

Introduction à la logique Booléenne

Table des matières

TABLE DES MATIÈRES.....	2
REPRÉSENTATION DES NOMBRES ENTIERS.....	3
LA BASE 2 (BINAIRE).....	3
LA BASE 16 (HEXADÉCIMAL).....	3
LA BASE 8 (OCTAL).....	4
LE DÉCIMAL CODÉ EN BINAIRE (DCB OU BCD EN ANGLAIS).....	4
ALGÈBRE DE BOOLE - EQUATIONS LOGIQUES.....	4
DÉFINITIONS:.....	4
OPÉRATEURS LOGIQUES:.....	5
AXIOMES.....	5
THÉORÈMES.....	5
TABLES DE VÉRITÉ.....	6
SYMBOLES LOGIQUES.....	7
LOGIQUE COMBINATOIRE.....	8
SYNTHÈSE.....	8
<i>Synthèse à partir de la table de vérité.....</i>	<i>8</i>
EXEMPLE : FONCTION MAJORITÉ À TROIS VARIABLES.....	8
TABLES DE KARNAUGH.....	9
<i>Méthode.....</i>	<i>9</i>
<i>Exemple : fonction majorité à trois variables.....</i>	<i>10</i>
FONCTIONS INCOMPLÈTEMENT DÉFINIES.....	10
<i>Table de Karnaugh.....</i>	<i>10</i>
SYSTÈME À FONCTIONS MULTIPLES.....	10
<i>Méthode.....</i>	<i>11</i>
LOGIQUE SÉQUENTIELLE.....	11
SYSTÈMES SYNCHRONES ET ASYNCHRONES.....	11

Représentation des nombres entiers

la base 2 (Binaire)

On peut représenter des nombres par une combinaison de zéros et de uns. Chaque chiffre binaire est appelé BIT. 8 bits forment un octet (BYTE). Mais plusieurs codifications sont envisageables. La plus utilisée est le binaire (indice b) -> décimal (indice d) :

1011001_b représente :

$$\begin{aligned} & 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ = & 1 \times 64 + 0 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 0 \times 2 + 1 \times 1 \\ = & 89_d \end{aligned}$$

à l'inverse, pour transformer 89_d 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.

Ex : 89 _d :	89 :2=44	reste	1	LSB (Less Significant Bit)
	44 :2=22	reste	0	
	22 :2=11	reste	0	
	11 :2=5	reste	1	
	5 :2=2	reste	1	
	2 :2=1	reste	0	
	1 :2=0	reste	1	MSB (Most Significant Bit)

$$89_d = 1011001_b$$

la base 16 (hexadécimal)

On utilise les chiffres 0 à 9 puis les lettres A à F.

$$3A5_h \text{ vaut } 3 \times 16^2 + 10 \times 16^1 + 5 \times 16^0 = 3 \times 256 + 160 + 5 = 933_d .$$

Transformer de l'hexadécimal en binaire est très simple : il suffit de remplacer chaque chiffre par sa valeur binaire sur quatre bits :

3A5_h = 0011 1010 0101_b (on peut vérifier que ça vaut 933_d). En effet, 001110100101_b

$$\begin{aligned} = & 0.2^{11} + 0.2^{10} + 1.2^9 + 1.2^8 + 1.2^7 + 0.2^6 + 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 \\ = & 0 + 0 + 512 + 256 + 128 + 0 + 32 + 0 + 0 + 4 + 0 + 1 \\ = & 933_d \end{aligned}$$

Ou :

$$\begin{aligned} = & 3_d \times 16^2 + 10_d \times 16 + 5_d \\ = & 3 \times 256 + 10 \times 16 + 5 \\ = & 933_d \end{aligned}$$

Contrairement à ce que beaucoup de gens croient, aucune machine ne compte en hexadécimal. Elles travaillent toutes en binaire, et ne se servent de l'hexadécimal que pour dialoguer avec nous (nous nous trompons trop souvent dans de longues listes de 0 et 1).

la base 8 (Octal)

La Base 8 est la moins utilisée, mais voici quand même la méthode pour passer de l'octal au décimal.

$$472_o \text{ vaut } 4 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 = 4 \times 64 + 56 + 2 = 314_d.$$

Transformer de l'octal en binaire est également très simple : il suffit de remplacer chaque chiffre par sa valeur binaire sur trois bits :

$472_o = 100\ 111\ 010_b$ (on peut vérifier que ça vaut 314_d). En effet, 100111010_b

$$\begin{aligned} &= 1 \cdot 2^8 + 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 256 + 0 + 0 + 32 + 16 + 8 + 0 + 2 + 0 \\ &= 314_d \end{aligned}$$

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

Si vous achetez un voltmètre numérique, la valeur mesurée est transmise à l'afficheur en numérique. Mais elle est auparavant transformée en décimal, chaque chiffre décimal est transmis à un afficheur en binaire naturel (sur 4 bits). C'est le BCD : la juxtaposition des valeurs binaires (sur quatre bits) des chiffres décimaux. Donc 583_d se notera $0101\ 1000\ 0011_{bcd}$.

Algèbre de Boole - Equations Logiques

Un processeur est composé de transistors permettant de réaliser des fonctions sur des signaux numériques. Ces transistors, assemblés entre eux forment des composants permettant de réaliser des fonctions très simples. A partir de ces composants il est possible de créer des circuits réalisant des opérations très complexes. L'algèbre de Boole (du nom du mathématicien anglais *Georges Boole 1915 - 1864*) est un moyen d'arriver à créer de tel circuit.

L'algèbre de Boole est une algèbre se proposant de traduire des signaux en expressions mathématiques. Pour cela, on définit chaque signal élémentaire par des variables logiques et leur traitement par des fonctions logiques. Des méthodes (table de vérité) permettent de définir les opérations que l'on désire réaliser, et à transcrire le résultat en une expression algébrique. Grâce à des règles, ces expressions peuvent être simplifiées. Cela va permettre de représenter grâce à des symboles un circuit logique, c'est-à-dire un circuit qui schématise l'agencement des composants de base (au niveau logique) sans se préoccuper de la réalisation au moyen de transistors (niveau physique).

Définitions:

- Etat: Les états logiques sont représentés par 0 et 1.
- Variable: C'est une grandeur représentée par un symbole, qui peut prendre un état (0 ou 1).
- Fonction: Elle représente un groupe de variables reliées par des opérateurs logiques.

Exemple:

0 = lampe éteinte
1 = lampe allumée

Opérateurs Logiques:

Voici les 3 types d'opérateurs élémentaires:

1. **ET** (AND)
2. **OU** (OR)
3. **NON** (NOT)

Les symboles:

1. ET => . (un point)
2. OU => + (un plus)
3. NON => une barre au dessus

axiomes

Pour qu'une algèbre puisse être dite de Boole, elle doit vérifier :

Commutativité	$a+b=b+a$	$a.b=b.a$
Associativité	$(a+b)+c=a+(b+c)$	$(ab)c=a(bc)$
Distributivité	$a(b+c)=ab+ac$	$a+(bc)=(a+b)(a+c)$
Éléments neutres	$a+0=a$	$a.1=a$
Complémentarité	$\overline{\overline{a+a}}=1$	$\overline{\overline{a}}.a=0$

théorèmes

Une algèbre de Boole vérifie les théorèmes suivants :

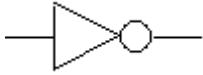
Idempotence	$a+a=a$	$aa=a$
Absorption	$a+ab=a$	$a(a+b)=a$
De Morgan	$\overline{\overline{a+b}}=\overline{\overline{a}}.\overline{\overline{b}}$	$\overline{\overline{a}}.\overline{\overline{b}}=a+b$
Élément neutre	$a+1=1$	$a.0=0$

Tables de vérité

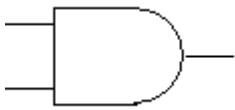
Une table de vérité est un tableau permettant de décrire toutes les possibilités de sorties en fonction de entrées. On place donc les variables d'entrées dans les colonnes de gauche en les faisant varier de telle façon à couvrir l'ensemble des possibilités. La colonne (ou les colonnes si la fonction a plusieurs sorties) de droite décrit la sortie. Voici par exemple les tables de vérités des portes logiques:

Fonctions Logique	Entrées		Sortie
	B	A	S
AND $S = A \cdot B$	0	0	0
	0	1	0
	1	0	0
	1	1	1
OR $S = A + B$	0	0	0
	0	1	1
	1	0	1
	1	1	1
NAND $S = \overline{A \cdot B}$	0	0	1
	0	1	1
	1	0	1
	1	1	0
NOR $S = \overline{A + B}$	0	0	1
	0	1	0
	1	0	0
	1	1	0
XOR $S = A \oplus B$	0	0	0
	0	1	1
	1	0	1
	1	1	0
XNOR $S = \overline{A \oplus B}$	0	0	1
	0	1	0
	1	0	0
	1	1	1
INV $S = \overline{IN}$	0		1
	1		0

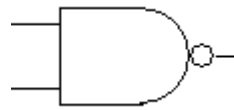
Symboles logiques



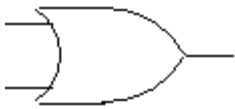
INV



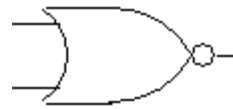
AND



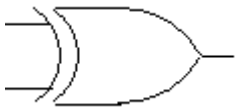
NAND



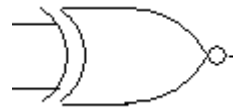
OR



NOR



XOR



XNOR

Logique Combinatoire

Un système logique est dit *combinatoire* si l'état de sa sortie ne dépend que de l'état de ses entrées. Le système combinatoire ne doit donc pas présenter de réactions de la sortie sur l'entrée, de sorte à ce que l'état de la sortie ne dépende pas de l'histoire du système.

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

Synthèse

Lors de la synthèse de circuits logiques, l'un des problèmes fréquemment rencontrés consiste à réaliser une fonction logique à partir d'une table de vérité ou d'un cahier des charges. Ce résumé présente brièvement quelques méthodes permettant de résoudre ce problème.

Synthèse à partir de la table de vérité

En partant de la table de vérité, il est relativement aisé de définir la fonction logique. En effet, en raisonnant sur les valeurs «1» de la table, il suffit d'additionner (addition logique, donc fonction OU) les termes considérés. L'expression des termes est définie par le produit (multiplication logique, donc fonction ET) des variables d'entrées ou de leur complément considérés à la valeur logique «1».

L'exemple ci-dessous illustre ceci pour la détection de majorité sur trois variables.

Exemple : fonction majorité à trois variables

La fonction de majorité utilisée ici est définie comme vraie (état «1») si au moins deux des trois variables d'entrées (c, b, a) sont à l'état «1».

La table de vérité est la suivante :

c	b	a	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

On en déduit la fonction y :

$$y = \bar{c} \cdot b \cdot a + c \cdot \bar{b} \cdot a + c \cdot b \cdot \bar{a} + c \cdot b \cdot a$$

Tables de Karnaugh

La fonction obtenue par la méthode précédente n'est pas obligatoirement optimale dans le sens qu'elle ne fait pas nécessairement intervenir un nombre minimum d'opérations logiques. Ce point est important pour toute réalisation (discrète, intégrée ou encore logiciel) car le nombre d'opérations logiques fixe la complexité du système, en nombre d'éléments, en surface de silicium ou en temps de calcul. Il est donc intéressant de définir une méthode qui permette de simplifier l'expression obtenue au paragraphe 2. On peut simplifier une équation à l'aide de l'algèbre de Boole, mais il existe une méthode plus simple : les tables de Karnaugh. On se limitera à trois et quatre variables d'entrées.

Méthode

- On introduit la table de vérité dans une table de Karnaugh. Cette table est une représentation bidimensionnelle de la table de vérité. La figure 1.a illustre ceci pour les trois variables c, b et a. La figure 1.b illustre la table de Karnaugh pour les quatre variables d, c, b, et a.

	ba	00	01	11	10
c					
0					
1					

Figure 1.a

	ba	00	01	11	10
dc					
00					
01					
11					
10					

Figure 1.b

- On recherche ensuite visuellement les blocs (ensembles de cases adjacentes) de dimensions un, deux, quatre ou huit qui contiennent la valeur «1».
- On élimine tout bloc déjà contenu dans un ou plusieurs blocs plus grands.
- La fonction simplifiée se déduit de l'expression logique des blocs restants.

Remarque : Les blocs de dimension 1 s'expriment par le produit des 3 variables dans le cas d'un système à trois variables. Ces blocs s'expriment par le produit des quatre variables dans le cas d'un système à quatre variables.

Les blocs de dimension 2 s'expriment par le produit de 2 variables dans le cas d'un système à trois variables. Ces blocs s'expriment par le produit des trois variables dans le cas d'un système à quatre variables.

Les blocs de dimension 4 s'expriment par l'une des trois variables dans le cas d'un système à trois variables. Ces blocs s'expriment par le produit de deux variables dans le cas d'un système à quatre variables.

Les blocs de dimension 8 s'expriment par l'une des quatre variables dans le cas d'un système à quatre variables.

Exemple : fonction majorité à trois variables

La table de vérité du paragraphe 2.2 permet d'écrire la table de Karnaugh de la figure 2 :

	ba	00	01	11	10	
c						
0		0	0	1	0	a · b
1		0	1	1	1	a · c
						b · c

Figure 2

Remarque : Les blocs à une dimension n'ont pas été représentés pour ne pas surcharger la figure.

On en déduit la fonction simplifiée :

$$y = a \cdot b + a \cdot c + b \cdot c$$

Cette expression nécessite 5 opérations logiques contre 11 pour l'expression précédente.

Fonctions incomplètement définies

On appelle «fonction incomplètement définie » une fonction pour laquelle certaines valeurs des variables d'entrée ne nécessitent pas une valeur définie (« 0 » ou « 1 »). On note cet état par la lettre Φ . Le fait de faire intervenir cet état indéfini permet encore de simplifier l'équation logique.

Table de Karnaugh

La méthode reste la même que celle exposée au paragraphe 3 en prenant toutefois garde au fait que l'état Φ peut être considéré comme un « 0 » ou un « 1 ».

Remarque : Lorsque l'état Φ a été posé à « 0 » ou « 1 », il n'est plus possible de changer son état pour l'utiliser dans un autre bloc.

Système à fonctions multiples

On appelle «système à fonctions multiples » un système dans lequel les variables d'entrées sont combinées de façon à générer plusieurs fonctions de sortie.

Méthode

Une première approche de simplification consiste à déterminer les différentes fonctions de sortie à l'aide des tables de Karnaugh. On recherche ensuite dans l'expression de ces fonctions plusieurs termes communs. Il est clair que cette approche ne permet pas de garantir une simplification maximale des fonctions obtenues.

Une autre approche consiste à simplifier directement les fonctions de sortie en créant des tables de Karnaugh intermédiaires. Ces tables de Karnaugh font intervenir les produits des fonctions de sorties. Le volume V du Traité d'électricité, par exemple, donne plus de détail au sujet de cette méthode (§ 2.5).

Logique Séquentielle

La *logique séquentielle* se distingue de la logique combinatoire par le fait que dans cette dernière, les sorties ne réagissent qu'aux entrées, sans que le système ne soit sensible à l'histoire de ces entrées, ce qui est le cas en logique séquentielle. Il faut alors toujours prendre en compte les séquences d'entrée et de sortie du système que l'on veut analyser. Cet état de fait rend l'analyse et la synthèse de systèmes séquentiels plus pointue.

SYSTÈMES SYNCHRONES ET ASYNCHRONES

Il y a une autre distinction importante qu'il nous reste à présenter : les systèmes logiques peuvent être *synchrones* ou *asynchrones*.

- *Système asynchrone* : si on laisse un système électronique évoluer de lui-même, on disposera de l'information en sortie lorsque dès les délais électro-physiques (temps de propagations) se seront écoulés. Dans ce cas, le système est asynchrone : le temps est continu car la sortie peut être lue n'importe quand.

- *Système synchrone* : Dans ce cas, une horloge électronique cadence la marche du système, et on connaît les instants où l'on peut lire les sorties, on connaît le temps de réponse d'un système.