4^{-2}$

Système Octal (base 8)

Le système octal ou base 8 comprend huit chiffres qui sont {0, 1, 2, 3, 4, 5, 6, 7}. Les chiffres 8 et 9 n'existent pas dans cette base. Ecrivons à titre d'exemple, les nombres 4527_8 et 1274.632_8 :

Exemples :

$(4527)_8= 4*8^3 + 5*8^2 + 2*8^1 + 7*8^0$

$(1274.632)_8= 1*8^3 + 2*8^2 + 7*8^1 +4*8^0+ 6*8^{-1} + 3*8^{-2} + 2*8^{-3}$

2.5 Système Hexadécimal (base 16)

Le système Hexadécimal ou base 16 contient seize éléments qui sont {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}. Les chiffres A, B, C, D, E, et représentent respectivement 10, 11, 12, 13, 14 et 15.

Exemples :

$(3256)_{16}= 3*16^3 + 2*16^2 + 5*16^1 + 6*16^0$
$(9C4F)_{16}= 9*16^3 + 12*16^2 + 4*16^1 + 15*16^0$
$(A2B.E1)_{16}= 10*16^2 + 2*16^1 + 11*16^0 +14*16^{-1}+ 1*16^{-2}$

3. CHANGEMENT DE BASE

Il s'agit de la conversion d'un nombre écrit dans une base $\mathbf{B_1}$ à son équivalent dans une autre base $\mathbf{B_2}$

3.1 Conversion d'un nombre N de base B en un nombre décimal

La valeur décimale d'un nombre **N**, écrit dans une base **B**, s'obtient par sa forme polynomiale décrite précédemment.

Exemples :

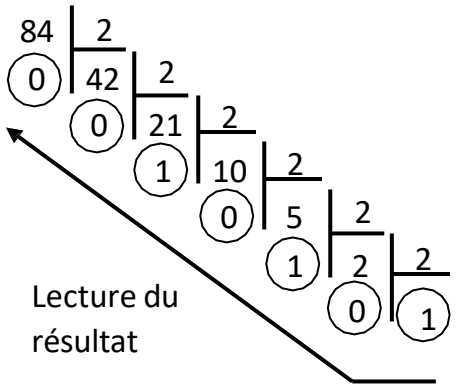
$(1011101)_2= 1*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1+ 1*2^0=(93)_{10}$
$(231102)_4= 2*4^5 + 3*4^4 + 1*4^3 + 1*4^2 + 0*4^1+ 2*4^0=(2898)_{10}$
$(7452)_8= 7*8^3 + 4*8^2 + 5*8^1+ 2*8^0=(3882)_{10}$
$(D7A)_{16}= 13*16^2 + 7*16^1 + 10*16^0 =(3450)_{10}$

3.1.1 Conversion d'un nombre décimal entier

Pour convertir un nombre décimal entier en un nombre de base **B** quelconque, il faut faire des divisions entières successives par la base **B** et conserver à chaque fois le reste de la division. On s'arrête l'lorsqu'on obtient un résultat inferieur à* la base **B**. Le nombre recherche **N** dans la base **B** s'écrit de la gauche vers la droite en commençant par le dernier résultat allant jusqu'au premier reste.

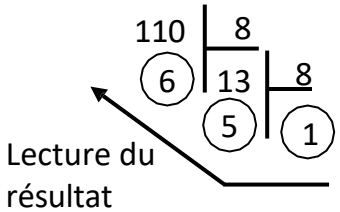
Exemples :

➲ $(84)_{10}=(\ ?\ )_2$



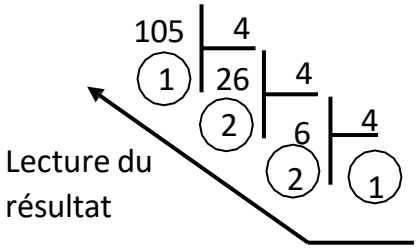
$(84)_{10}=(1010100)_2$

➲ $(110)_{10}=(\ ?\ )_8$



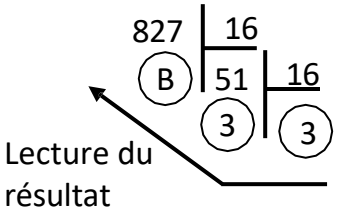
$(110)_{10}=(156)_8$

➲ $(105)_{10}=(\ ?\ )_4$



$(105)_{10}=(1221)_4$

➲ $(827)_{10}=(\ ?\ )_{16}$



$(827)_{10}=(33B)_8$

3.1.2 Autres conversions

Pour faire La conversion d'un nombre d'une base quelconque $\mathbf{B_1}$ vers une autre base $\mathbf{B_2}$ il faut passer par la base **10**. Mais si la base $\mathbf{B_1}$ et $\mathbf{B_2}$ s'écrivent respectivement sous la forme d'une puissance de 2 on peut passer par la base 2 (binaire) :

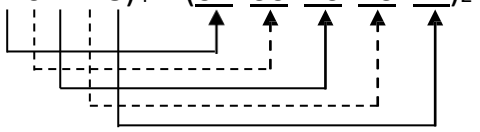
Base tétrale (base 4) : $4=2^2$ chaque chiffre tétral se convertit tout seul sur 2 bits. Base

octale (base 8) : $8=2^3$ chaque chiffre octal se convertit tout seul sur 3 bits.

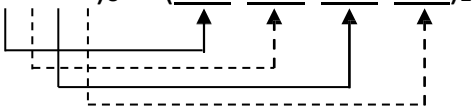
Base hexadécimale (base 16) : $16=2^4$ chaque chiffre hexadécimal se convertit tout seul sur 4 bits.

Exemples :

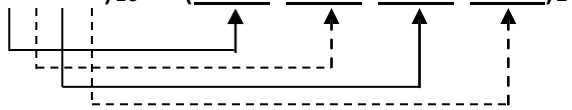
⊙ $(1\ 0\ 2\ 2\ 3)_4 = (\underline{01}\ \underline{00}\ \underline{10}\ \underline{10}\ \underline{11})_2$



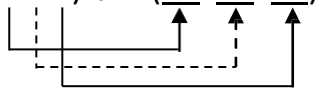
⊙ $(6\ 5\ 3\ 0)_8 = (\underline{110}\ \underline{101}\ \underline{011}\ \underline{000})_2$



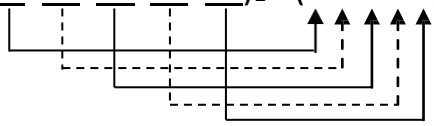
⊙ $(9\ A\ 2\ C)_{16} = (\underline{1001}\ \underline{1010}\ \underline{0010}\ \underline{1100})_2$



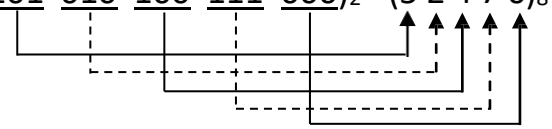
⊙ $(7\ E\ 9)_{16} = (\underline{13}\ \underline{32}\ \underline{21})_4$



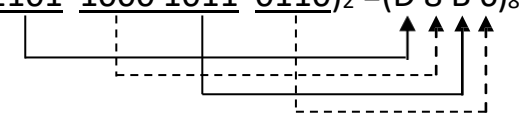
- $(\underline{11}\ \underline{10}\ \underline{01}\ \underline{00}\ \underline{10})_2 = (3\ 2\ 1\ 0\ 2)_4$



- $(\underline{101}\ \underline{010}\ \underline{100}\ \underline{111}\ \underline{000})_2 = (5\ 2\ 4\ 7\ 0)_8$



- $(\underline{1101}\ \underline{1000}\ \underline{1011}\ \underline{0110})_2 = (D\ 8\ B\ 6)_8$



4. CODAGE DE L'INFORMATION

Le codage de l'information est nécessaire pour le traitement automatique de celui-ci. Parmi les codes les plus rencontrés, autre que le code binaire naturel on cite le code **DCB**, le code **GRAY**, le code **p** parmi **n**, le code

4.1 Les codes numériques

4.1.1 Le code binaire Naturel

C'est une représentation numérique des nombres dans la base 2

Décimal	Code Binaire Naturel			
	a_3	a_2	a_1	a_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

➲ Ce code présente l'inconvénient de changer plus qu'un seul bit quand on passe d'un nombre à un autre immédiatement supérieur.

4.1.2 Le code binaire réfléchi (code GRAY)

Son intérêt réside dans des applications d'incrémentation où un seul bit change d'état à chaque incrémentation.

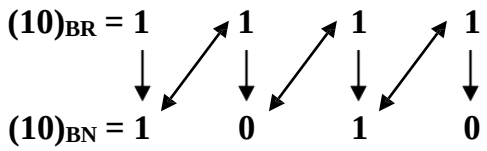
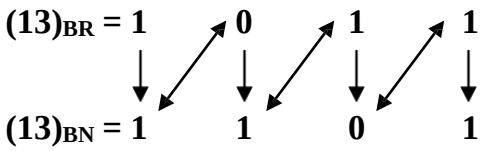
Décimal	Code Binaire Naturel				Code Binaire Réfléchi			
	a_3	a_2	a_1	a_0	a'_3	a'_2	a'_1	a'_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Remarques :

⊙ Conversion du Binaire Naturel vers le Binaire Réfléchi : il s'agit de comparer les bits b_{n+1} et le bit b_n du binaire naturel, le résultat est b_r du binaire réfléchi qui vaut 0 si $b_{n+1}=b_n$ ou 1 sinon. Le premier bit à gauche reste inchangé.

$(6)_{10}=(?)_{BR}$	$(10)_{10}=(?)_{BR}$
$(6)_{BN}$ = 1 ⟷ 1 ⟷ 0 ↓ ↓ ↓ $(6)_{BR}$ = 1 0 1 $(6)_{10}=(110)_{BN}=(101)_{BR}$	$(10)_{BN}$ = 1 ⟷ 0 ⟷ 1 ⟷ 0 ↓ ↓ ↓ ↓ $(10)_{BR}$ = 1 1 1 1 $(10)_{10}=(1010)_{BN}=(1111)_{BR}$

⊙ Conversion du Binaire Réfléchi vers le Binaire Naturel: il s'agit de comparer le bit b_{n+1} du binaire naturel et le bit b_n du binaire réfléchi le résultat est b_n du binaire naturel qui vaut 0 si $b_{n+1}=b_n$ ou 1 sinon. Le premier bit à gauche reste inchangé.

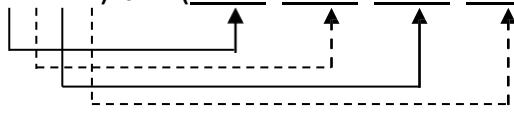
$(10)_{10}=(?)_{BN}$	$(13)_{10}=(?)_{BN}$
	
$(10)_{10}=(1111)_{BR}=(1010)_{BN}$	$(13)_{10}=(1011)_{BR}=(1101)_{BN}$

5.1.2 Le code décimal codé binaire (code DCB)

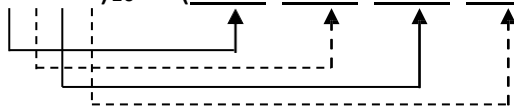
Sa propriété est d'associer 4 bits représentent chaque chiffre en binaire naturel. L'application la plus courante est celle de l'affichage numérique ou chaque chiffre est associé à un groupe de 4 bits portant le code DCB.

Exemples :

- $(9\ 4\ 2\ 7)_{10} = (\underline{1001}\ \underline{0100}\ \underline{0010}\ \underline{0111})_{DCB}$

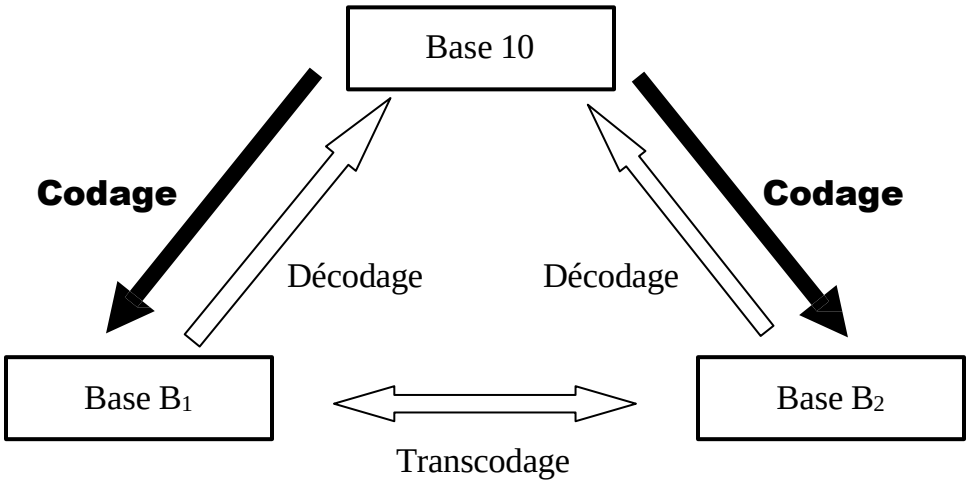


- $(6\ 8\ 0\ 1)_{10} = (\underline{0110}\ \underline{1000}\ \underline{0000}\ \underline{0001})_{DCB}$



4.2 Le Transcodage

Une des applications liée au codage des informations est le passage d'un code à un autre. Cette opération est appelée transcodage :



- Le codage des informations se fait au moyen d'un circuit combinatoire appelé **Codeur**.
- Le décodage des informations se fait au moyen d'un circuit combinatoire appelé **Décodeur**.
- Un transcodeur est un **Décodeur** associé à un **Codeur**.

Chapitre 2

ALGEBRE DE BOOLE ET FONCTIONS LOGIQUES

1. OBJECTIFS

- Etudier les règles et les théorèmes de l'algèbre de Boole.
- Comprendre le fonctionnement des portes logiques.

2. LES VARIABLES ET LES FONCTIONS LOGIQUES

2.1 Les variables logiques

Une variable logique est une grandeur qui ne peut prendre que deux états logiques. Nous les symbolisons par 0 ou 1.

Exemples :

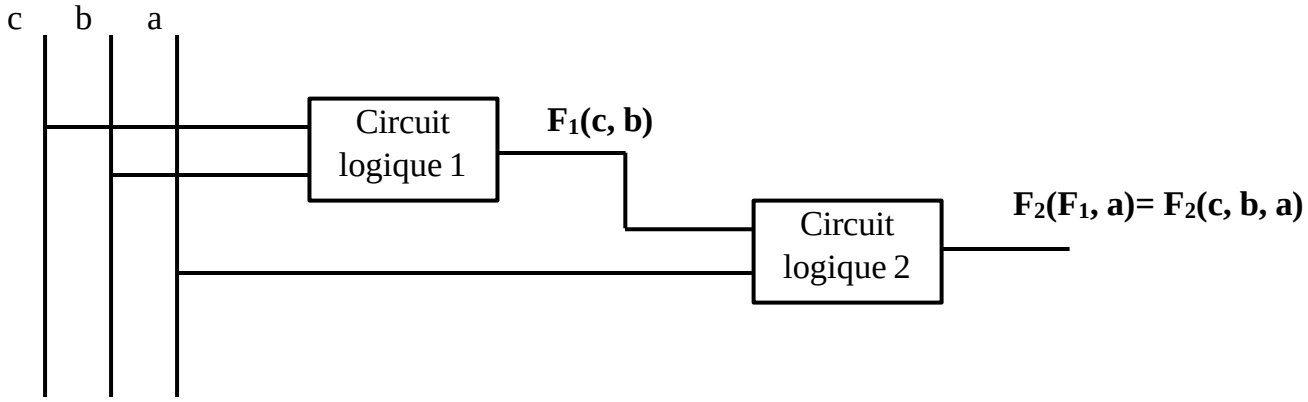
- Un interrupteur peut être soit fermée (1 logique), soit ouvert (0 logique). Il possède donc 2 états possibles de fonctionnement.
- Une lampe possède également 2 états possibles de fonctionnement qui sont éteinte (0 logique) ou allumée (1 logique).

2.2 Les fonctions logiques

Une fonction logique est une variable logique dont la valeur dépend d'autres variables,

- Le fonctionnement d'un système logique est décrit par une ou plusieurs propositions logiques simples qui présentent le caractère binaire "VRAI" ou "FAUX".
- Une fonction logique qui prend les valeurs 0 ou 1 peut être considérée comme une variable binaire pour une autre fonction logique.
- Pour décrire le fonctionnement d'un système en cherchant l'état de la sortie pour toutes les combinaisons possibles des entrées, on utilisera « La table de vérité ».

Exemple :



3. LES OPERATIONS DE BASE DE L'ALGEBRE DE BOOLE ET LES PROPRIETES ASSOCIEES

L'algèbre de Boole est un ensemble de variables à deux états {0 et 1} dites aussi booléennes muni de 3 operateurs élémentaires présentés dans le tableau suivant :

Opération logique	**Addition**	**Multiplication**	**Inversion**
	OU	**ET**	**NON**
Notation Algébrique	A OU B=A+B	A ET B=A.B	Non A=$\overline{A}$
Table de vérité	A B A+B 0 0 0 0 1 1 1 0 1 1 1 1	A B A.B 0 0 0 0 1 0 1 0 0 1 1 1	A NON A 0 1 1 0

3.1 Les propriétés des opérations de base

Quelques propriétés remarquables sont à connaitre :

Fonctions	**OU**	**ET**	**Commentaires**
1 variable	A+A=A	A.A=A	Idempotence
	A+1=1	A.0=0	Elément absorbant
	A+0=A	A.1=A	Elément Neutre
	A+$\overline{A}$=1	A.$\overline{A}$=0	Complément
	$\overline{\overline{A}}$=A		Involution

Fonctions	OU	ET	Commentaires
2 variables	A+B=B+A	A.B=B.A	Commutativité
3 variables	A+(B+C)=(A+B)+C =A+B+C	A.(B.C)=(A.B).C =A.B.C	Associativité
	A+B.C=(A+B).(A+C)	A.(B+C)=A.B+A.C	Distributivité

3.2 Les théorèmes de l'algèbre de Boole

Pour effectuer tout calcul Booléen, on utilise, en plus des propriétés, un ensemble de théorèmes :

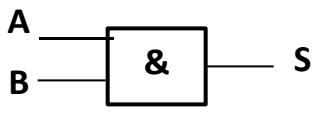
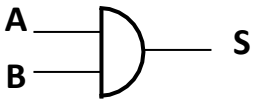
Théorèmes	OU	ET
De DEMORGAN	$\overline{A+B}=\overline{A}.\overline{B}$	$\overline{A.B}=\overline{A}+\overline{B}$
	Ce théorème peut être généralisé à plusieurs variables	
	$\overline{A+B+\ldots+Z}=\overline{A}.\overline{B}.\ldots.\overline{Z}$	$\overline{A.B.\ldots.Z}=\overline{A}+\overline{B}+\ldots+\overline{Z}$
D'absorption	A+AB=A	A.(A+B)=A
D'allègement	$A+\overline{A}B=A+B$	$A.(\overline{A}+B)=A.B$
	$A.B+\overline{A}C+BC=AB+\overline{A}C$	

4. MATERIALISATION DES OPERATEURS LOGIQUES

4.1 Les portes logiques de base

Les portes logiques sont des circuits électroniques dont les fonctions de transfert (relations entre les entrées et les sorties) matérialisant les opérations de base appliquées à des variables électriques.

4.1.1 La porte ET (AND)

Symbole logique		**Equation**	**Circuit intégré**
Symbole International (CEI)	Symbole Européen (MIL)		
		S=A.B	TTL : 7408 CMOS : 4081

Si $\mathbf{V_0}$ représente le niveau BAS de tension (état **0**) et $\mathbf{V_1}$ représente le niveau HAUT (état **1**), on relève en sortie du circuit les tensions données dans la table de fonctionnement et on en déduit la table de vérité.

Table de fonctionnement		
$\mathbf{V_A}$	$\mathbf{V_B}$	$\mathbf{V_S}$
V_0	V_0	V_0
V_0	V_1	V_0
V_1	V_0	V_0
V_1	V_1	V_1

Table de vérité		
A	**B**	**S**
0	0	0
0	1	0
1	0	0
1	1	1

4.1.2 La porte OU (OR)

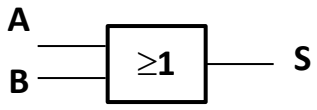
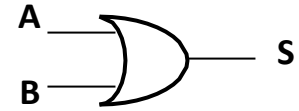
Symbole logique		**Equation**	**Circuit intégré**
Symbole International (CEI)	Symbole Européen (MIL)		
		S=A+B	TTL : 7432 CMOS : 4071

Table de fonctionnement		
$\mathbf{V_A}$	$\mathbf{V_B}$	$\mathbf{V_S}$
V_0	V_0	V_0
V_0	V_1	V_1
V_1	V_0	V_1
V_1	V_1	V_1

Table de vérité		
A	**B**	**S**
0	0	0
0	1	1
1	0	1
1	1	1

Remarque : Il existe des portes logiques OU et ET à 2, 3, 4, 8, et 13 entrées sous forme de circuit intégrés.

4.1.3 La porte NON (NOT)

C'est une porte à une seule entrée, elle matérialise l'operateur inverseur.

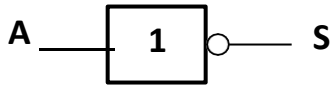
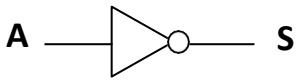
Symbole logique		**Equation**	**Circuit intégré**
Symbole International (CEI)	Symbole Européen (MIL)		
		$S=\overline{A}$	TTL : 7404 CMOS : 4069

Table de fonctionnement	
V_A	**V_S**
V_0	V_1
V_1	V_0

Table de vérité	
A	**S**
0	1
1	0

4.1.4 La porte OU-exclusif (XOR)

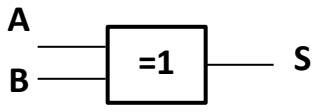
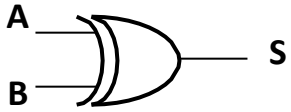
Symbole logique		**Equation**	**Circuit intégré**
Symbole International (CEI)	Symbole Européen (MIL)		
		$S=A\oplus B$ $=\overline{A}B*A\overline{B}$	TTL : 7486 CMOS : 4070

Table de fonctionnement		
V_A	**V_B**	**V_S**
V_0	V_0	V_0
V_0	V_1	V_1
V_1	V_0	V_1
V_1	V_1	V_0

Table de vérité		
A	**B**	**S**
0	0	0
0	1	1
1	0	1
1	1	0

La fonction OU-exclusif vaut **1** si une seule des entrées est à l'état **1** et l'autre est l'état **0**.

Généralisations de la fonction OU-EXCLUSIF : La sortie de la fonction OU-EXCLUSIF prend l'état logique **1** si un nombre impair des variables d'entrée est à l'état logique **1**.

Exemple : OU-exclusif a trois entrées

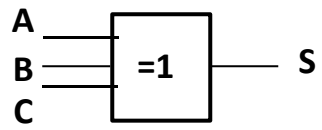
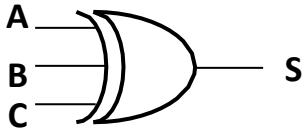
Symbole logique		**Equation**	**Circuit intégré**
Symbole International (CEI)	Symbole Européen (MIL)		
		S=A⊕B⊕C	TTL : 74386

Table de fonctionnement			
V_A	V_B	V_C	V_S
V_0	V_0	V_0	V_0
V_0	V_0	V_1	V_1
V_0	V_1	V_0	V_1
V_0	V_1	V_1	V_0
V_1	V_0	V_0	V_1
V_1	V_0	V_1	V_0
V_1	V_1	V_0	V_0
V_1	V_1	V_1	V_1

Table de vérité			
A	**B**	**C**	**S**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

4.2 Les portes universelles

Autre que les portes logiques de base (ou élémentaires), il existe des portes appelées portes logique universelles (complètes) telles que les portes NON-ET et NON-OU.

4.2.1 La porte NON-ET (NAND)

Elle est équivalente à une porte suivie d'un inverseur.

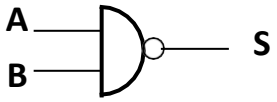
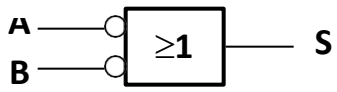
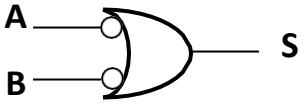
Symbole logique		**Equation**	**Circuit intégré**
Symbole International (CEI)	Symbole Européen (MIL)		
		$S=\overline{A\|B}$ $S=\overline{A.B}$ $S=\overline{A}+\overline{B}$	TTL : 7400 CMOS : 4011-4093
			

Table de fonctionnement		
V_A	**V_B**	**V_S**
V_0	V_0	V_1
V_0	V_1	V_1
V_1	V_0	V_1
V_1	V_1	V_0

Table de vérité		
A	**B**	**S**
0	0	1
0	1	1
1	0	1
1	1	0

Pour la porte NAND à trois entrées on trouve :

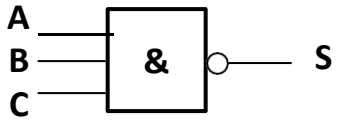
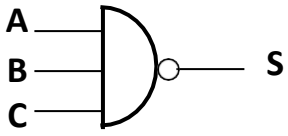
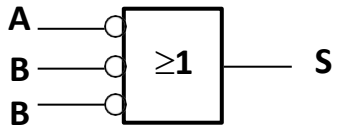
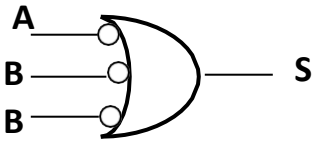
Symbole logique		**Equation**	**Circuit intégré**
Symbole International (CEI)	Symbole Européen (MIL)		
		$S=\overline{A\|B\|C}$ $S=\overline{A.B.C}$ $S=\overline{A}+\overline{B}+\overline{C}$	TTL : 7410 CMOS : 4023
			

Table de fonctionnement			
V_A	V_B	V_C	V_S
V_0	V_0	V_0	V_1
V_0	V_0	V_1	V_1
V_0	V_1	V_0	V_1
V_0	V_1	V_1	V_1
V_1	V_0	V_0	V_1
V_1	V_0	V_1	V_1
V_1	V_1	V_0	V_1
V_1	V_1	V_1	V_0

Table de vérité			
A	B	C	S
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

4.2.2 La porte NON-OU (NOR)

Elle est équivalente à une porte suivie d'un inverseur.

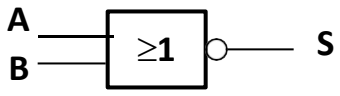
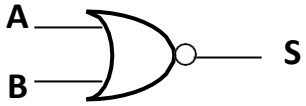
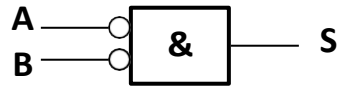
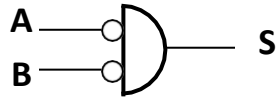
Symbole logique		Equation	Circuit intégré
Symbole International (CEI)	Symbole Européen (MIL)		
		$S=A\downarrow B$ $S=\overline{A+B}$ $S=\overline{A}.\overline{B}$	TTL : 7402 CMOS : 4001
			

Table de fonctionnement		
V_A	V_B	V_S
V_0	V_0	V_1
V_0	V_1	V_0
V_1	V_0	V_0
V_1	V_1	V_0

Table de vérité		
A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

Pour la porte NOR à trois entrées on trouve :

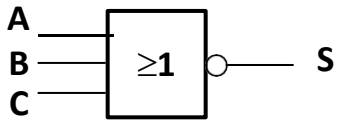
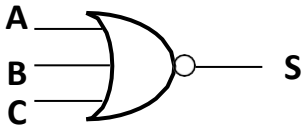
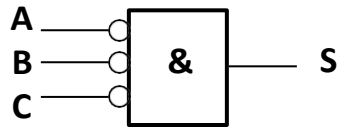
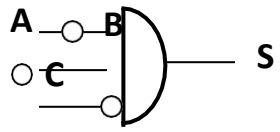
Symbole logique		**Equation**	**Circuit intégré**
Symbole International (CEI)	Symbole Européen (MIL)	$S=A\downarrow B\downarrow C$ $S=\overline{A+B+C}$ $S=\overline{A}.\overline{B}.\overline{C}$	TTL : 7427 CMOS : 4025
			
			

Table de fonctionnement			
V_A	**V_B**	**V_C**	**V_S**
V_0	V_0	V_0	V_1
V_0	V_0	V_1	V_0
V_0	V_1	V_0	V_0
V_0	V_1	V_1	V_0
V_1	V_0	V_0	V_0
V_1	V_0	V_1	V_0
V_1	V_1	V_0	V_0
V_1	V_1	V_1	V_0

Table de vérité			
A	**B**	**C**	**S**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

4.2.3 Exercice

1) Démontrer si les foncions universelles sont associatives :

   $(A|B)|C \stackrel{?}{=} A|(B|C) \stackrel{?}{=} A|B|C$

   $(A\downarrow B)\downarrow C \stackrel{?}{=} A\downarrow(B\downarrow C) \stackrel{?}{=} A\downarrow B\downarrow C$

2) Réaliser la fonction NAND à trois entrées à l'aide des opérateurs NAND à deux entrées.

Réponse :

1)

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

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

  $(A|B)|C\neq A|(B|C)$ alors la fonction NAND n'est pas associative

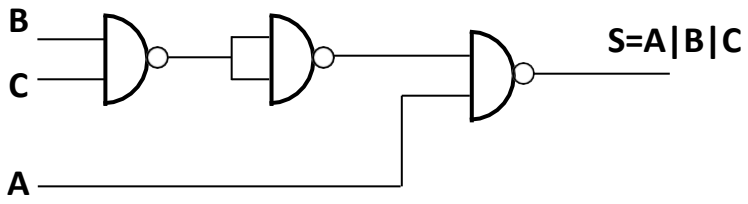
- $(A\downarrow B)\downarrow C=(\overline{A+B})\downarrow C=(\overline{A}.\overline{B})\downarrow C=(\overline{\overline{A}.\overline{B}})+C=(\overline{\overline{A}.\overline{B}}).\overline{C}=(A+B).\overline{C}$

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

  $(A\downarrow B)\downarrow C\neq A\downarrow(B\downarrow C)$ alors la fonction NOR n'est pas associative

2)

- $A|B|C=\overline{A.B.C}=\overline{A}+\overline{BC}= \overline{\overline{\overline{A}+\overline{BC}}} = \overline{A.B.C}=A|[(B|C)|(B|C)]$



Chapitre 3

REPRESENTATION ET SIMPLIFICATION DES FONCTIONS LOGIQUES COMBINATOIRES

1. OBJECTIFS

- Etudier la représentation algébrique d'une fonction logique,
- Comprendre la simplification algébrique d'une fonction logique,
- Faire la synthèse des applications combinatoires.

2. REPRESENTATION D'UNE FONCTION LOGIQUE

Une fonction logique est une combinaison de variables binaires reliées par les opérateurs ET, OU et NON. Elle peut être représentée par une écriture algébrique ou une table de vérité ou un tableau de KARNAUGH ou un logigramme.

2.1 Représentation algébrique

Une fonction logique peut être représentée sous deux formes :

- S. D. P : $\Sigma(\Pi)$ somme des produits,
- P. D. S. : Π (Σ) produit des sommes,

2.1.1 Forme somme des produits (Forme disjonctive)

Elle correspond à une somme de produits logiques : $F=\Sigma(\Pi(e_i))$, ou e_i représente une variable logique ou son complément.

Exemple : $F_{1(A, B, C)}=AB+\overline{B}C$.

Si chacun des produits contient toutes les variables d'entrée sous une forme directe ou complémentée, alors la forme est appelée : **« première forme canonique »** ou forme **« canonique disjonctive »**. Chacun des produits est appelé **minterme.**

Exemple : $F_{1(A, B, C)}=A\overline{B}\overline{C}+AB\overline{C}+\overline{A}\overline{B}C+A\overline{B}C$.

2.1.2 Forme Produit de sommes (Forme conjonctive)

Elle correspond à un produit de sommes logiques : $F=\Pi(\Sigma(e_i))$, ou e_i représente une variable logique ou son complément.

Exemple : $F_{2(A, B, C)}=(A+B).(A+\overline{B}+C)$.

Si chacune des sommes contient toutes les variables d'entrée sous une forme directe ou complémentée, alors la forme est appelée : **« deuxième forme canonique »** ou forme **« canonique conjonctive »**. Chacun des produits est appelé **maxterme.**

Exemple : $F_{2(A, B, C)}=(A+B+C).(A+B+\overline{C}).(A+\overline{B}+C)$

2.2 Table de vérité

Une fonction logique peut être représentée par une table de vérité qui donne les valeurs que peut prendre la fonction pour chaque combinaison de variables d'entrées.

2.2.1 Fonction complètement définie

C'est une fonction logique dont la valeur est connue pour toutes les combinaisons possibles des variables.

Exemple : La fonction « Majorité de 3 variables » : MAJ(A, B, C)

La fonction MAJ vaut 1 si la majorité (2 ou 3) des variables sont à l'état 1.

Table de vérité				
Combinaison	**A**	**B**	**C**	**S=MAJ(A, B, C)**
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

2.2.2 Fonction incomplètement définie

Il s'agit d'une fonction dont sa valeur est non spécifiée pour certaines combinaisons de variables. On l'indique le symbole X ou ∅ ; c'est-à-dire la fonction est indifférente pour certaines combinaisons de variables d'entrées correspondants à des situations qui soient :

- Ne peuvent jamais suivent dans le système,
- Ne changent pas le comportement du système.

Exemple : Soit un clavier qui comporte 3 boutons poussoirs P_1, P_2 et P_3 qui commandent une machine et qui possèdent un verrouillage mécanique tel que 2 boutons adjacents ne peuvent pas être enfoncés simultanément :

P_1⊙	P_2⊙	P_3⊙
Marche manuelle	Arrêt	Augmenter la vitesse

On suppose que P_i appuyé vaut **1** et relâché vaut **0**. D'où la table de vérité de la fonction « **clavier** » qui détecte au moins un poussoir déclenché :

Table de vérité				
Combinaison	**A**	**B**	**C**	**Clavier**
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	∅
4	1	0	0	1
5	1	0	1	1
6	1	1	0	∅
7	1	1	1	∅

2.2.3 Equivalence entre la table de vérité et les formes canonique

- Pour établir l'expression canonique disjonctive (la somme canonique) de la fonction : il suffit d'effectuer la somme logique (ou réunion) des mintermes associées aux états pour lesquels la fonction vaut « 1 ».
- Pour établir l'expression canonique conjonctive (le produit canonique) de la fonction : il suffit d'effectuer le produit logique (ou intersection) des maxtermes associées aux états pour lesquels la fonction vaut « 0 ».

Exemple : La fonction « Majorité de 3 variables » : MAJ(A, B, C)

Table de vérité						
Combinaison	**A**	**B**	**C**	**S=MAJ(A, B, C)**	**Minterme**	**Maxterme**
0	0	0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	$A+B+C$
1	0	0	1	0	$\overline{A}\,\overline{B}\,C$	$A+B+\overline{C}$
2	0	1	0	0	$\overline{A}\,B\,\overline{C}$	$A+\overline{B}+C$
3	0	1	1	1	$\overline{A}\,B\,C$	$A+\overline{B}+\overline{C}$
4	1	0	0	0	$A\,\overline{B}\,\overline{C}$	$\overline{A}+B+C$
5	1	0	1	1	$A\,\overline{B}\,C$	$\overline{A}+B+\overline{C}$
6	1	1	0	1	$A\,B\,\overline{C}$	$\overline{A}+\overline{B}+C$
7	1	1	1	1	$A\,B\,C$	$\overline{A}+\overline{B}+\overline{C}$

- On remarque que **MAJ(A,B,C)=1** pour les combinaisons 3, 5, 6, 7. On écrit la fonction ainsi spécifiée sous une forme dite numérique : **MAJ= R(3,5,6,7)**, Réunion des états 3, 5, 6, 7. La première forme canonique de la fonction **NAJ** s'en déduit directement :

$$MAJ_{(A,\,B,\,C)}=\overline{A}BC+A\overline{B}C+AB\overline{C}+ABC.$$

- On remarque que **MAJ(A,B,C)=0** pour les combinaisons 0, 1, 2, 4. On écrit la fonction ainsi spécifiée sous une forme dite numérique : **MAJ= I(0,1,2,4)**, Intersection des états 0, 1, 2, 4. La deuxième forme canonique de la fonction **NAJ** s'en déduit directement :

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

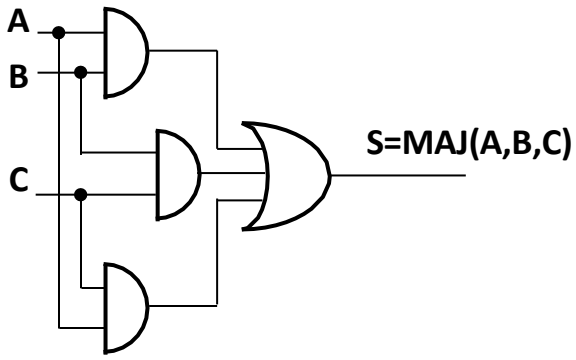
- **NB** : On s'intéresse généralement à la représentation d'une fonction sous la forme d'une somme ou somme canonique (forme disjonctive).

2.3 Logigramme

C'est une méthode graphique basée sur les symboles ou les portes.

Exemple : La fonction « Majorité de 3 variables » : MAJ(A,B,C)

$$MAJ_{(A,B,C)}=AB+BC+AC.$$



2.4 Le tableau de KARNAUGH (TK)

La méthode du tableau de KARNAUGH permet de visualiser une fonction et d'en tirer intuitivement une fonction simplifiée. L'élément de base de cette méthode est la table de KARNAUGH qui est représenté sous forme d'un tableau formé par des lignes et des colonnes.

2.4.1 Adjacence des cases

Deux mots binaires sont dits adjacents s'ils ne diffèrent que par la complémentaire d'une et d'une seule variable. Si deux mots adjacents sont sommés, ils peuvent être fusionnés et la variable qui en diffère sera éliminée. Les mots ABC et ABC sont adjacents puisqu'ils ne diffèrent que par la complémentarité de la variable C. Le théorème d'adjacence stipule donc qu'ABC et ABC= $\overline{A}$B.

2.4.2 Construction du tableau :

Le tableau de KARNAUGH a été construit de façon à faire ressortir l'adjacence logique visuelle.

- Chaque case représente une combinaison des variables (minterme),
- La table de vérité est transportée dans le tableau en mettant dans chaque case la valeur de la fonction correspondante.

La fonction représentée par un tableau de KARNAUGH s'écrit comme la somme des produits associés aux différentes cases contenant la valeur 1.

2.4.3 Règles à suivre pour un problème à n variables : (n>2)

Le tableau de KARNAUGH comporte 2^n cases ou combinaisons, L'ordre des variables n'est pas important mais il fait que respecter la règle suivante :

- Les monômes repérant les lignes et les colonnes sont attribués de telle manière que 2 monômes consécutifs ne diffèrent que de l'état d'une variable, il en résulte que 2 cases consécutives en ligne ou en colonne repèrent des combinaisons adjacentes, on utilise donc le code GRAY.

Exemple

- n=2

A \ B	$\overline{B}$(0)	B(1)
$\overline{A}$(0)	00	01
A(1)	10	11

- n=3

A \ BC	$\overline{BC}$(00)	$\overline{B}C$(01)	BC(11)	$B\overline{C}$(10)
$\overline{A}$(0)	000	001	011	010
A(1)	100	101	111	110

- n=4

AB \ CD	$\overline{C}\overline{D}$(00)	$\overline{C}D$(01)	CD(11)	$C\overline{D}$(10)
$\overline{A}\overline{B}$(00)	0000	0001	0011	0010
$\overline{A}B$(01)	0100	0101	0111	0110
AB(11)	1100	1101	1111	1110
$A\overline{B}$(10)	1000	1001	1011	1010

NB : Le Tableau de KARNAUGH à une structure enroulée sur les lignes et les colonnes. Il a une forme sphérique.

2.4.4 Exemple de remplissage du tableau de KARNAUGH à partir de la table de vérité :

Table de vérité					
Combinaison	**A**	**B**	**C**	**D**	$F_{(A,B,C,D)}$
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

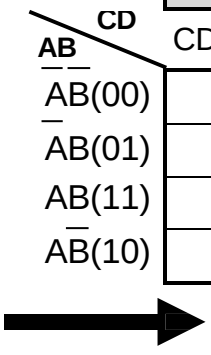


Tableau de KARNAUGH				
AB \ CD	CD(00)	$\overline{C}D$(01)	CD(11)	$C\overline{D}$(10)
$\overline{A}\overline{B}$(00)	0	1	0	0
$\overline{A}B$(01)	1	1	1	0
AB(11)	0	1	0	0
$A\overline{B}$(10)	0	0	1	0

3. SIMPLIFICATION DES FONCTIONS LOGIQUES

L'objectif de la simplification des fonctions logiques est des minimiser le nombre de termes afin d'obtenir une réalisation matérielle plus simple donc plus facile à construire et à dépanner et moins couteuse.

Deux méthodes de simplification sont utilisées :

- La simplification algébrique.
- La simplification graphique par tableau de KARNAUGH.

3.1 Simplification algébrique des expressions logiques

Pour obtenir une expression plus simple de la fonction par cette méthode, il faut utiliser :

- Les théorèmes et les propriétés de l'algèbre de Boole (voir chapitre 2).
- La multiplication par 1 ($X+\overline{X}$).
- L'addition d'un terme nul ($X\overline{X}$).

Exemple : Simplification de La fonction « Majorité» : MAJ(A,B,C)

$$MAJ_{(A,B,C)}=\overline{A}BC+A\overline{B}C+AB\overline{C}+ABC.$$

$$MAJ_{(A,B,C)}=\overline{A}BC+A\overline{B}C+AB\overline{C}+ABC+ABC+ABC.$$

$$MAJ_{(A,B,C)}=BC(\overline{A}+A)+AB(\overline{C}+C)+AC(\overline{B}+B).$$

$$MAJ_{(A,B,C)}=BC+AB+AC$$

NB : Les règles et propriétés de l'algèbre de Boole permettent de simplifier les fonctions mais reste une méthode relativement lourde. Elle ne permet jamais de savoir si l'on aboutit ou pas à une expression minimale de la fonction.

Nous pourrons alors utiliser la méthode du tableau de KARNAUGH

3.2 Simplification graphique des expressions logiques (par tableau de KARNAUGH)

Le tableau de KARNAUGH permet de visualiser une fonction et d'en tirer intuitivement une fonction simplifiée

3.2.1 Regroupement des cases adjacentes

La méthode consiste à réaliser des groupements des cases adjacentes. Ces groupements des case doivent être de taille maximale (nombre max de casse) et

égale à 2^k (c'est-à-dire 2, 4, 8, 16, …). On cesse d'effectuer les groupements lorsque tous les uns appartiennent au moins à l'un d'eux.

NB : Avant de tirer les équations du tableau de KARNAUGH il faut respecter les règles suivantes :

- Grouper tous les uns.
- Grouper le maximum des uns dans un seul groupement.
- Un groupement a une forme un rectangulaire.
- Le nombre des uns dans un groupement est une puissance de 2 est égal à 2^k.
- Un 1 peut figurer dans plus qu'un groupement.
- Un groupement doit respecter les axes de symétries du T. K.

Regroupement des 2 cases adjacentes

Simplification de la fonction Majorité de 3 variables (MAJ(A,B.C))

A \ BC	$\overline{B}\overline{C}$(00)	$\overline{B}C$(01)	BC(11)	$B\overline{C}$(10)
$\overline{A}$(0)	0	0	1	0
A(1)	0	1	1	1

$G_1=AB\overline{C}+A\overline{B}C=AC$

$G_2=\overline{A}BC+ABC=BC$

$G_3=AB\overline{C}+A\overline{B}C=AB$

$$MAJ(A,B,C)=G_1+G_2+G_3=AB+BC+AC$$

Règle : La réunion de deux cases adjacentes contenant 1 chacune élimine une seule variable celle qui change d'état en passant d'une case à l'autre.

Regroupement des 4 cases adjacentes

Fonction F_1

AB \ CD	$\overline{C}\overline{D}$(00)	$\overline{C}D$(01)	CD(11)	$C\overline{D}$(10)
$\overline{A}\overline{B}$(00)	0	0	0	1
$\overline{A}B$(01)	1	1	0	1
$A\overline{B}$(11)	1	1	0	1
AB(10)	0	0	0	1

$F_{1(A,B,C,D)}=B\overline{C}+C\overline{D}$

Fonction F_2

AB \ CD	$\overline{C}\overline{D}$(00)	$\overline{C}D$(01)	CD(11)	$C\overline{D}$(10)
$\overline{A}\overline{B}$(00)	1	0	0	1
$\overline{A}B$(01)	0	0	0	0
$A\overline{B}$(11)	1	0	0	1
AB(10)	1	0	0	1

$F_{2(A,B,C,D)}=A\overline{D}+\overline{B}\overline{D}$

Fonction F_3

AB \ CD	$\overline{C}\overline{D}$(00)	$\overline{C}D$(01)	CD(11)	$C\overline{D}$(10)
$\overline{A}\overline{B}$(00)	1	0	1	1
$\overline{A}B$(01)	1	0	0	0
AB(11)	1	1	1	1
$A\overline{B}$(10)	1	0	1	1

$$F_{3(A,B,C,D)}=\overline{C}\overline{D}+AB+\overline{B}C$$

Règle : 2 variables disparaissent quand on regroupe 4 cases adjacentes, on peut alors remplacer la somme des 4 cases (4 mintermes à 4 variables chacun) par un seul terme qui comporte que 2 variables uniquement.

- *<u>Regroupement des 8 cases adjacentes</u>*

Fonction F_4

AB \ CD	$\overline{C}\overline{D}$(00)	$\overline{C}D$(01)	CD(11)	$C\overline{D}$(10)
$\overline{A}\overline{B}$(00)	1	0	0	1
$\overline{A}B$(01)	1	0	0	1
AB(11)	1	0	0	1
$A\overline{B}$(10)	1	0	0	1

$$F_{4(A,B,C,D)}=\overline{D}$$

Règle : 2 variables disparaissent quand on regroupe 8 cases adjacentes, on peut alors remplacer la somme des 8 cases (8 mintermes à 4 variables chacun) par un seul terme qui comporte que 1 variable uniquement.

Remarque : On se limitera à des tableaux de 4 variables, pour résoudre par exemple des problèmes à 5 variables, on les décompose chacun a deux problèmes a 4 variables.

3.2.2 Traitement des problèmes à 5 variables

Pour résoudre ce problème on va le décomposer en 2 problèmes à 4 variables en appliquant le théorème d'expansion (SHANNON).

$$\text{on a : } F_{(A,B,C,D,E)}=\overline{E}\,F_{(A,B,C,D,0)}+E\,F_{(A,B,C,D,1)}$$

NB : Le théorème d'expansion de SHANNON reste applicable quelque soit le nombre de variables on a :

$$F_{(A,B,C,\ldots,Z)}=\overline{Z}\,F_{(A,B,C,\ldots,0)}+Z\,F_{(A,B,C,\ldots,1)}$$

Exemple : Simplifier la fonction $F_{(A,B,C,D,E)}=\Sigma(4, 5, 6, 7, 24, 25, 26, 27)$

$F_{(A,B,C,D,0)}$

AB \ CD	$\overline{C}\,\overline{D}$(00)	$\overline{C}D$(01)	CD(11)	$C\overline{D}$(10)
$\overline{A}\,\overline{B}$(00)	0	0	0	1
$\overline{A}B$(01)	0	0	0	1
AB(11)	0	0	0	1
$A\overline{B}$(10)	0	0	0	1

$F_{(A,B,C,D,1)}$

AB \ CD	$\overline{C}\,\overline{D}$(00)	$\overline{C}D$(01)	CD(11)	$C\overline{D}$(10)
$\overline{A}\,\overline{B}$(00)	0	1	0	0
$\overline{A}B$(01)	0	1	0	0
AB(11)	0	1	0	0
$A\overline{B}$(10)	0	1	0	0

$F_{(A,B,C,D,0)}=C\overline{D}$ $\qquad$ $F_{(A,B,C,D,1)}=\overline{C}D$

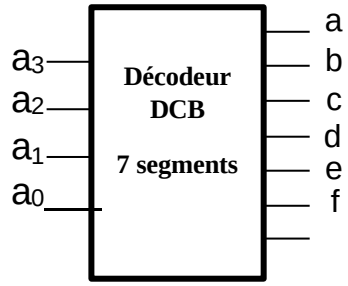
Ce qui en résulte : $F_{(A,B,C,D,E)}=\overline{E}C\overline{D}+E\overline{C}D$

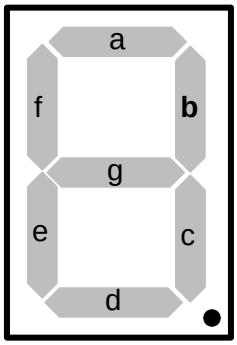
4. RESUME : SYNTHESE D'UNE FONCTION LOGIQUE

- **Etape 1** : Lecture et analyse de l'énoncée de la fonction.
- **Etape 2** : écriture de la fonction sous forme canonique d'une table de vérité.
- **Etape 3** : Simplification de l'expression de la fonction par la méthode algébrique ou par la méthode du T. K.
- **Etape 3** : Réalisation du logigramme :
  - Avec un seul types des opérateurs en utilisant les fonctions logiques universelles.
  - Avec un minimum des opérateurs en utilisant les fonctions logiques de base

1.1.1 Le décodeur DCB 7 segments

Le décodeur 7 segments accepte en entrée les 4 bits DCB (a_0, a_1, a_2, a_3) et rend actives les sorties qui vont permettre de faire passer un courant dans les segments d'un afficheur numérique pour former les chiffres décimaux (de 0 à 9).





R**emarque :** Il y’a 6 combinaisons intitulés **10, 11, 12, 13, 14, 15** que l’on notera ∅. Les autres chiffres sont affichés comme suit :

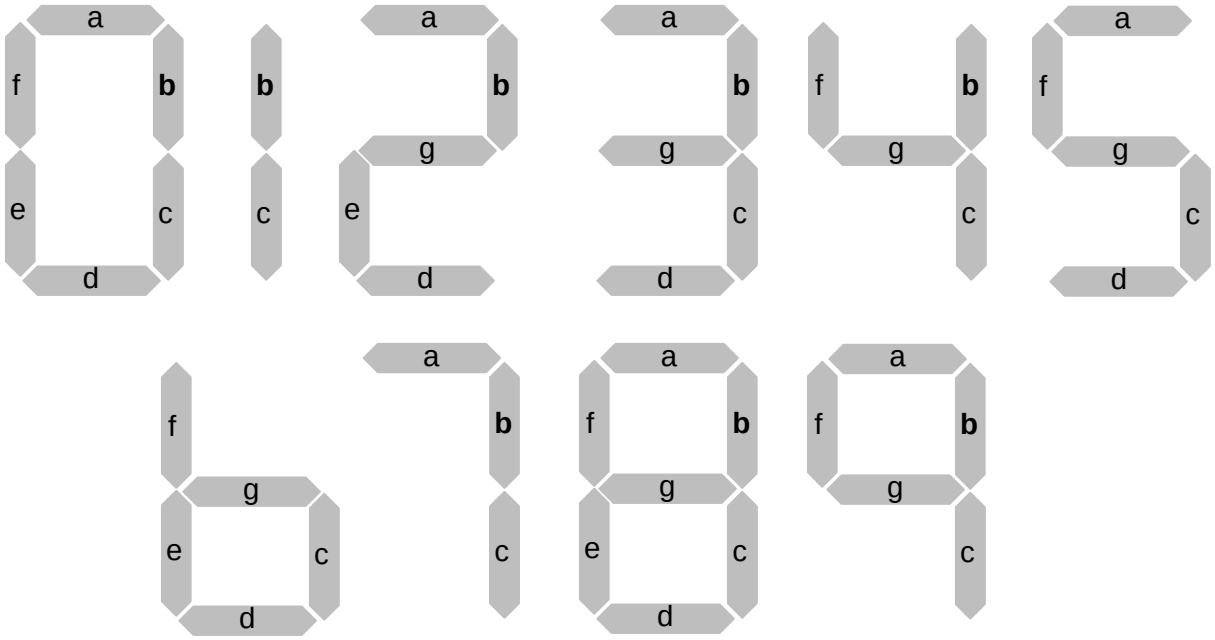


Table de vérité											**Affichage**
Entrées				**Sorties**							
a_3	**a_2**	**a_1**	**a_0**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	0	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	0	0	1	1	9

Exemple : Décodeur DCB

Segment a

a_1a_0 \ a_3a_2	$\bar{a}_3\bar{a}_2$00	$\bar{a}_3a_2$01	$a_3a_2$11	$a_3\bar{a}_2$10
$\bar{a}_1\bar{a}_0$00	1	0	∅	1
$\bar{a}_1a_0$01	0	1	∅	1
a_1a_0 11	1	1	∅	∅
$a_1\bar{a}_0$10	1	0	∅	∅

$a=\bar{a}_2a_1+a_2a_0+\bar{a}_2\bar{a}_0+a_3$

Segment b

a_1a_0 \ a_3a_2	$\bar{a}_3\bar{a}_2$00	$\bar{a}_3a_2$01	$a_3a_2$11	$a_3\bar{a}_2$10
$\bar{a}_1\bar{a}_0$00	1	1	∅	1
$\bar{a}_1a_0$01	1	0	∅	1
a_1a_0 11	1	1	∅	∅
$a_1\bar{a}_0$10	1	0	∅	∅

$b=\bar{a}_2+\bar{a}_1\bar{a}_0+a_1\ a_0=\bar{a}_2+a_1\odot a_0$

Segment c

a_1a_0 \ a_3a_2	$\bar{a}_3\bar{a}_2$00	$\bar{a}_3a_2$01	$a_3a_2$11	$a_3\bar{a}_2$10
$a_1a_0$00	1	1	∅	1
$\bar{a}_1a_0$01	1	1	∅	1
a_1a_0 11	1	1	∅	∅
$a_1\bar{a}_0$10	0	1	∅	∅

$c=a_2+\bar{a}_1+a_0$

Segment d

a_1a_0 \ a_3a_2	$a_3a_2$00	$a_3a_2$01	$a_3a_2$11	$a_3\bar{a}_2$10
$a_1a_0$00	1	0	∅	1
$\bar{a}_1a_0$01	0	1	∅	0
a_1a_0 11	1	0	∅	∅
$a_1\bar{a}_0$10	1	1	∅	∅

$d=a_2a_0+a_3\ \bar{a}_0+a_2a_1+a_1\bar{a}_0+a_2\bar{a}_1a_0$

Segment e

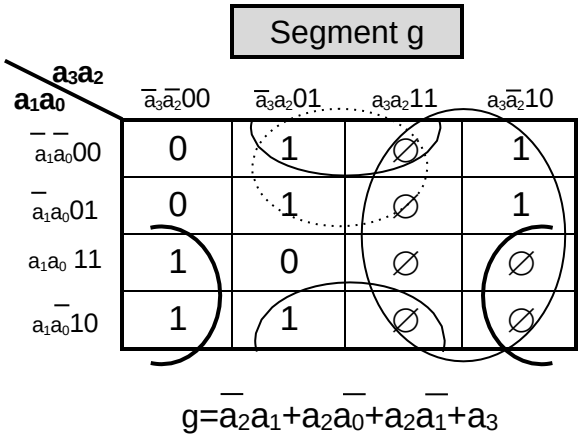
a_1a_0 \ a_3a_2	$\bar{a}_3\bar{a}_2$00	$\bar{a}_3a_2$01	$a_3a_2$11	$a_3\bar{a}_2$10
$\bar{a}_1\bar{a}_0$00	1	0	∅	1
$\bar{a}_1a_0$01	0	0	∅	0
a_1a_0 11	0	0	∅	∅
$a_1\bar{a}_0$10	1	1	∅	∅

$e=a_1\bar{a}_0+a_2\bar{a}_0$

Segment f

a_1a_0 \ a_3a_2	$a_3a_2$00	$\bar{a}_3a_2$01	$a_3a_2$11	$a_3\bar{a}_2$10
$\bar{a}_1\bar{a}_0$00	1	1	∅	1
$\bar{a}_1a_0$01	0	1	∅	1
a_1a_0 11	0	0	∅	∅
$a_1\bar{a}_0$10	0	1	∅	∅

$f=a_1a_0+a_2a_1+a_2a_0+a_3$

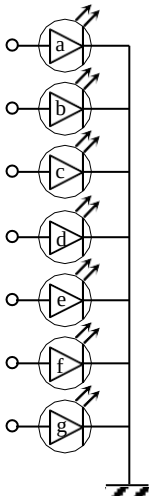


$g=\bar{a}_2a_1+a_2\bar{a}_0+a_2\bar{a}_1+a_3$

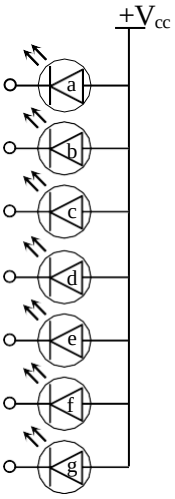
Remarque : L'afficheur est composée de 7 LEDS (segments), a, d, c, d, e, f, g qui nécessitent en fonction du type d'afficheur (anode commune ou cathode commune) une polarisation spécifique :

- Pour un afficheur à anodes communes : Les anodes sont reliées ensembles au niveau haut et les sorties du décodeur sont actives au niveau bas **(CI : 74LS47**) et sont reliées aux cathodes de l'afficheur.

- Pour un afficheur à cathodes communes : Les cathodes sont reliées ensembles à la masse et les sorties du décodeur sont active au niveau haut **(CI : 74LS48**) et sont reliées aux anodes de l'afficheur.



Afficheur à cathodes communes



Afficheur à anodes comm