7. Simplification des fonctions booléennes

L'objectif de la simplification des fonctions logiques est de :

- Réduire le nombre de termes dans une fonction, ce qui permet de réduire le nombre de portes logiques

- Obtenir un circuit plus petit, plus rapide et moins cher

7.1 Simplification algébrique

Elle consiste à appliquer les règles de l'algèbre de Boole afin d'éliminer des variables ou des termes

Règle 1: Appliquer les règles suivantes :

$$A.B + A.\overline{B} = A \qquad (A + B)(A + \overline{B}) = A$$
$$A + A\,B = A \qquad A(A + B) = A$$
$$A + \overline{A}\,B = A + B \qquad A(\overline{A} + B) = AB$$

Exemple

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

2) $F(A,B,C) = ABC + AB\overline{C} + A\overline{B}CD = AB(C+\overline{C}) + A\overline{B}CD$
$= AB + A\overline{B}CD$
$= A(B + \overline{B}(CD))$
$= A(B + CD)$
$= AB + ACD$

Règle 2: Rajouter un terme déjà existant

Exemple

$A\,B\,C + \overline{A}BC + A\overline{B}C + AB\overline{C} =$
$ABC + \overline{A}BC + ABC + A\overline{B}C + ABC + AB\overline{C} =$
$BC + AC + AB$

Règle 3: Simplifier la forme canonique ayant le nombre de termes minimum

Exemple : simplifiez la fonctions suivante $F(A,B,C) = \sum (2,3,4,5,6,7) = A+B$

7.2 Table de Karnaugh

En examinant la méthode de simplification algébrique on remarque que cette dernière devient très difficile si le nombre des variables et grand. La méthode de Karnaugh est une technique de simplification rapide.

- **Principe d'adjacence**

Examinons l'expression suivante : $A \cdot B + A \cdot \overline{B}$

Les deux termes possèdent les mêmes variables. La seule différence est l'état de la variable B qui change

Ces termes sont adjacents

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

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

$A.B.C.D + A.B.\overline{C}.D = A.B.D$

Ces termes ne sont pas adjacents

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

$A.B.C + A.\overline{B}.\overline{C}$

$A.B.C.D + \overline{A}.B.\overline{C}.D$

- **Principe de la méthode**

- C'est un tableau de 2^n cases, N étant le nombre de variables.
- La méthode peut s'appliquer aux fonctions logiques de 2,3,4,5 et 6 variables.
- Les lignes et les colonnes sont numérotées en code Gray
- Pour chaque min terme (TV) lui correspond une case égale 1
- Pour chaque max terme (TV) lui correspond une case égale à 0
- Les 0 peuvent être omis pour alléger l'écriture

- **Simplification**

- On repère les cases adjacentes contenant un 1 et on les regroupe par paquets de 2^N (32, 16, 8, 4, 2, 1)
- Deux cases sont adjacentes lorsqu'elles sont situées côte à côte horizontalement et verticalement
- Deux cases situées aux extrémités d'une même ligne ou d'une même colonne sont adjacentes (forme cylindrique)
- les 4 cases des coins sont des cases adjacentes
- La taille d'un groupe doit être une puissance de 2
- Avec la méthode de Karnaugh on essaye de faire le minimum des regroupements (minimisation du nombre de termes) qui contiennent le maximum de 1 (minimisation du nombre de variables)

- On élimine les variables qui **changent d'états dans un groupe**
- La fonction finale est égale à la somme des termes dont l'état des variables ne change pas à l'intérieur d'un regroupement.
- une ou plusieurs cases peuvent être communes à plusieurs regroupements
- On s'arrête lorsqu'il y a plus de 1 en dehors des regroupements

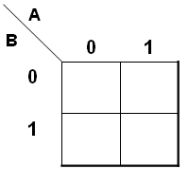


Tableau à 2 variables

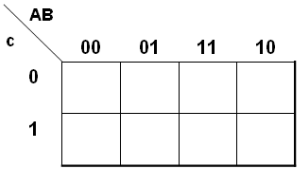


Tableau à 3 variables

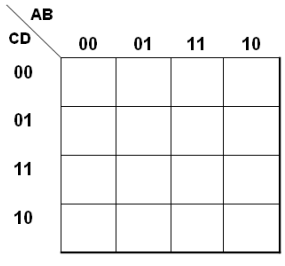


Tableau à 4 variables

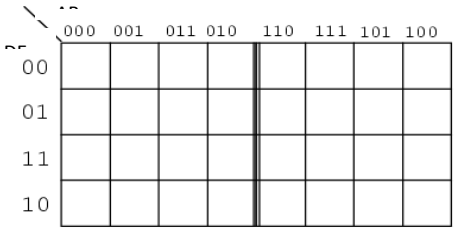
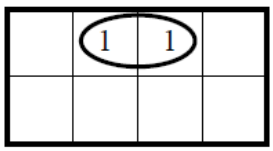


Tableau 5 variables

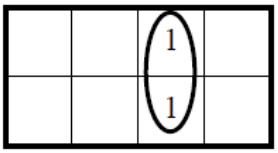
Remarques

- Au-delà de 6 variables, la méthode de Karnaugh n'étant plus valable, on utilise la méthode de Mac Cluskey

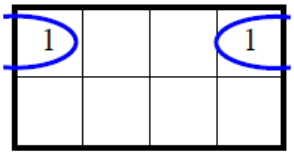
Exemples



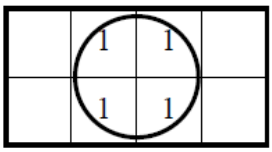
$F = B\overline{C}$



$F = AB$

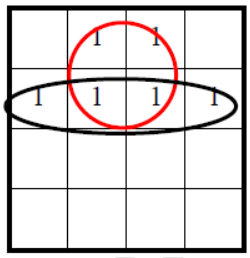


$F = \overline{B}\,\overline{C}$

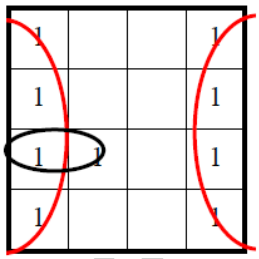


$F = B$

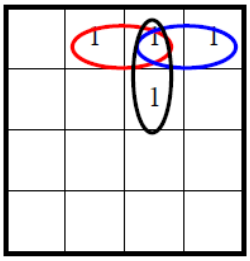
$F = \overline{C}$



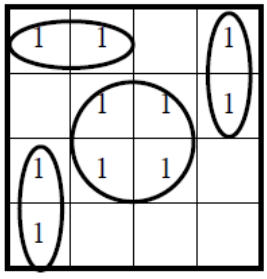
$F = B\overline{C} + \overline{C}D$



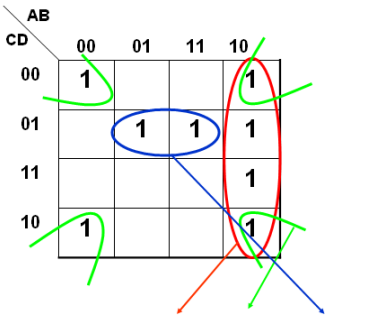
$F = \overline{B} + \overline{A}\,CD$



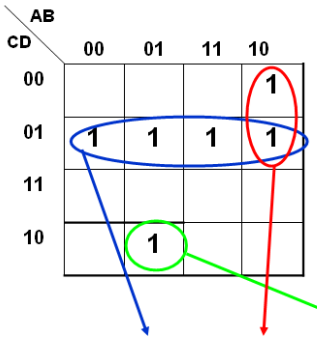
$= B\overline{C}\,\overline{D} + A\overline{C}\,\overline{D} + AB\,\overline{C}$



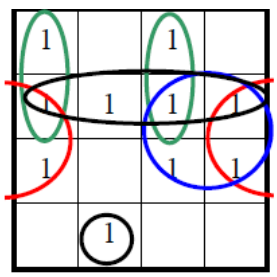
$= BD + \overline{A}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C$



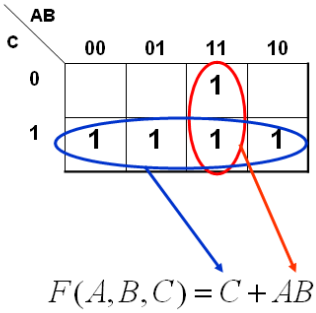
$F(A,B,C,D) = A\overline{B} + \overline{B}\,\overline{D} + B\overline{C}D$



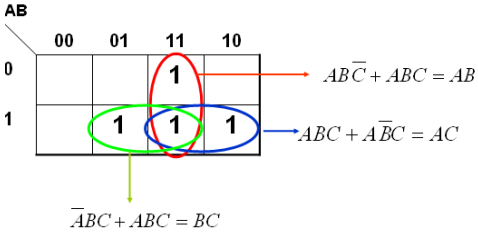
$F(A,B,C,D) = \overline{C}.D + A.\overline{B}.\overline{C} + \overline{A}.B.C.\overline{D}$



$= \overline{C}D + AD + \overline{B}\,D + \overline{A}\,\overline{B}\,\overline{C} + AB\,\overline{C} + \overline{A}\,BC\overline{D}$



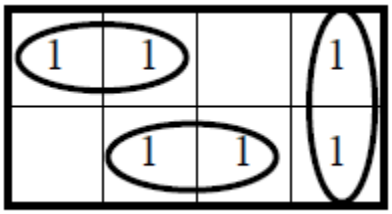
$F(A,B,C) = C + AB$



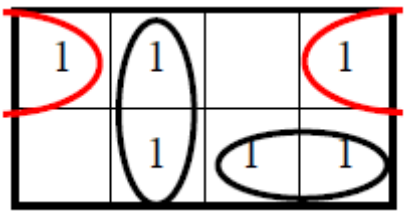
$F(A,B,C) = AB + AC + BC$

Remarques

- Le résultat de la simplification peut ne pas être unique.



$$= \overline{A}\,\overline{C} + BC + A\overline{B}$$



$$= \overline{B}\,\overline{C} + \overline{A}\,B + AC$$

- On peut appliquer la méthode de Karnaugh sur des Maxterms

1	1	1	1
1	1	1	1
1	0	0	1
1	1	1	1

$$= \overline{B} + \overline{C} + \overline{D}$$

- **Cas des fonctions incomplètement spécifiées**

Une fonction est dite incomplètement **spécifiée** quand sa valeur est indifférente (ne change pas le résultat) ou n'existent pas pour certaines combinaisons de variables d'entrées. On utilise le symbole X ou ∅ pour la valeur non spécifiée de la fonction.

Dans le tableau de Karnaugh, le symbole X peut prendre indifféremment la valeur 0 ou 1, on remplace donc par 1 uniquement ceux qui permettent de simplifier une expression par regroupement (augmenter la taille du groupe).

Exemple :

		X	
1	1		1
X	X	X	X
1	1	X	X

$$= C + \overline{A}\,D + \overline{B}\,D.$$

Références

Livres disponibles(Bibliothèque de la faculté)

1.Architecture des ordinateurs. Jean-Jacques Schwarz 2005
2. Architecture et technologie des ordinateurs. Zanella, Paolo 2005
3. Architecture des ordinateurs. Philippe Darche 2002
4. Architecture des machines et des systèmes informatiques. Alain Cazes 2003
5. Architecture de l'ordinateur. Robert Strandh 2005
6. Architecture de l'ordinateur. Andrew Tanenbaum 2005
7. De l'Algèbre de Boole aux circuits numériques. Karima Khadidja Mokhtari 2014
8. Logique combinatoire et composants numériques. Mouloud Sbai 2013
9. Introduction aux circuits logiques. Letocha Jean 1985 (Bibliothèque centrale)

Livres téléchargeables

1.D A Patterson & J L Hennessy, Computer Organization and Design : The hardware/software interface, Morgan-Kaufmann (Fifth edition) 2013

2.A S Tanenbaum and T Austin, Structured Computer Organization, Pearson (International edition), 2012

3.R E Bryant & D R O'Hallaron, Computer Systems : A Programmer's Perspective, Pearson (Global edition) 2015

4.Computer Organization and Architecture 8TH EDITION by William Stallings. Prentice Hall, Inc.,2010

5.Computer System Architecture (3rd Edition) M. Morris Mano 1992