UNIVERSITE MOULAY ISMAIL **FACULTE DES SCIENCES** **DEPARTEMENT DE PHYSIQUE** **MEKNES**	**A.U : 2019-2020** **SEMESTRE II** **Master degree** **Nanophysics and Advanced Optics**

Cours : **codage**

Pr A. ROUKHE

Sommaire

Chapitre 1 Eléments d’ Algèbre de Boole

Chapitre 2 Les Fonctions Logiques

Chapitre 3 Minimisation des Fonctions Logiques

Chapitre 4 Logique Séquentielle.

Chapitre 1 ELEMENTS D'ALGEBRE DE BOOLE

I. INTRODUCTION

L'algèbre de Boole se trouve à la base de tout l'électronique digitale et des technologies numériques modernes. Des fonctionnalités complexes peuvent en effet être créées en combinant divers opérateurs booléens (portes logiques) implémentés dans le matériel électronique. Le succès de l'ère digitale provient du fait qu'elle a pu intégrer un nombre de plus en plus croissant d'éléments dans très peu d'espace (augmentation de la densité des circuits intégrés ou puce).

Au début de l'ère digitale, qu'on dénommé SSI (*Small-Scale Integration*), les puces ne contenaient que quelques portes logiques. Aujourd'hui, à l'époque du VLSI (*Very Large-Scale Integration*), un microprocesseur peut contenir des millions de portes logiques !

Historique :

George Boole (philosophe et mathématicien anglais, 1815–1864) a démontré que si l'on peut trouver un espace dans lequel certains axiomes se vérifient, alors on se trouve dans un cas singulier, où un certain nombre de théorèmes peuvent s'appliquer.

- 1938: Utilisation de l'algèbre de BOOLE pour l'étude des circuits électriques
- 1854: Georges Boole propose en un système algébrique qui sera plus tard connu sous le nom d'algèbre booléenne. Dans ce système, les propositions logiques sont indiquées par des symboles et peuvent être exécutées par des opérateurs mathématiques abstraits qui correspondent aux lois de la logique.
- XXème siècle : Développement de l'électronique digitale à 2 niveaux, ce qui donne un grand intérêt à cette algèbre.

L'algèbre de BOOLE couvre toutes les algèbres à deux éléments, elle permet de représenter toutes les fonctions de : $\{0,1\}^n \rightarrow \{0,1\}$

Algèbre de Boole

Définitions 1 :

Un système logique est dit combinatoire, lorsque la ou les sorties ne dépendent que de la combinaison de ses entrées.

États logiques : Variable logique représentée par un identificateur (lettre ou nom) qui peut prendre les seules valeurs 0 ou 1 .

Fonction logique: de n variables binaires : groupe de variables reliées par des opérateurs logiques.

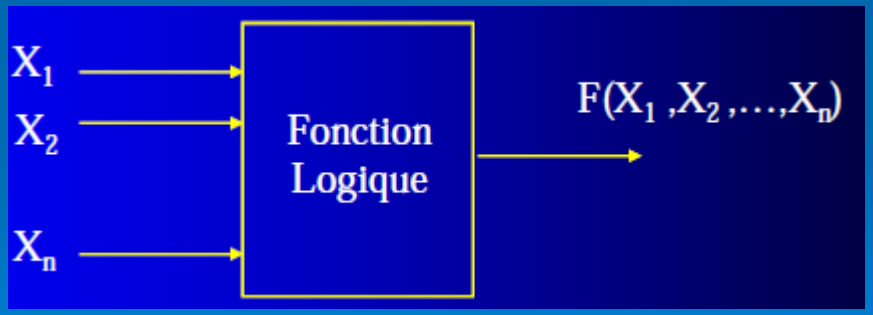
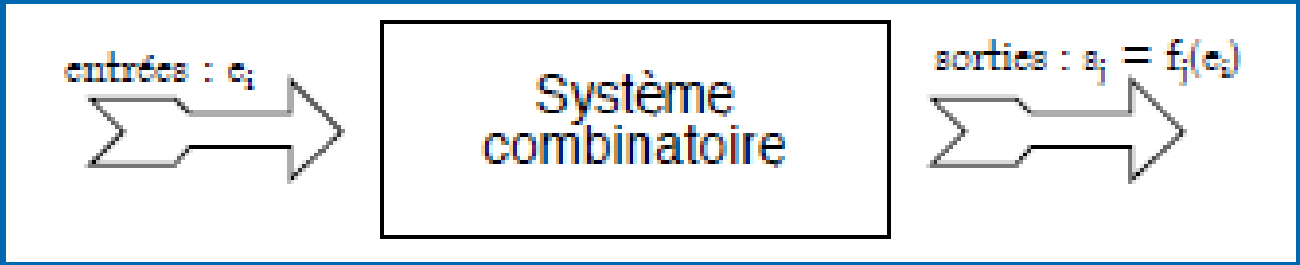


Table de vérité: tableau regroupant toutes les combinaisons de valeurs (0 ou 1) que peuvent prendre les variables binaires X1,X2..Xn

Définitions 2:

- Dans un système logique combinatoire (les entrées et les sorties ne peuvent prendre que 0 ou 1 comme valeur), les sorties ne sont fonctions que des entrées.



- Soit B un ensemble muni de deux opérations internes + et . on dit que (B,+ , .) est une algèbre de Boole si

(+) et (.) sont associatives et commutatives

(+) admet un neutre 0 et (.) admet un neutre 1

(.) est distributive sur (+) et (+) est distributive sur .

tout élément a de B admet un complémentaire dans B tel que

$a + \overline{a} = 1$ **et** $a \times \overline{a} = 0$

On se limite : Base de l'algèbre de Boole
Propriétés indispensables aux systèmes logiques

A1 : Quantité Booléenne simple.

C'est une quantité siceptiple de prendre seulement deux valeurs

aux quelles on peut correspondre (0,1) ; (V,F) ou (-1,1)

A2 : Quantité Booléenne Complexe (général)

Xm : quantité Complexe ou générale

$$X_m \in \{0,1\}^n \qquad X_m = \begin{Vmatrix} x_1 \\ \vdots \\ x_n \end{Vmatrix} \qquad X_i \in \{0,1\}$$

Systèmes d’opérateurs, complétude et minimalité :

Définition :

Un système d’opérateur sur {0,1} est dit complet si il permet de représenter n’importe quelle fonction booléenne.

Un système d’opérateur complet est dit minimal si l’on ne peut en extraire un sous-système qui soit lui-même complet.

On démontre qu’un système d’opérateur est complet s’il est possible de reconstruire les opérateurs OU, ET et $\overline{}$ avec celui-ci.

1 **Codages**

Définitions On appelle alphabet un ensemble fini de lettres (ou chiffres) par exemple:

alphabet binaire II = (0,1) (un chiffre binaire est appelé **BIT**)

alphabet octal VIII = (0,1,2,3,4,5,6,7)

alphabet décimal X = (0,1,2,3,4,5,6,7,8,9)

alphabet hexadécimal H = (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

On appelle **mot** construit sur un alphabet A tout élément de A^n.

Par exemple sur l'alphabet binaire on peut construire 4 mots de longueur 2:

00 , 01 , 10 , 11

Un mot binaire de 8 bits s'appelle un **OCTET** (ou BYTE)

Numérotation binaire

L'utilisation d'un signal binaire permet entre autre de coder des nombres en base 2. Ainsi le nombre 1101 en base 2 vaut 1x23+1x22+0x21+1x20=13 en base 10. Le bit le plus à droite du nombre binaire est le bit de poids faible (Less Significant Bit, ou LSB) ; il représente "uniquement" l'information de parité du nombre. Le bit le plus à gauche est le bit de poids fort (Most Significant Bit, ou MSB) ; il représente plus de la moitié de la valeur du nombre.

Le codage en base 2 consomme plus de chiffres qu'en base dix.

On appelle **codage** de A dans B une correspondance biunivoque entre des mots de A^n et des mots de B^k .

Si tous les codes ont le même nombre de lettres le **codage** est dit **normal**

Un **codage** est dit **pondéré** si on peut affecter a chaque chiffre de rang i un poids ai tel que la somme pondérée des chiffres d'un mot est égale au numéro du mot. La table ci-dessous donne un exemple de codage pondéré avec des poids 5 , 2 , 2 , 1:

Les **codages naturels** (ou systèmes de numération en base n)

sont des codages dont les poids sont n^0, n^1, n^2, ...(n étant le nombre de chiffres de l'alphabet)

Les **codages naturels** les plus utilisés sont: le binaire (n=2), l'octal (n=8), le décimal (n=10), l'hexadécimal (n=16)

ND	5	2	2	1	<== POIDS
0	0	0	0	0	0+0+0+0=0
1	0	0	0	1	0+0+0+1=1
2	0	0	1	0	0+0+2+0=2
3	0	0	1	1	0+0+2+1=3
4	0	1	1	0	0+2+2+0=4
5	1	0	0	0	5+0+0+0=5
6	1	0	0	1	5+0+0+1=6
7	1	0	1	0	5+0+2+0=7
8	1	0	1	1	5+0+2+1=8
9	1	1	1	0	5+2+2+0=9

2. Codages naturels: binaire naturel (numération en base 2)

C'est le système le plus utilisé en électronique numérique. Il comprend deux symboles {0,1} appelés bits.

Le binaire naturel est un codage binaire pondéré de poids 2^0, 2^1, 2^2,.....

Un nombre binaire s'écrit de la façon suivante

$$(N)_2 = (a_n \ldots\ldots\ldots\ldots a_0)_2 \text{ où } a_i \in \{0,1\}$$

a_0 est le bit le moins significatif (**LSB** : Low Significant Bit).

a_n est le bit le plus significatif (**MSB** : Most Significant Bit)

Selon la forme polynomiale, nous pouvons écrire :

$$N = \sum_{i=0}^{n} a_i 2i \text{ , où } a_i \in \{0,1\}$$

3. Système octal (b = 8)

Ce système dispose de huit symboles : {0,1,2,3,4,5,6,7}.

De la même manière, un nombre octal s'écrit:

$N = (a_n \ldots\ldots\ldots\ldots a_0)_8 \quad$ où $a_i \in \{0,1,2,3,4,5,6,7\}$

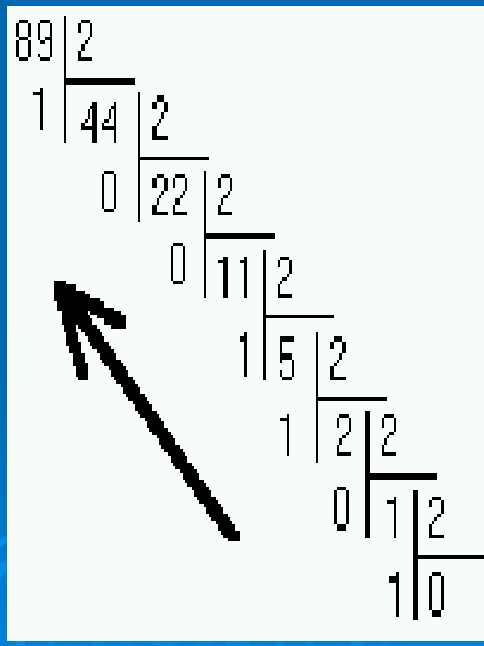
Passage binaire => décimal:

On peut convertir un nombre de la base 10 à la base 2 par la méthode des divisions successives par 2 ; ainsi $37_{10}=100101_2$.

1011001 binaire représente :

$1x2^6 + 0x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0 =$

$1x64 + 0x32 + 1x16 + 1x8 + 0x4 + 0x2 + 1x1 = 89$décimal



à l'inverse, pour transformer 89d en binaire, on peut utiliser la méthode des divisions successives par 2 : on divise successivement par 2 jusqu'à un résultat de 0, les restes successifs (de bas en haut) forment le nombre binaire.

hexadécimal (numération en base 16) On utilise les chiffres 0 à 9 puis les lettres A à F

conversion hexadécimal => décimal:

3 A 5 $_{\text{hexa}}$ vaut **3**x16^2 + **10**x16^1 +**5**x16^0 =3x256+160+5 = **933**$_{\text{déci}}$

Inversement on passe du décimal à l'hexa par divisions successives par 16.

conversion hexadécimal => binaire:

il suffit de remplacer chaque chiffre par sa valeur binaire sur quatre bits :

3 A 5 $_{\text{h}}$ = **0011 1010 0101** $_{\text{b}}$

En pratique aucune machine ne compte en hexadécimal. Elles travaillent toutes en binaire, et ne se servent de l'hexa que pour dialoguer avec nous (nous nous trompons trop souvent dans de longues listes de 0 et 1).

4. Codages BCD

le Décimal Codé en Binaire (DCB ou BCD en anglais)

Le code DCB signifie Décimal Codé Binaire. Chaque chiffre du nombre décimal est codé individuellement en son équivalent binaire sur quatre bits (quartet), ce qui n'est pas le cas pour le code binaire naturel où on convertit le nombre décimal dans son intégralité.

Exemple :

Nombre décimal	*code DCB*	*Code binaire*
127	$(000100100111)_{DCB}$	$(1111111)_2$
255	$(001001010101)_{DCB}$	$(11111111)_2$
64	$(01100100)_{DCB}$	$(1000000)_2$

Dans ce codage chaque chiffre décimal est codé en binaire sur 4 bits

Ce codage permet le passage facile du décimal au binaire et est surtout utilisé pour afficher des valeurs numériques. Il pose deux problèmes principaux :

un certain nombre de combinaisons ne sont pas utilisées (celles qui correspondent à A à F en hexa). Sur 8 bits on représente les nombres de 0 à 99 au lieu de 0 à 255 en binaire naturel. Sur 16 bits, on se limite à 9999 au lieu de 65535...

les calculs sont beaucoup plus compliqués qu'en binaire naturel. Parfois pour contourner cette difficulté on utilise le **BCD excess 3** obtenu en éliminant les 3 premières combinaisons 0000 , 0001 , 0010.

Remarques :

- Le code DCB est un code non pondéré. Il n'obéit pas à la Forme Polynomiale.
- Dans le code DCB, il faut plus de bits pour exprimer le même nombre, qu'en code binaire.
- Le code DCB n'utilise que dix quartets parmi 16. Si l'un des quartets interdits (1010,1011,1100,1101,1110,1111) se manifeste dans un calculateur utilisant le code DCB, c'est alors un signe d'erreur.

5. Codages continus

Deux mots binaires sont **adjacents** s'ils ne diffèrent que par un seul bit (par exemple 11010 et 10010 sont adjacents).

Si tous les mots successifs d'un codage sont adjacents le code est dit **continu**.

Si le premier et le dernier mot du codage sont adjacents le code est dit **cyclique**.

Le codage continu cyclique le plus utilisé est le **binaire réfléchi** (ou **code GRAY**)

Ce codage est utilisé dans les codeurs de position sur certaines machines à commande numérique; il évite les erreurs lorsque l'on passe d'une position à la suivante.

binaire réfléchi

0	**0 0 0 0**		**8**	**1 1 0 0**
1	**0 0 0 1**		**9**	**1 1 0 1**
2	**0 0 1 1**		**10**	**1 1 1 1**
3	**0 0 1 0**		**11**	**1 1 1 0**
4	**0 1 1 0**		**12**	**1 0 1 0**
5	**0 1 1 1**		**13**	**1 0 1 1**
6	**0 1 0 1**		**14**	**1 0 0 1**
7	**0 1 0 0**		**15**	**1 0 0 0**

Remarque:

Dans le code binaire réfléchi, deux représentations codées successives ne différent que d'un seul bit

Le code utilisé dans ce cas est le code GRAY.

Avantage: Un seul bit change en passant d'un nombre au prochain.

Exemple: 7 →8 : 0100 → 1100

→Mieux que le BCD : 0111 → 1000 ou 4 bits changent à la fois

→Donc entre les deux passages 7 et 8, d'autres valeurs erronées sont captes.1.6.2CODE GRAY

Symbole Décimale	Code Gray
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
16	1000

la correspondance entre le décimal, le binaire et le binaire réfléchi sur 2 bits :

Décimal	Binaire naturel	Binaire réfléchi
0	$(00)_2$	$(00)_2$
1	$(01)_2$	$(01)_2$
2	$(10)_2$	$(11)_2$
3	$(11)_2$	$(10)_2$

correspondance entre le décimal, le code binaire et le code binaire réfléchi sur 3 bits :

Décimal	binaire	binaire réfléchi
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

7. Code ASCII 7 bits

Le code ASCII signifie **American Standard Code for Information Interchange,** il a été mis au point par l'organisation de normalisation américaine appelée ANSI. Ce code constitue une norme universelle pour l'échange d'information entre le micro-ordinateur et ses périphériques (imprimante, clavier, écran, etc.).

Le code ASCII est un code alphanumérique. Il permet de représenter les chiffres, les lettres majuscules et les lettres minuscules ainsi que des caractères spéciaux (? ! + - : / # @ & ….).

Le code ASCII 7 bits fait correspondre à chaque caractère alphanumérique un code binaire sur 7 bits permettant ainsi de représenter au maximum 128 caractères différents.

Le tableau ci-dessous nous donne une partie du code ASCII où les 7 bits sont désignés par b_6, b_5, b_4, b_3, b_2, b_1, b_0

Remarque: De nombreux autres codages binaires sont utilisés pour des applications particulières: codes détecteurs et correcteurs d'erreurs (bit de parité, code de Hamming) codes ASCII etc...