Cours de logique combinatoire

Eric SIMON
Ludovic MACAIRE

IUT A
Département Génie Mécanique et Productique

Janvier 2015

Table des matières

Chapitre 1 NUMERATION ET CODAGE

1. Introduction

Tout d'abord, il ne faut pas confondre « nombre » et « chiffre ». Les nombres entiers sont des objets mathématiques désignant une quantité. Un nombre peut s'écrire différemment suivant le langage dans lequel on l'écrit. Par exemple, toutes les écritures suivantes désignent le même nombre :

- « quatre » en français
- « four » en anglais
- « vier » en allemand
- « 4 » en écriture décimale
- « IV » en chiffre romain
- « 100 » en écriture binaires

Par ailleurs, deux écritures identiques peuvent avoir un sens différent. Par exemple dans deux langues différentes, on peut avoir :

- « pain » signifie *pain* en français
- « pain » signifie *souffrance* en anglais

C'est la même chose pour les nombres. Nous verrons ainsi que :

- « 110 » signifie *cent-dix* en écriture décimale
- « 110 » signifie *six* en écriture binaire
- « 110 » signifie *deux cent soixante douze* en écriture hexadécimale

Dans ce chapitre, nous allons étudier les différentes façons d'écrire les nombres entiers. Nous verrons également comment passer d'une écriture à une autre. Nous allons aussi étudier comment écrire les entiers négatifs, avec deux représentations différentes pour le signe. Enfin, nous verrons comment faire des additions et des soustractions directement en binaire.

2. Les systèmes de numération

Définition : la numération permet de représenter un nombre par la juxtaposition ordonnée de symboles pris parmi un ensemble donné.

2.1. Les différents systèmes de numération

Numération décimale :

Ce système de numération, le plus courant dans la vie quotidienne, dispose de …… symboles appelés chiffres. ……

Numération octale

Ce système utilise, non plus 10, mais ……. symboles : ……

Il n'est plus très employé aujourd'hui puisqu'il servait au codage des nombres dans les ordinateurs de première génération.

Numération binaire

Ce système de numération ne comprend que …….. symboles appelés BIT (BInary digiT) : …….

Remarque : Octet (byte en anglais) : ensemble de 8 bits. On exprime généralement la capacité des mémoires en Kilo-octets (Ko), un Ko vaut 1024 octets.

Numération hexadécimale

…… symboles sont utilisés : les chiffres de …….. et les lettes de …….

2.2. Principe d'une base

Le nombre de …….. que possède le système de numération est appelé base. Nous désignerons une base par la lettre B correspondant à ce nombre de symboles.

La base du système décimal est …….. (B=…..) alors que celle du système octal est ……. (B=…….).

Tout …….. peut s'écrire dans une base B quelconque.

Un nombre N s'écrit en juxtaposant n symboles :

……

Où les a_i sont les symboles de la base

Ce nombre N a pour valeur décimale :

……

On appelle cette forme la forme ……. Les termes B^i sont appelés …… ou ……... L'élément a_i est le symbole de rang i et son poids est B^i.

Exemple : base 10

$(485)_{10}$ = …….

Chaque chiffre du nombre est à multiplier par une puissance de 10 : c'est ce que l'on nomme le poids du chiffre. L'exposant de cette puissance est nul pour le chiffre situé le plus à droite et s'accroît d'une unité pour chaque passage à un chiffre vers la gauche.

Cette façon d'écrire les nombres est appelée …….

Exemple : base 8

Cette base obéira aux mêmes règles que la base 10, vue précédemment, ainsi on peut décomposer $(745)_8$ de la façon suivante

$(745)_8$ = ……

$(745)_8$ = ……

$(745)_8$ = ……

$(745)_8$ = ……

Lorsque l'on écrit un nombre, il faudra bien préciser la ……. dans laquelle on l'exprime pour lever les éventuelles indéterminations. Ainsi le nombre sera mis entre parenthèses (745 dans notre exemple) et indicé d'un nombre représentant sa base (8 est mis en indice).

Exemple : base 2

Dans le système binaire, chaque chiffre peut avoir 2 valeurs différentes : 0, 1. De ce fait, le système a pour base 2. Exemple : mot de 5 bits :

$(10\ 110)_2$ = ……

$(10\ 110)_2$ = ……

donc : $(10110)_2$ = ……...

Tous les systèmes de numération de position obéissent aux règles que nous venons de voir.

Exemple : base 16

Le système hexadécimal utilise les 16 symboles suivant : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. De ce fait, le système a pour base 16. Un nombre exprimé en base 16 pourra se présenter de la manière suivante : $(3BF)_{16}$ La correspondance entre base 2, base 10 et base 16 est indiquée dans le tableau ci-après :

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

Le nombre $(3BF)_{16}$ peut se décomposer comme suit : $(3BF)_{16} = 3 \times 16^2 + B \times 16^1 + F \times 16^0$

En remplaçant B et F par leur équivalent en base 10, on obtient :

$(3BF)_{16} = 3 \times 16^2 + 11 \times 16^1 + 15 \times 16^0$

$(3BF)_{16}$ = 3 x 256 + 11 x 16 + 15 x 1 donc

$(3BF)_{16} = (959)_{10}$

3. Changements de base

3.1. Méthodes de conversion d'un nombre décimal N en binaire

- Méthode directe (adéquate pour des petits nombres) :

On sait que $0<N<B^{n-1}$ (ici, B=2)

On détermine le nombre de bits minimum pour coder N

On positionne les bits a_i à 0 ou 1 de telle façon que :

$a_{n-1}.2^{n-1} + a_{n-2}.2^{n-2} + \ldots + a_0.2^0 = N$

Exemple : on veut coder 23

Codage sur 5 bits, car $2^5-1 = 31$

Il reste à positionner correctement les 5 bits pour obtenir la valeur 23 : …….

- Méthode des divisions euclidiennes successives

Soit $N = a_{n-1}.2^{n-1} + a_{n-2}.2^{n-2} + \ldots + a_0.2^0$, avec B=2

La méthode consiste à effectuer les divisions euclidiennes successives de N par 2 jusqu'à ce que le quotient devienne nul:

les …… de ces divisions lus de bas en haut représentent le nombre binaire.

Exemple : on veut coder 44

3.2. Méthode de conversion d'un nombre décimal N en octal

Comme pour la conversion en binaire, on effectue ici la méthode des divisions euclidiennes successives par……...

Exemple : $(47)_{10}$

3.3. Passage du binaire en octal

Exprimons $(47)_{10}$ dans le système octal et le système binaire. Nous obtenons :

Nous pouvons remarquer qu'après …….. divisions en binaire nous avons le même quotient qu'après une seule en octal. De plus le premier reste en octal obtenu peut être mis en relation directe avec les trois premiers restes en binaire :

$(111)_2$ = …….

$(111)_2$ = ……

et il en est de même pour le caractère octal suivant :

$(101)_2$ = ……..

$(101)_2$ = ……

Il suffit donc de faire des regroupements de …… bits sur le mot binaire. En effet, un mot de 3 bits permet de coder les nombres entiers décimaux compris entre 0 et 7.

Exemple de conversion binaire octal et octal binaire :

3.4. Conversion d'un nombre décimal N en hexadécimal.

Comme pour la conversion en binaire, on effectue ici la méthode des divisions euclidiennes successives par ……..

3.5. Passage du binaire en hexadécimal

Il suffit de faire des regroupements de …… bits sur le mot binaire. En effet, avec ……. bits il est possible de coder les nombres de …….. à …….

4. Représentation binaire des nombres signés

4.1. Remarque préliminaire sur l'addition de deux nombres binaires

L'addition de deux nombres binaires vérifie les règles suivantes :

4.2. Représentation “module plus signe”

C’est la solution la plus simple pour représenter les nombres négatifs. Un ……. ……. est ajouté au module pour la représentation du signe. Habituellement, on utilise la correspondance suivante :

Et le signe est placé à …… du module :

Exemple : $(-23)_{10}$: …….

Par conséquent si le format (nombre de bits utilisés) d’un nombre N est de n+1 éléments binaires, alors ce nombre est compris entre ……et ……...

Exemple : n=2

Remarque : il existe ……. représentations différentes du nombre 0.

Remarque : ce mode de représentation présente l’avantage d’être simple à mettre en œuvre, mais il se révèle peu adapté aux …….. Pour ce domaine d’application, il est préférable d’utiliser le mode ……...

4.3. Représentation des nombres en complément à 2

Le complément à 2 est le codage des nombres signés (+ et -). Il permet d'effectuer des soustractions.

4.3.1. Méthode de codage complément à 2

Définition : le complément à 2 N_{c2} d'un nombre N exprimé à l'aide de n chiffres dans la base binaire est : ……...

Remarque : ……..:

Comme le '1' de poids 2^n ne rentre pas dans le format défini des n bits, le nombre 2^n est interprété comme le nombre…….. par le calculateur, d'où : ……

Le complément à 2 s'identifie donc à …….. ……. du nombre N.

Nous allons maintenant proposer une méthode pour trouver rapidement le complément à 2.

On peut écrire $N_{c2} = 2^n – N$ = ……

……
……
……...

Exemple :

Remarque : chaque élément binaire de $(2^n-1) – N$ est l'inverse de l'élément correspondant de N. On parle de …… de N.

Finalement, l'opposé de N devient (-N) = ……..

En résumé : on calcule le complément à 1 du nombre. Le complément à 1 s'obtient en inversant chaque bit (les 0 deviennent des 1 et les 1 des 0). Puis on ajoute 1.

Par exemple, pour n = 4 :

N_{10}	N_2	compl. à 1	compl. à 2	Opposé (-N)
7				
6				
5				
4				
3				
2				
1				
0				

Remarques :

- On observe que l'élément le plus à gauche représente encore …..……., avec la convention :

- Le nombre 0 n'a qu'une représentation

- La combinaison ……. n'est jamais rencontrée. Cette combinaison résulte de la suppression de la double représentation du 0. Comme elle est représentative d'un nombre négatif, la valeur décimale correspondante est -8. En effet, l'addition de ce nombre avec le 1 donne -7.

- L'échelle représentative des nombres exprimés en complément à 2 sur n bits est :

……..

Ce codage permet donc la représentation des nombres négatifs utilisée dans les calculateurs.

Pour calculer la valeur numérique d'un nombre codé en complément à 2, on affecte un poids négatif au bit de poids fort :

Exemple : $(1010)_{C2} = 1.(-2)^3 + 0.(2)^2 + 1.(2)^1 + 0.(2)^0$

$(1010)_{C2} = 1.(-8) + 2 = -6$

4.3.2. Addition d'un nombre positif à un nombre négatif

Il suffit de remplacer le nombre négatif par son complément à 2.

Exemple avec 17 et -17:

Exemple avec 17 et -12 :

Exemple avec 12 et -17 :

4.3.3. Addition de deux nombres négatifs

On remplace chaque nombre par son complément à 2 et on effectue la somme.

Exemple avec -17 et -12 :

5. Code de Gray

Dans ce code,…….. change entre deux valeurs adjacentes.
Il sera utilisé dans les tableaux de karnaugh.

Chapitre 2 ALGEBRE DE BOOLE

L'algèbre de Boole est une structure algébrique qui ne contient que deux éléments, que l'on appelle couramment variables booléennes. Ces variables ne peuvent avoir que deux états, 1 ou 0 (vrai ou faux), et respectent quelques règles de calcul que nous détaillerons dans ce chapitre.

L'algèbre de Boole est très utilisé dans de nombreux domaines, par exemple en informatique, pour le calcul de conditions, ou en électronique.

L'algèbre de Boole utilise plusieurs opérateurs que l'on nomme opérateurs booléens, opérateurs logiques, ou encore fonctions logiques ou portes logiques (terme plus utilisé en électronique).
Les principaux opérateurs sont les opérateurs ET, OU, complément, que nous détaillons ci-dessous.

L'algèbre de Boole permet de modéliser des raisonnements logiques, en exprimant un « état » en fonction de conditions.

Par exemple (wikipedia) : **Décrocher = (Décision de répondre ET Sonnerie) OU décision d'appeler**

Décrocher serait « VRAI » soit si à la fois on entend la *sonnerie ET* l'on *décide de répondre*, soit (*OU*) si simplement l'on *décide d'appeler.*

1. Les opérations logiques élémentaires

1.1. Opérateur ET

L'opérateur ET correspond à **……..** de deux conditions. Par exemple : je souhaite acheter une voiture rouge ET climatisée. Les voitures correspondant à mon choix seront donc à la fois rouges et climatisées.

C'est donc un opérateur binaire qui affecte à la variable de sortie l'état 1 si et seulement si les variables d'entrées sont simultanément à 1.

Il est représenté par un point '.' ou '∧'.

Table de vérité

1.2. Opérateur OU

L'opérateur OU correspond à ……...de deux conditions. Par exemple :

je souhaite acheter une voiture rouge OU climatisée. Les voitures correspondant à mon choix seront donc soit rouges, soit climatisées, soit les deux à la fois.

C'est donc un opérateur binaire qui affecte à la variable de sortie l'état 1 si et seulement si une variable d'entrée est à 1. Il est représenté par le signe '+' ou 'V'.

Table de vérité

1.3. Opérateur NON

L'opérateur NON correspond au …….... d'une condition. Par exemple : je souhaite acheter une voiture NON diesel. Les voitures correspondant à mon choix seront donc des voitures essences.

Il est représenté par une barre.

Table de vérité

2. Les opérations logiques induites

2.1. L'opérateur NON ET ou NAND

Comme son nom l'indique, cette fonction logique correspond à l'association d'un NON et d'un ET.

Table de vérité :

C’est donc un opérateur binaire qui affecte à la variable de sortie l’état 0 si et seulement si les variables d’entrée sont à 1 simultanément. On note parfois l’opérateur NAND par une …… ……. ou par une ……..

Equation :

2.2. L’opérateur NON OU ou NOR

Cette fonction logique correspond à l’association d’un NON et d’un OU.

Table de vérité :

Il s’agit là aussi d’un opérateur binaire qui affecte à la variable de sortie l’état 1 si et seulement si les variables d’entrée sont à 0 simultanément. On note parfois cet opérateur par une flèche descendante.

Equation :

2.3. L’opérateur OU EXCLUSIF ou XOR

Cette fonction logique ne prend la valeur 1 que si une seule des entrées est à 1. On le note par un symbole + avec un rond.

Table de vérité :

Equation :

3. Définition d'une fonction logique (ou binaire)

Une **fonction binaire** est une application qui, à un mot binaire d'entrée $A = a_{n-1}…a_0$ associe une variable binaire X:

F : S^n -> S avec S = {0, 1}

A -> X

Une fonction binaire est dite …… si pour une des combinaisons d'entrées correspond un ……. de la sortie. La fonction est alors indépendante du temps. Lorsque ce n'est pas le cas, on parle de fonction séquentielle, ce qui constitue la deuxième partie du cours.

La **table de vérité** représente l'état de la variable de sortie X de la fonction pour chacune des combinaisons des n variables d'entrées $a_{n-1}…a_0$. Elle est donc composée de …… lignes.

Exemple d'une fonction de 3 variables :

Un **système logique** combinatoire est un système composé de plusieurs entrées binaires et de plusieurs sorties binaires. Les relations entres les variables de sortie et les variables d'entrée sont définies par des ……. Ce sont donc ces …… qui caractérisent le système.

4. Réalisation des fonctions logiques

Les fonctions logiques peuvent être réalisées avec plusieurs technologies différentes :

- Avec des circuits électriques, les fonctions binaires de base sont réalisées en mettant en série et en parallèle des contacts.
- Avec des composants électronique ou pneumatique. On dispose dans ce cas d'un certain nombre de composants qui permettent de réaliser directement les fonctions binaires de base (ET, OU, …).

4.1. Les circuits électriques

4.1.1. Définitions

Rappelons quelques définitions de base sur les circuits électriques : lorsqu'un courant électrique peut circuler dans un circuit électrique, ce circuit est dit …… ou ……. Il est dit ……. ou …… si le courant électrique ne peut circuler dans ce circuit.

Par conséquent un circuit électrique dit de commutation a seulement deux états logiques : l'état logique …… (le courant ne passe pas) et l'état logique …….. (le courant passe).

4.1.2. Les contacts électriques

Ce sont des contacts électriques qui, lorsqu'ils sont ……, vont permettre au circuit de passer d'un état à un autre (contact ouvert ou contact fermé). Ces contacts sont mis en œuvre par action mécanique, soit de type …….. (bouton poussoir), soit de type …… (relais électromagnétique). A chaque contact on affecte une variable binaire traduisant si le contact est actionné (…..) ou laissé au repos (……).

Un contacteur normal laisse passer le courant quand on l'actionne, un contacteur inverse quand on le laisse au repos.

Exemple : contact bouton poussoir normal :

On affecte à ce contact la variable binaire a.

C'est un contact qui est ouvert au repos (a=0) et qui se ferme lorsqu'il est actionné (a=1).

Contact inverse :

4.1.3. Réalisation de fonctions logiques à contact

Remarque : de nombreux automates programmables utilisent le langage à contacts (Ladder diagram) pour programmer des fonctions logiques.

4.2. Les circuits électroniques

Grâce aux systèmes numériques, on dispose de circuits réalisant les fonctions de base de la logique combinatoire. Ces circuits sont composés d'un à plusieurs transistors, mais nous ne détaillerons pas ces structures très complexes qui mettent en jeu de nombreux montages différents pour les transistors. Notons seulement qu'il existe de nombreuses familles logiques basées sur deux technologies : les familles TTL qui utilisent des transistors bipolaires et les familles MOS qui utilisent des transistors à effet de champ.

Les fonctions logiques de base sont aussi appelées des portes logiques. Le montage sera réalisé soit par association des fonctions "NON" "OU" "ET", soit uniquement à l'aide de NAND ou uniquement à l'aide de NOR ; le schéma établi porte alors le nom de logigramme.

Représentation des portes logiques NON, ET, OU, OU EX

Représentation des portes NAND, NOR

5. Les règles de base de l'algèbre binaire

Règles	Fonction OU	Fonction ET
Commutativité	$a + b =$	$a \cdot b =$
Associativité	$a + (b + c) =$	$a \cdot (b \cdot c) =$
Distributivité	OU par rapport à ET : $a + (b \cdot c) =$	ET par rapport à OU : $a \cdot (b + c) =$
Élément neutre	$a + 0 =$	$a \cdot 1 =$
Complémentaire	$a + \bar{a} =$	$a \cdot \bar{a} =$
Forçage (absorbant)	$a + 1 =$	$a \cdot 0 =$

Règles	Fonction OU	Fonction ET
Nihil Potence (involution)	$\bar{\bar{a}} =$	
Idem Potence	$a + a =$	$a \cdot a =$
Absorption 1	$a + ab =$	$a(a + b) =$
Absorption 2	$a + \bar{a}b =$	$a(\bar{a} + b) =$
Consensus	$ab + \bar{a}c + bc =$ (identité de Blake)	$(a + b)(\bar{a} + c)(b + c) =$
De Morgan	$\overline{a + b} =$	$\overline{a.b} =$

Démonstration de De Morgan :

- Complément d'une somme logique

Complétons le tableau suivant :

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

Conclusion :

Généralisation :

- Complément d'un produit logique

Complétons le tableau suivant :

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

Conclusion

Généralisation

6. Ecriture canonique d'une fonction logique à partir de sa table de vérité

6.1. Première forme canonique

Il va s'agir d'écrire l'équation sous la forme d'une somme de produits. Pour cela, il faut repérer dans la table de vérité toutes les combinaisons pour lesquelles F vaut 1.

Exemple :

On a deux lignes où la fonction vaut 1.

On écrit ces produits en fonction de la combinaison des variables : si une variable vaut 1 alors on écrit le nom de cette variable, si elle vaut 0 on écrit son nom complémenté. Ce qui donne ici :

On appelle parfois les produits issus d'une seule ligne de la table de vérité des mintermes. La première forme canonique est unique.

6.2. Deuxième forme canonique

Il va s'agir cette fois d'écrire l'équation sous la forme d'un produit de sommes. Pour cela, il faut repérer dans la table de vérité toutes les combinaisons pour lesquelles F vaut 0.

On écrit ces sommes en fonction de la combinaison des variables : si une variable vaut 0 alors on écrit le nom de cette variable, si elle vaut 1 on écrit son nom complémenté. Ce qui donne ici :

6.3. Représentation décimale d'une fonction logique

Pour condenser l'écriture de la fonction, il est possible de repérer les mintermes (ou maxtermes) par un numéro. Ces numéros peuvent être par exemple l'équivalent décimal du nombre binaire représenté par les variables d'entrée de la fonction :

N = ……..

où les a_i représentent les variables d'entrées. Cette méthode d'écriture nécessite donc de définir une convention de poids binaire pour chaque variable. Après avoir choisi une convention, il ne faut plus en changer.

Une fonction logique peut alors s'écrire comme une …… des états des variables d'entrées pour lesquelles elle vaut 1. On note alors F = …….

Elle peut également s'écrire comme le produit des états pour lesquelles elle vaut 0. On note dans ce cas F = …….

Exemple avec la fonction précédente :

7. Simplification d'une fonction logique

La représentation d'une fonction logique par une …….. n'est généralement pas la façon la plus condensée d'écrire cette fonction. Il peut y avoir des redondances, ce qui se traduirait par l'utilisation de plus de composants qu'il n'en faut pour réaliser électroniquement cette fonction, d'où l'intérêt de chercher la forme la plus condensée possible de la fonction.

Pour simplifier une fonction logique, il est possible d'utiliser les principes de l'algèbre de boole. Mais il n'est pas toujours facile de savoir si l'on a obtenu la forme la plus condensée de la fonction, ou s'il reste encore des simplifications à appliquer.

Il existe donc une autre méthode qui permet de trouver, de façon systématique, une expression algébrique la plus simple possible : c'est la représentation par les tableaux de Karnaugh.

7.1. Méthode algébrique

Cette méthode de simplification consiste à appliquer les principes de base de l'algèbre de Boole vus au paragraphe 4. On peut pour cela regrouper des termes afin de les simplifier :

On peut également ajouter un terme déjà existant dans le but de faire apparaître des simplifications :

On peut aussi supprimer un terme superflu :

7.2. Méthode avec les tableaux de Karnaugh

7.2.1. Définition

Le tableau de Karnaugh est une technique de représentation géométrique permettant des simplifications. Pour une fonction logique F de n variables, le tableau est constitué de 2^n cases (une case est associée à chaque état d'entrée). Chaque case contient la ……. de la fonction F correspondant à l'état d'entrée associé à cette case. Cette représentation correspond donc à une version synthétique d'une table de vérité, puisque une …….. de la table de vérité correspond à une …… du tableau de Karnaugh.

On utilise généralement les tableaux de Karnaugh pour la simplification des fonctions logiques de ……. à …….. variables.

7.2.2. Représentation des tableaux de Karnaugh

7.2.2.1. Cas de deux variables binaires

Dans le cas de deux variables binaires, nous avons quatre possibilités (ou combinaisons) à envisager correspondant à la table de vérité suivante :

A chaque combinaison des variables est associée une valeur de la fonction.

Le principe de KARNAUGH est d'associer une surface à chaque combinaison des variables, en adoptant la représentation suivante :

Enfin dans chaque case on met la valeur que prend la fonction pour la combinaison d'entrée correspondante.

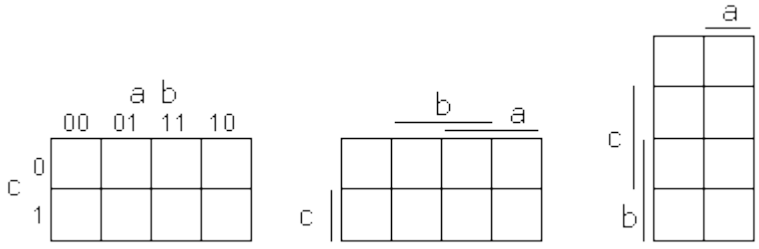
Une première simplification va consister à ne faire figurer que les ……. dans le tableau. La fonction s'écrit alors comme la …….. des produits de variables des cases valant …...

Remarque : Un tableau de Karnaugh peut se représenter sous les formes suivantes :

Ces trois représentations sont équivalentes**.**

7.2.2.2. Cas de trois variables binaires

Pour trois variables binaires, il y a 2^3 combinaisons, soit 8 cases :



Règle importante : adjacence des cases

Entre deux cases voisines, il faut qu'une et une seule variable change d'état. De telles cases sont dites adjacentes.

Ainsi pour deux variables, on obtient le codage suivant de a et b : …….. Ce codage correspond au code de ……...

7.2.2.3. Cas de quatre variables binaires

Pour 4 variables binaires a, b, c et d, nous avons 2^4 combinaisons, soit 16 cases :

7.2.3. Simplification avec un tableau de karnaugh

La méthode de simplification de Karnaugh consiste à …… des cases pour lesquelles la fonction vaut 1, par puissance de 2 les plus …….. possible. On lit la fonction en ne **conservant** pour chaque groupement que **……**. L'expression algébrique minimale est trouvée en déterminant le plus petit ensemble de groupements les plus grands possibles.

Soit le tableau de Karnaugh suivant:

On peut en déduire la fonction logique sous sa première forme canonique en repérant les 1 dans le tableau:

F = ……

On pourrait faire la simplification algébrique suivante :

Cela revient à regrouper les deux cases adjacentes ….. et ……, correspondant aux deux termes …….. et ……. On regroupe alors ces deux termes en un seul en supprimant la variable qui change d'état, soit …… ici :

On obtient F = …….

En résumé, on va entourer des groupes de 1 dans le tableau en respectant les règles suivantes:

- Les groupements peuvent utiliser plusieurs fois le même 1
- Ils ne doivent pas contenir de 0
- Tous les 1 doivent être contenus dans un groupement
- Ils doivent être les plus grands possible
- Ils doivent être rectangulaire (ou carré)
- Leur dimensions doivent être des puissances entières de 2 (2^0=1, 2^1=2, 2^2=4, etc) Exemples : un carré de 2 par 2 ou un rectangle de 1 par 4
- Le tableau est circulaire : on peut utiliser par exemple un 1 de la première colonne avec un de la dernière colonne

Nous présentons quelques exemples de regroupements possibles de cases :

- On peut faire des paquets de 2 variables :

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

On obtient alors F= …….

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

On obtient F = …….

- Des paquets de 4 variables :

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

On obtient F = …….

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

On obtient F = …….

- Ou un paquet de 8 variables :

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

On obtient F = …….

Remarque : utilisation des combinaisons physiquement impossibles :

Certaines combinaisons d'entrées d'un système ne pourront jamais se produire. On parle de combinaisons physiquement impossibles.

Comme ces combinaisons d'entrées ne se produiront jamais, on peut mettre la valeur …. dans le tableau de karnaugh pour effectuer des simplifications.

8. Résolution d'un problème de logique combinatoire

Maintenant que nous savons déterminer l'expression d'une fonction logique à partir de sa table de vérité, nous allons pouvoir résoudre des problèmes de logique combinatoire en concevant des systèmes logiques. La démarche générale est la suivante :

- Pour commencer, poser le problème correctement en envisageant tous les cas possibles. Pour cela on met l'énoncé sous la forme ……..……. en faisant apparaître toutes les variables indépendantes d'entrées. Dans certains cas, l'énoncé peut ne pas préciser l'état de sortie pour certaines combinaisons des variables, par exemple des combinaisons physiquement impossibles
- Établir le …… correspondant. Les cases correspondant aux combinaisons d'entrées impossibles ne seront pas remplies. Penser à utiliser ces cases pour les ……..
- Lire la fonction à partir du tableau en utilisant les règles de minimisation

Chapitre 3 Le multiplexeur - Démultiplexeur

1. Multiplexeur

1.1. Définition

Le multiplexeur (MUX) est un circuit qui réalise un …….. de l'une des entrées vers une sortie unique. Le MUX est également appelé sélecteur de données. L'aiguillage est fixé par une …….. Dans le cas particulier de deux entrées, un seul bit suffit pour la commande.

Exemple avec deux entrées :

Si la sélection est faîte sur quatre entrées, il faut …….. bits de commande. En généralisant à 2^n entrées, n bits de commandes sont nécessaires. On obtient alors le schéma général suivant :

Chaque entrée est repérée par une adresse codée sur n bits

Exemple : multiplexeur à 2 entrées

Multiplexeur à 4 entrées :

Remarque : Il existe également des multiplexeurs de 8 ou 16 entrées.

1.2. Réalisation pratique

1.2.1. Multiplexeur 2 vers 1

Le multiplexeur 2 vers 1 dispose de deux entrées et une commande, soit trois variables pour une sortie.

On note E_0 et E_1 les entrées et C_0 la commande. On écrit la table de vérité du multiplexeur :

On utilise la méthode de Karnaugh pour simplifier la fonction. On obtient alors le schéma suivant :

1.2.2. Multiplexeur 4 vers 1

Le multiplexeur 4 vers 1 dispose de 4 entrées et de deux bits de commande, soit 6 variables.
On obtient alors l'expression suivante:

Mais pour la généralisation à 2^n entrées et la réalisation en pratique, il est préférable de trouver une autre méthode.

On propose les deux solutions suivantes :

- Première solution

On rappelle la table de fonctionnement du multiplexeur 4 vers 1 :

On remarque que c'est …. qui fait la sélection entre E_0 et E_1 d'une part, et entre E_2 et E_3 d'autre part. La sélection entre les deux configurations s'effectue par ….., d'où l'utilisation de multiplexeur 2 vers 1 en cascade.

- Deuxième solution

On a vu que le multiplexeur 4 vers 1 était équivalent à un interrupteur à quatre positions :

Une autre manière de le représenter est de prendre 4 interrupteurs à 2 positions dont un seul est fermé :

C'est le décodeur binaire qui réalise le décodage de la combinaison $(C_1\ C_0)_2$

Généralisation : un multiplexeur à 2^n entrées comporte 2^n interrupteurs dont un seul peut être fermé. Les interrupteurs sont commandés par les sorties d'un décodeur binaire à n variables d'entrées.

1.3. Génération de fonctions

Les multiplexeurs sont autant utilisés car ils permettent de réaliser n'importe quelle fonction logique avec un seul boîtier logique. On fixe les entrées à des valeurs données, suivant la

fonction à réaliser, et on utilise les entrées de sélection comme entrée de la fonction :

Exemple avec une fonction de deux variables :

Nous avons vu précédemment que toute fonction logique combinatoire peut se mettre sous une forme ……. La fonction de deux variables a et b se développe ainsi suivant l'expression :

où F(i, j) est la valeur particulière de la fonction logique lorsque a=i et b=j.

Un multiplexeur à 4 entrées, donc 2 commandes, écrit une sortie S reliée aux commandes C_1 C_0 par la relation :

Notons que ces deux expressions ont une forme très proche. On voit ainsi qu'il est possible de réaliser toutes les fonctions de deux variables en identifiant …. à F(a, b), ….. à a, …… à b, les valeurs des entrées du multiplexeur à celles de la fonction.

Ainsi les bits de commande du multiplexeur sont alors les variables de la fonction, et les entrées du multiplexeur permettent de sélectionner la fonction à réaliser.

On obtient le schéma suivant :

Exemple avec un multiplexeur à 4 entrées :

2. Démultiplexeur

Le démultiplexeur est un circuit qui réalise également un aiguillage d'information, mais dans le sens inverse d'un multiplexeur :