ELEC-H-305

Circuits logiques et numériques

2010-2011

Cours 4

Dragomir Milojevic

Dragomir.Milojevic@ulb.ac.be

Récapitulation

Mintermes/Maxtermes

Dans une TdV:

Mintermes

*C'est la **combinaison des variables** (avec un **ET**) pour lesquels la fonction logique vaut **1***

Maxtermes

*C'est la **combinaison des variables inversées** (avec un **OU**) pour lesquels la fonction logique vaut **0***

Fonction logique apparaît comme une somme de mintermes (ou le produit des maxtermes).

Représentations des fonctions

Récapitulation

1. **Tables de vérité**
2. Expressions
   I. **Somme de produit** (SdP)
   II. **Produit de somme** (PdS)
3. Forme des expressions
   I. Si les termes de l'expression sont les mintermes (ou les maxtermes): alors c'est la **forme canonique standard**
   II. Si les termes de l'expression sont les monômes: alors c'est la **forme canonique non-standard**

Représentations des fonctions

Passages d'une forme à l'autre

Table de vérité ➟ Forme canonique standard
Directement : la somme (le produit) des Mintermes (des Maxtermes). Il s'agit d'une simple ré-écriture.

Forme canonique standard ➟ non-standard
Simplification: axiomes et théorèmes (plus d'autres méthodes que nous allons voir par la suite ...). Implique la "qualité" d'une solution.

Forme canonique non-standard ➟ standard
Expansion des monômes, TdV. Donner une explication à une expression logique existante (*reverse engineering*).

Exemple complet de passage

Une fonction logique à trois variables:

Somme des Produits (SdP)

x	y	z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Mintermes:

x'y'z', x'yz', xy'z', xyz'

Equivalents décimaux:

0, 2, 4, 6

Fonction logique (SdP):

F= x'y'z'+x'yz'+ xy'z'+ xyz'

Exemple complet de passage

Produit de Somme: PdS

x	y	z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Maxtermes:

x+y+z', x+y'+z', x'+y+z', x'+y'+z'

Equivalents décimaux:

1, 3, 5, 7

Fonction logique (PdS):

F=(x+y+z')(x+y'+z')(x'+y+z')(x'+y'+z')

Exemple complet de passage

Simplification à l'aide des axiomes et des théorèmes (en partant de SdP):

```
F = x'y'z'+x'yz'+ xy'z'+ xyz'
  = x'z'(y+y')+xz'(y+y')
  = z'(x+x')
  = z'
```

Interprétation "logique":
Fonction `F` vaut `1` lorsque `z` vaut `0`, quelque soit les valeurs de `x` et de `y`

x	y	z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Exemple complet de passage

Pour retrouver les mintermes:

Expansion des monômes

Un seul monôme d'une seule variable z

Manquent x et y (il faut multiplier par (x+x')(y+y'))

$$F(x,y,z) = z'$$
$$= z'(x+x')(y+y')$$
$$= (z'x+z'x')(y+y')$$
$$= z'xy + z'xy' + z'x'y + z'x'y'$$
$$= xyz' + xy'z' + x'yz' + x'y'z'$$
$$= x'y'z' + x'y\,z' + xy'\,z' + xy\,z'$$
$$= \sum(0, 2, 4, 6)$$
$$= \prod\,(1, 3, 5, 7)$$

Représentation des fonctions

SdP sous forme canonique standard n'est pas optimale (penser à la réalisation physique: portes - circuit)

Nécessite la **simplification**.

Objectif de la simplification

A partir d'une expression quelconque arriver à une **expression équivalente** en utilisant le **moins de termes** possible (chaque terme supplémentaire implique une porte en plus)

Simplifications des fonctions

Méthodes de simplification envisagées

1. **A l'aide des axiomes et des théorèmes déjà démontrées**

   Problème: la "qualité" de l'expression dépend fortement de la capacité de manipulation des axiomes et des théorèmes.

2. **Tables de Karnaugh (Maurice Karnaugh, 1953)**

   Méthode graphique permettant de rendre la simplification plus aisée. Utilisée pour la simplification des problèmes avec peu de variables (jusqu'au 5).

3. **Méthode de Quine-McCluskey**

   Méthode systématique permettant de trouver l'expression la plus simple. Avantages: on peut résoudre les problèmes à grand nombre de variables et facile à automatiser.

Tables de Karnaugh

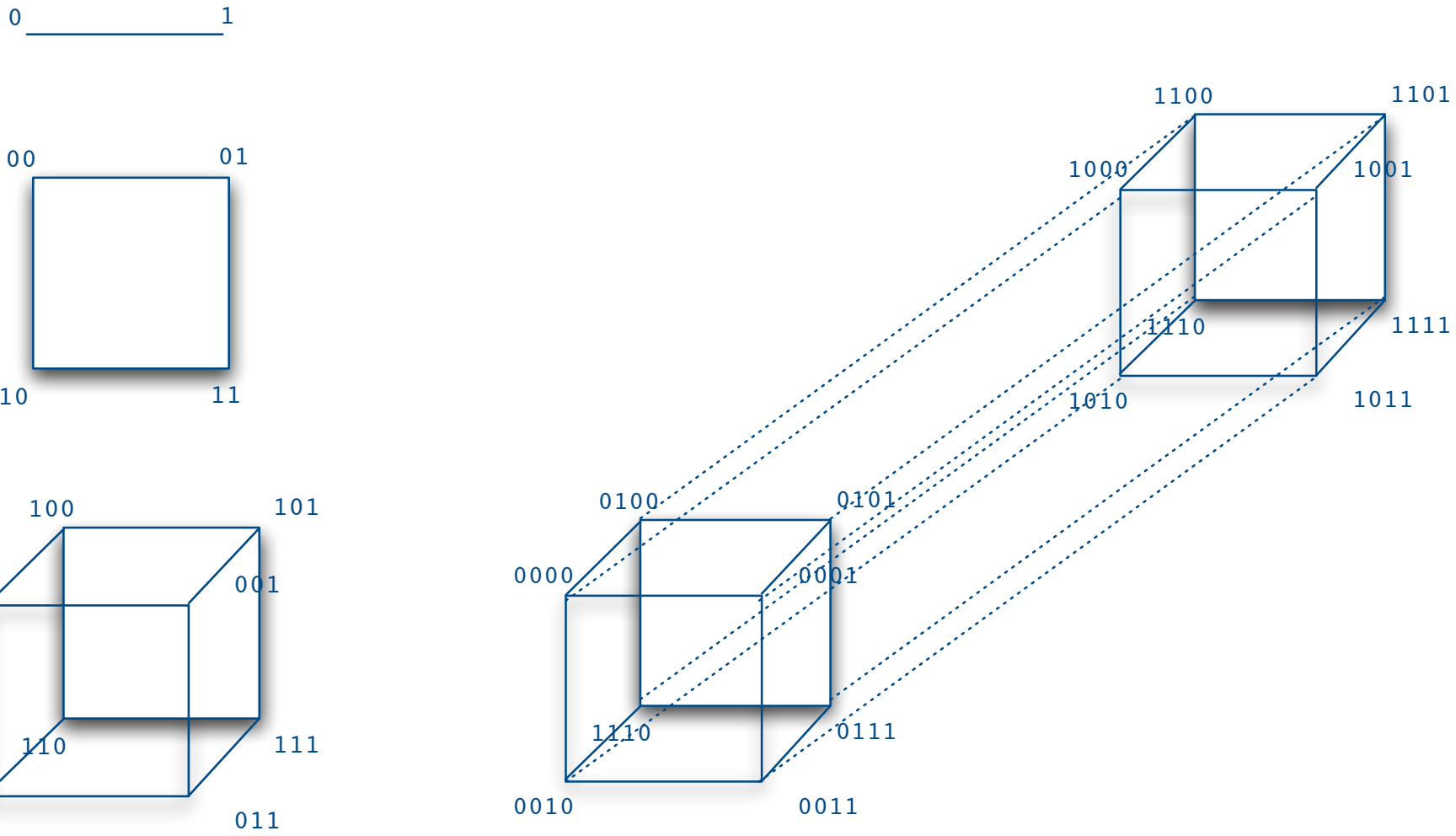
Notion de *n*-cube (rappel)

1. *n*-cube est un cube à *n*-dimensions avec 2^n sommets
2. Un mot binaire de *n*-bits est représenté comme un des sommets d'un *n*-cube
3. La distance de Hamming entre deux sommets appartenant à une même arrête est 1

Avantage : lorsque *n* est petit, le *n*-cube peut être représenté graphiquement

Tables de Karnaugh

Représentation graphique de *n*-cube



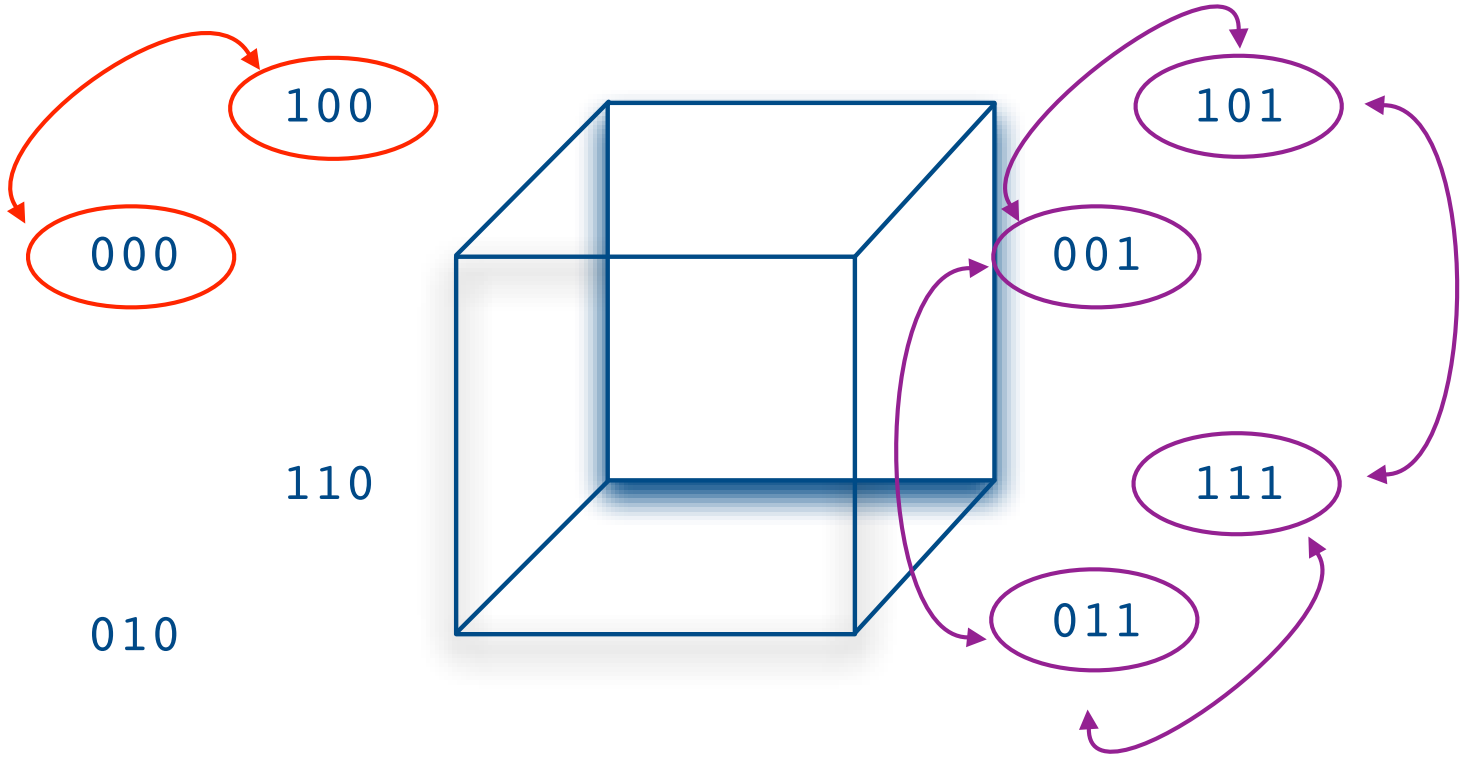
Tables de Karnaugh

Distance entre les sommets permet de introduire :

Notion d'**adjacence**

Deux sommets sur une même arrête ne diffèrent que d'un seul bit.

Pour un *3*-cube :



Tables de Karnaugh

La table de Karnaugh (K-Map)

La K-Map est une Table de Vérité agencée selon le principe des *n*-cubes !

➟ **C'est une représentation graphique (en 2D) des *n*-cubes**

La création d'une K-map:

1. A partir d'une TdV
Il suffit de la réorganiser correctement...

2. A partir d'une expression
A partir des mintermes ou des monômes (on peut toujours faire l'expansion par la pensé ou en développant)

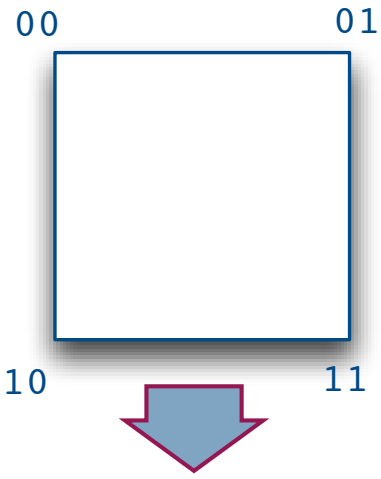
Tables de Karnaugh

K-Map à deux variables

TdV

	MSB x_2	LSB x_1
0	0	0
1	0	1
2	1	0
3	1	1

n-cube



$F(x_2, x_1)$	0	1
0	$(0)_{10}$	$(1)_{10}$
1	$(2)_{10}$	$(3)_{10}$

x_1

x_2

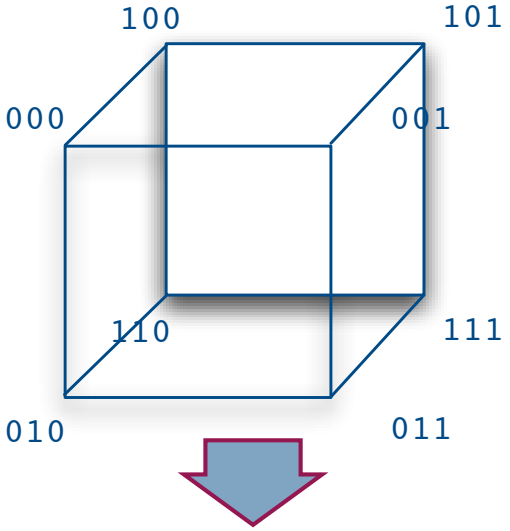
Valeur de la fonction

Tables de Karnaugh

K-Map à trois variables

TdV

	MSB x_3	x_2	LSB x_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1



$F(x_3,x_2,x_1)$	x_2' 00	01	x_2 11	10	x_2x_1
0	(0)$_{000}$	(1)$_{001}$	(3)$_{011}$	(2)$_{010}$	
1	(4)$_{100}$	(5)$_{101}$	(7)$_{111}$	(6)$_{110}$	
x_3					

x_1

Attention à l'inversion de l'ordre !

Tables de Karnaugh

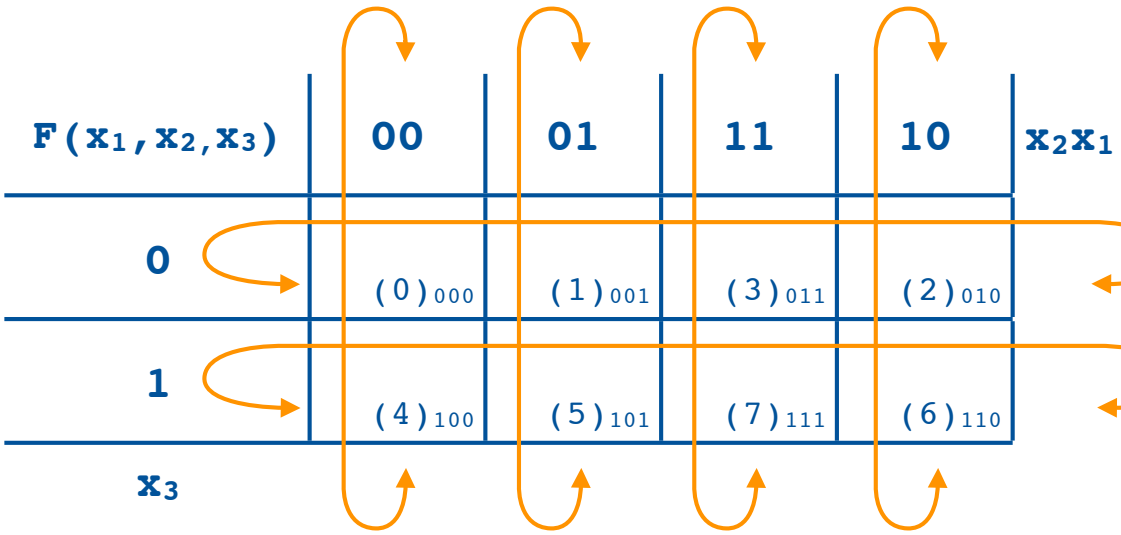
K-Map à trois variables (adjacences)

$(0)_{10}$ $(4)_{10}$ ➡ $(000)_2$ $(100)_2$

$(1)_{10}$ $(5)_{10}$ ➡ $(001)_2$ $(101)_2$

$(3)_{10}$ $(7)_{10}$ ➡ $(011)_2$ $(111)_2$

$(2)_{10}$ $(6)_{10}$ ➡ $(010)_2$ $(110)_2$



$(0)_{10}$ $(2)_{10}$ ➡ $(000)_2$ $(010)_2$

$(4)_{10}$ $(6)_{10}$ ➡ $(100)_2$ $(110)_2$

Tables de Karnaugh

K-Map à quatre variables

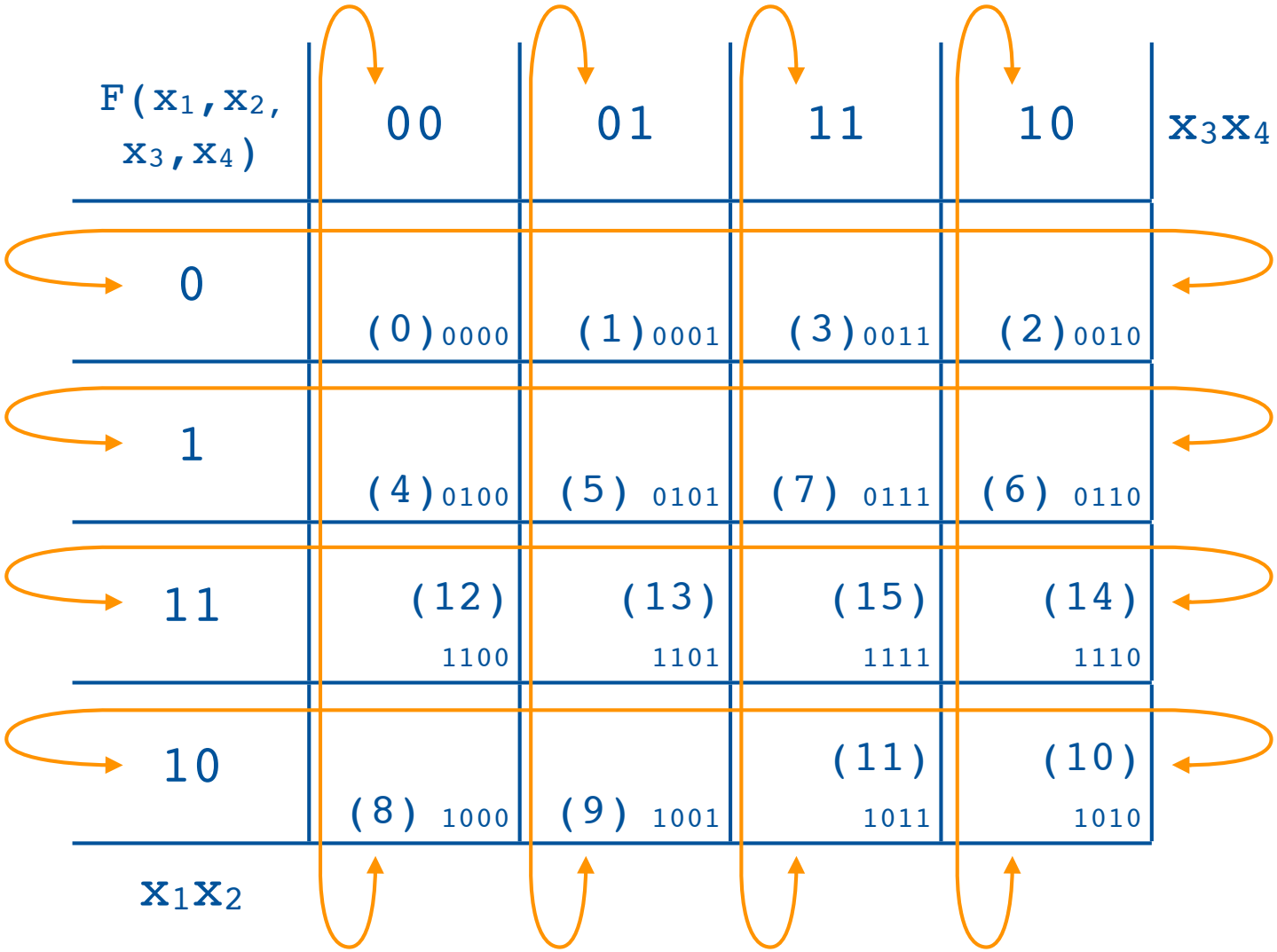
	x_4	x_3	x_2	x_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

$F(x_1,x_2,x_3,x_4)$	00	01	11	10	x_2x_1
00	(0) 0000	(1) 0001	(3) 0011	(2) 0010	
01	(4) 0100	(5) 0101	(7) 0111	(6) 0110	
11	(12) 1100	(13) 1101	(15) 1111	(14) 1110	
10	(8) 1000	(9) 1001	(11) 1011	(10) 1010	
x_4x_3					

Attention à l'inversion de l'ordre !

Tables de Karnaugh

K-Map à quatre variables (adjacences)



Tables de Karnaugh

A cinq variables

$F(x_1,x_2,x_3,x_4,x_5)$	00	01	11	10	x_4x_5
000	(0)	(1)	(3)	(2)	
001	(4)	(5)	(7)	(6)	
011	(12)	(13)	(15)	(14)	
010	(8)	(9)	(11)	(10)	
100	(16)	(17)	(19)	(18)	
101	(20)	(21)	(23)	(22)	
111	(28)	(29)	(31)	(30)	
110	(24)	(25)	(27)	(26)	
$x_1x_2x_3$					

TdV ...

Tables de Karnaugh

Une autre façon de voir la K-Map à cinq variables

$F(x_1,x_2,x_3,x_4,x_5)$	00	01	11	10	x_4x_5
00	(0)	(1)	(3)	(2)	
01	(4)	(5)	(7)	(6)	
11	(12)	(13)	(15)	(14)	
10	(8)	(9)	(11)	(10)	
x_2x_3					

$x_1=0$

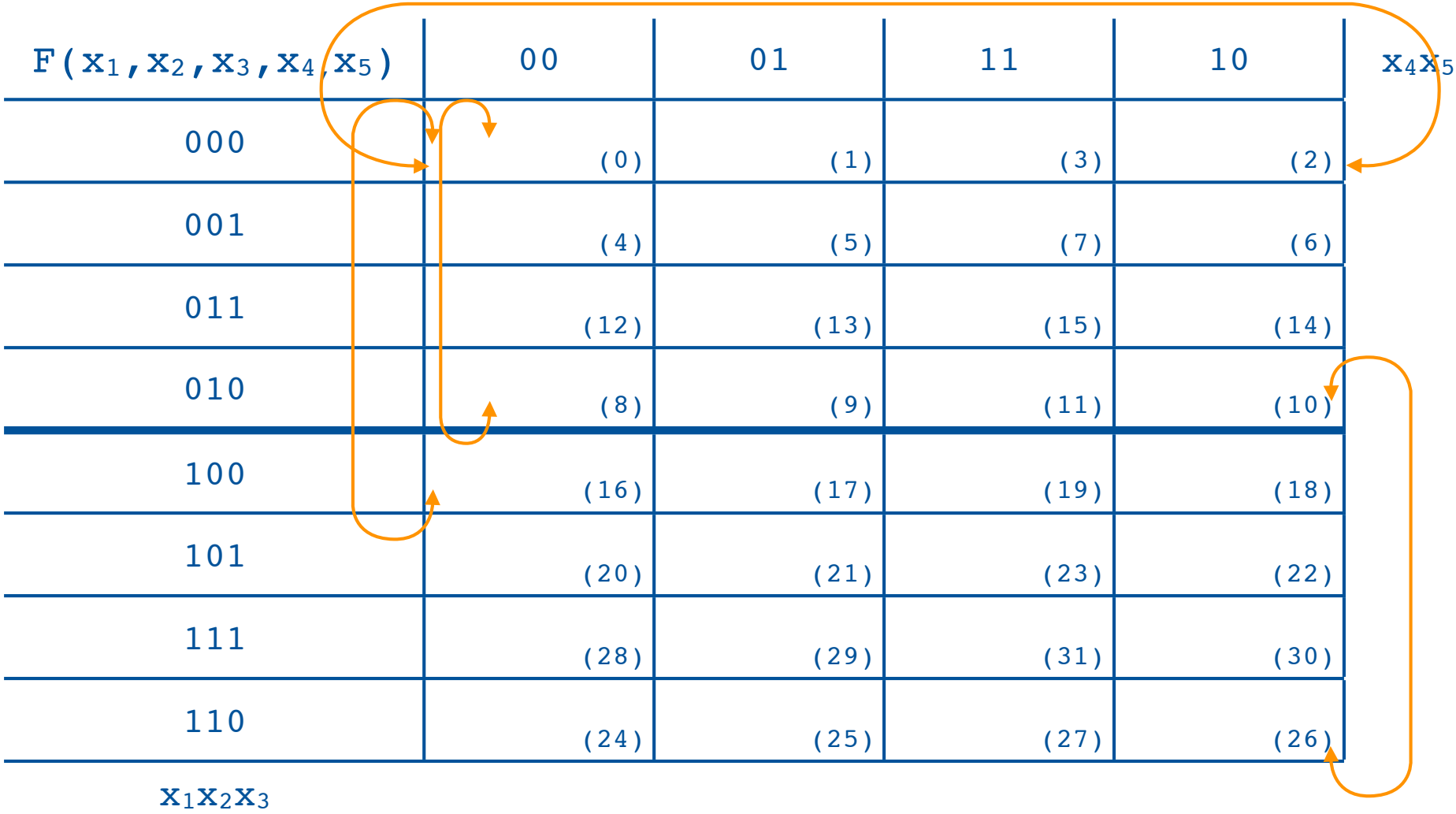
Deux K-Maps de 4 variables superposées (selon x_1)!

$F(x_1,x_2,x_3,x_4,x_5)$	00	01	11	10	x_4x_5
00	(16)	(17)	(19)	(18)	
01	(20)	(21)	(23)	(22)	
11	(28)	(29)	(31)	(30)	
10	(24)	(25)	(27)	(26)	
x_2x_3					

$x_1=1$

Tables de Karnaugh

A cinq variables (adjacences)



$F(x_1,x_2,x_3,x_4,x_5)$	00	01	11	10
000	(0)	(1)	(3)	(2)
001	(4)	(5)	(7)	(6)
011	(12)	(13)	(15)	(14)
010	(8)	(9)	(11)	(10)
100	(16)	(17)	(19)	(18)
101	(20)	(21)	(23)	(22)
111	(28)	(29)	(31)	(30)
110	(24)	(25)	(27)	(26)

x_4x_5

$x_1x_2x_3$

Tables de Karnaugh

... et a six variables (pour information uniquement)

	000	001	011	010	100	101	111	110
000	(0)	(1)	(3)	(2)	(4)	(5)	(7)	(6)
001	(8)	(9)	(11)	(10)	(12)	(13)	(15)	(14)
011	(24)	(25)	(27)	(26)	(28)	(29)	(31)	(30)
010	(16)	(17)	(19)	(18)	(20)	(21)	(23)	(22)
100	(32)	(33)	(35)	(34)	(36)	(37)	(39)	(38)
101	(40)	(41)	(43)	(42)	(44)	(45)	(47)	(46)
111	(56)	(57)	(59)	(58)	(60)	(61)	(63)	(62)
110	(48)	(49)	(51)	(50)	(52)	(53)	(55)	(54)

Tables de Karnaugh

Notion de sous-cube dans un n-cube

Un ensemble de 2^m sommets (appartenant à une arrête, une face,...) pour lesquels *n-m* variables restantes ont la même valeur:

$x_1, x_2, x_3 \ldots\ x_n$

$(l_1, l_2, l_3, \ldots, l_{n-m})(x_{n-m+1} \ldots\ x_n)$

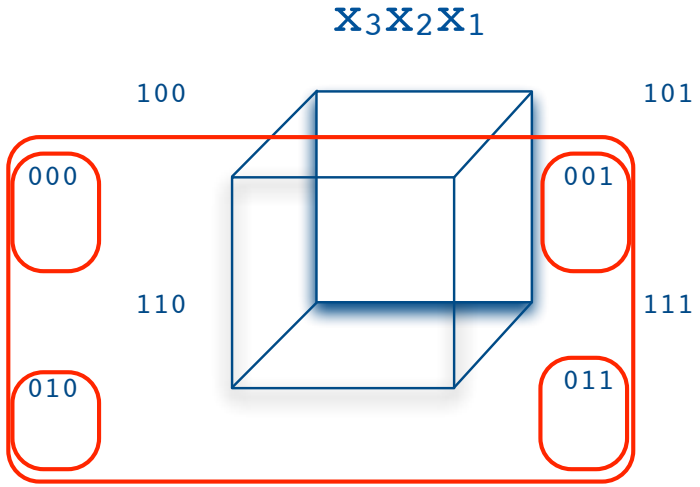
Partie fixe — Partie variable

Exemple:

3 variables, 4 sommets;

n=3, m=2 $(x_3)(x_2x_1)$

Partie fixe — Partie variable



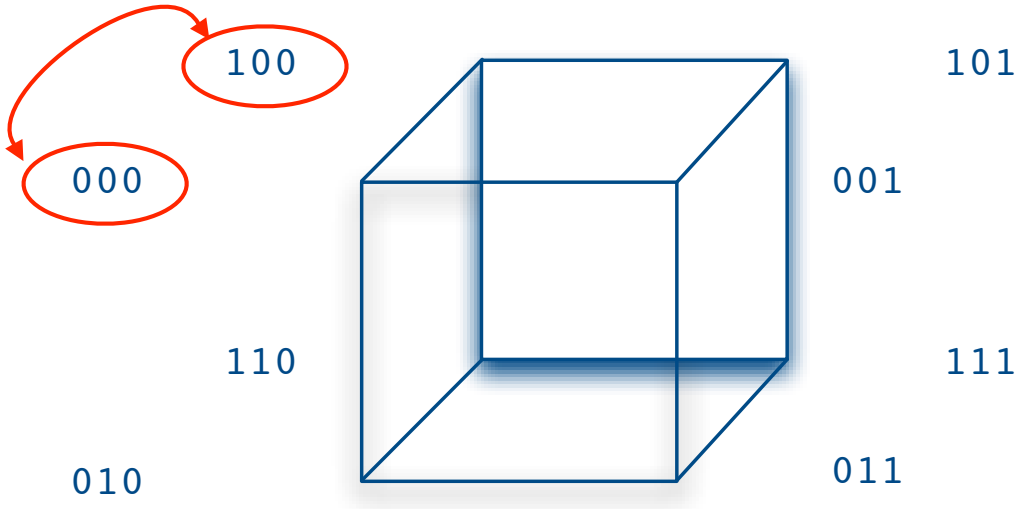
Interprétation:

Lorsque l_1 ou l_2 ou l_3 ou... ou l_{n-m} , et quelques soient $x_{n-m+1} \ldots\ x_n$ la fonction logique vaut '1'

Tables de Karnaugh

- Un sous-cube: sommets 000 et 100
- Les deux sommets correspondent à $x_1'x_2'x_3'$ et $x_1x_2'x_3'$
- Si la **fonction logique vaut 1** aux sommets 000 et 100 alors:

$$F = x_1'x_2'x_3' + x_1x_2'x_3' = (x_1'+x_1)x_2'x_3' = x_2'x_3'$$



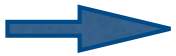
Lorsque x_2 et x_3 valent 0, quelque soit x_1, alors $F=1$

Tables de Karnaugh

Sous-cubes de taille 2 pour une K-Map à deux variables (tous les sous-cubes possibles)

F	0	1	x_2
0			
1			

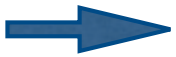
x_1



x_1', x_1

F	0	1	x_2
0			
1			

x_1



x_2', x_2

Tables de Karnaugh

Sous-cubes de taille 2 pour une K-Map à trois variables...

F	00	01	11	10	x_2x_3
0					
1					
x_1					

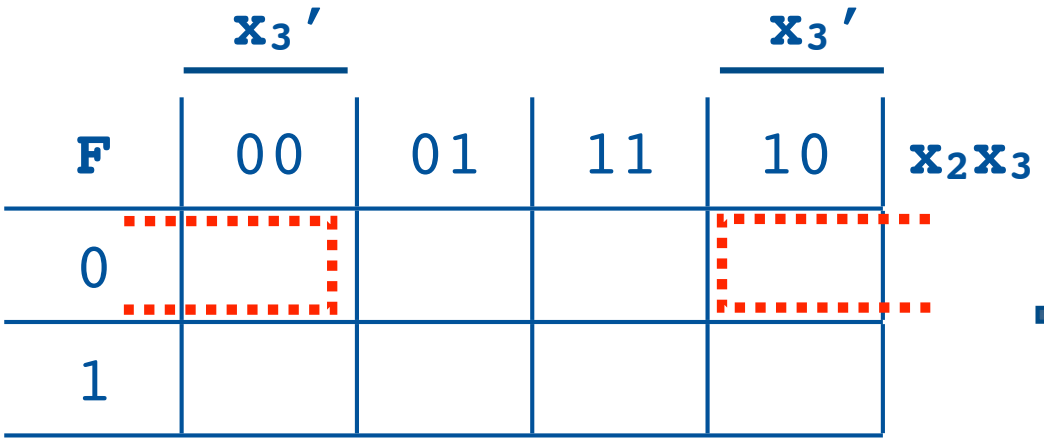
→ $x_2'x_3'$, $x_2'x_3$, x_2x_3, x_2x_3',...

F	00	01	11	10	x_2x_3
0					
1					
x_1					

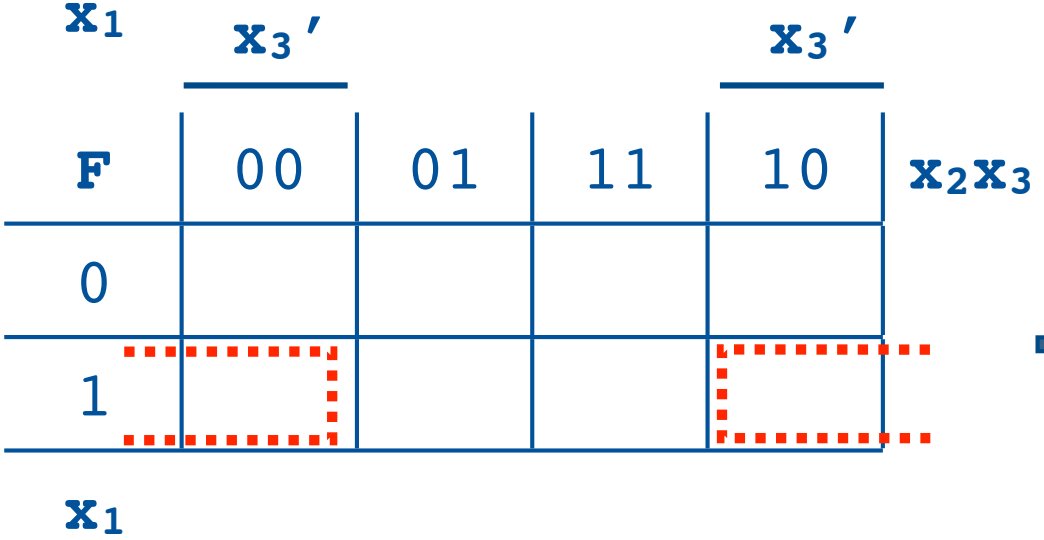
→ x_1x_2', $x_1'x_3$, x_1x_2 ,...

Tables de Karnaugh

... mais sont aussi des sous-cubes de taille 2 pour une K-Map à trois variables



$x_1'x_3'$



x_1x_3'

Tables de Karnaugh

Sous-cubes de taille 4 pour une K-Map à trois variables

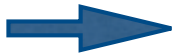
F	00	01	11	10	x_2x_3
0					
1					

x_1

x_1

F	00	01	11	10	x_2x_3
0					
1					

x_1



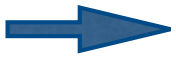
x_1'

Tables de Karnaugh

Sous-cubes de taille 4 pour une K-Map à trois variables

F	00	01	11	10	x_2x_3
0					
1					

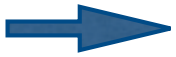
x_1



x_2', x_2

F	00	01	11	10	x_2x_3
0					
1					

x_1



x_3, x_3'

Tables de Karnaugh

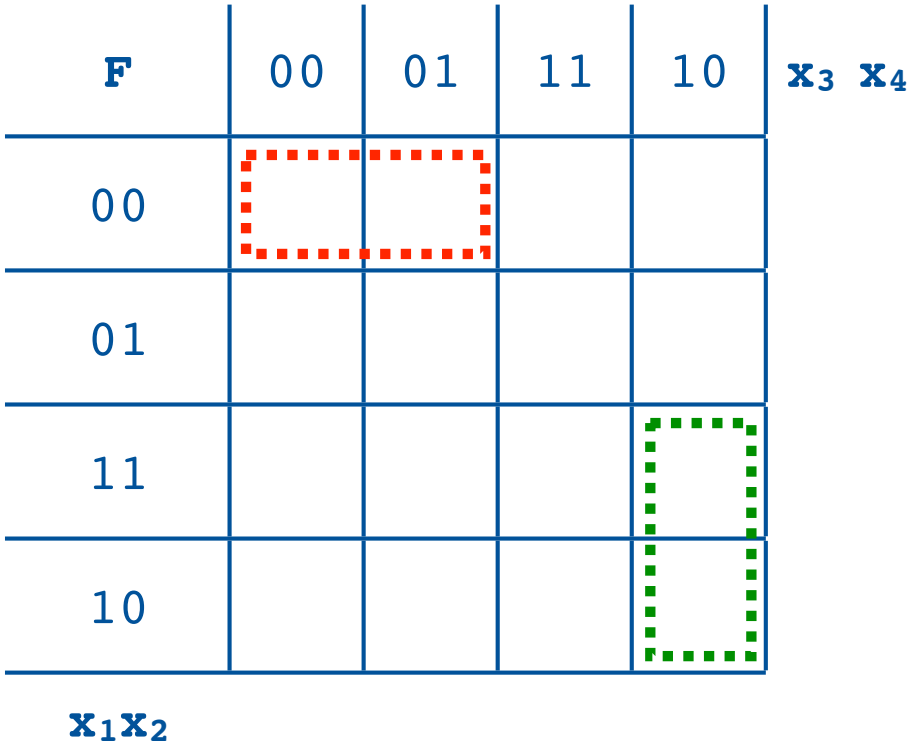
Remarque

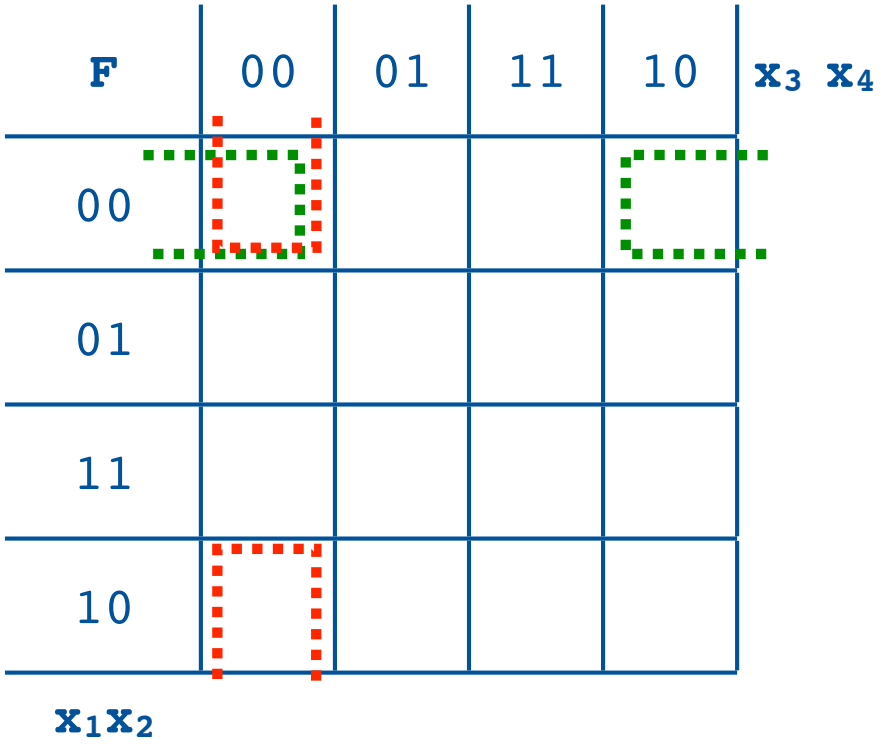
Pour une K-Map à 3 variables, un sous-cube d'une certaine taille implique le terme d'un certain nombre de variables:

1. *n*-cube de taille 1 - **trois** variables (c'est un minterme)
2. *n*-cube de taille 2 - **deux** variables (on a pu simplifier 1 var.)
3. *n*-cube de taille 4 - **une** variable (on a pu simplifier 2 var.)
4. *n*-cube de taille 8 - la **constante** (1)

Tables de Karnaugh

Sous-cubes 2 pour une K-Map à quatre variables



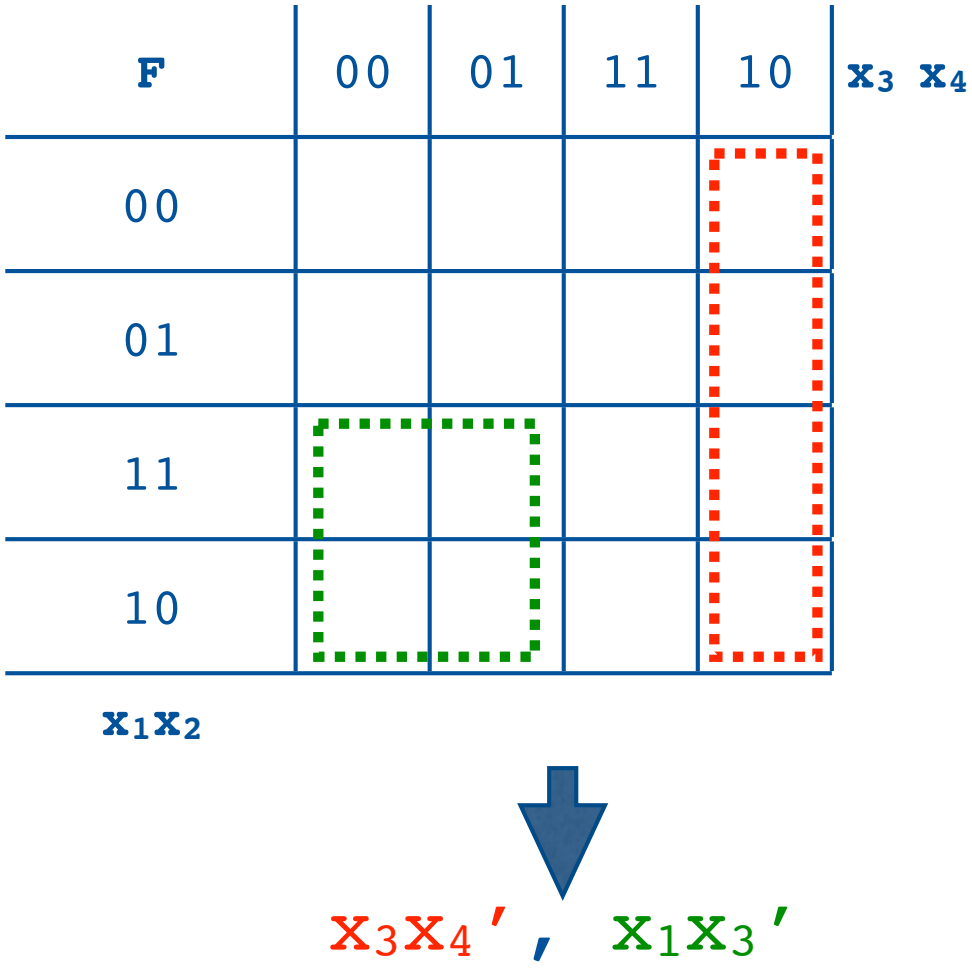


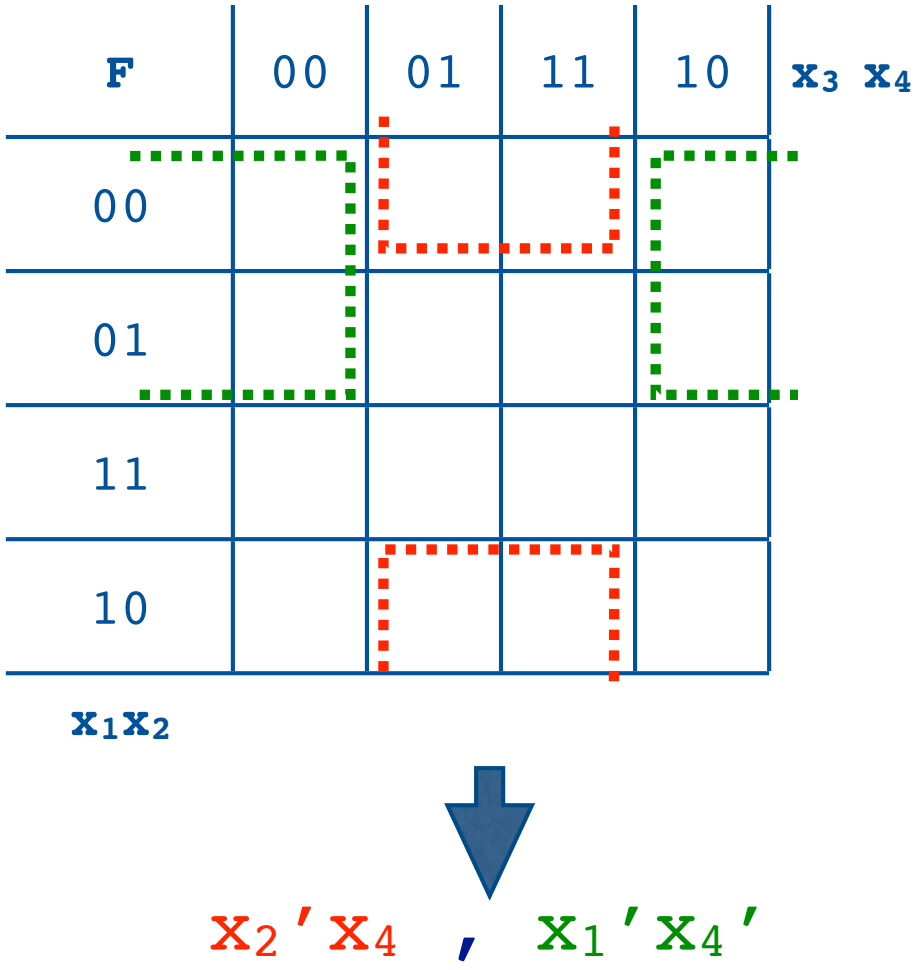
$x_1'x_2'x_3'$, $x_1x_3x_4'$

$x_2'x_3'x_4'$, $x_1'x_2'x_4'$

Tables de Karnaugh

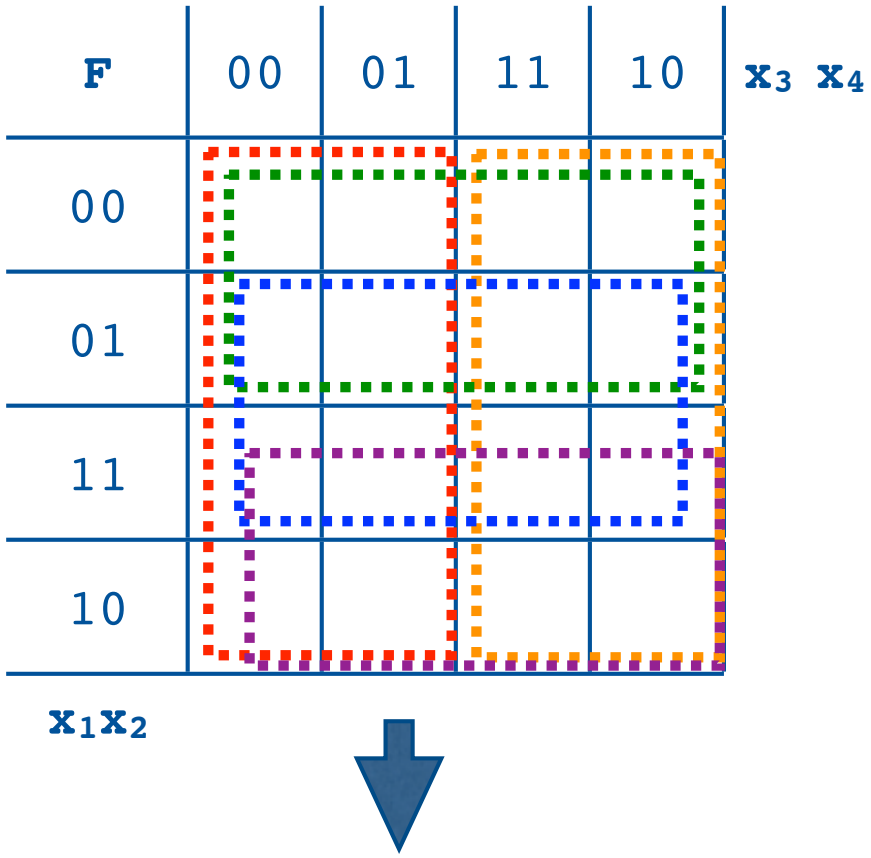
Sous-cubes 4 pour une K-Map à quatre variables





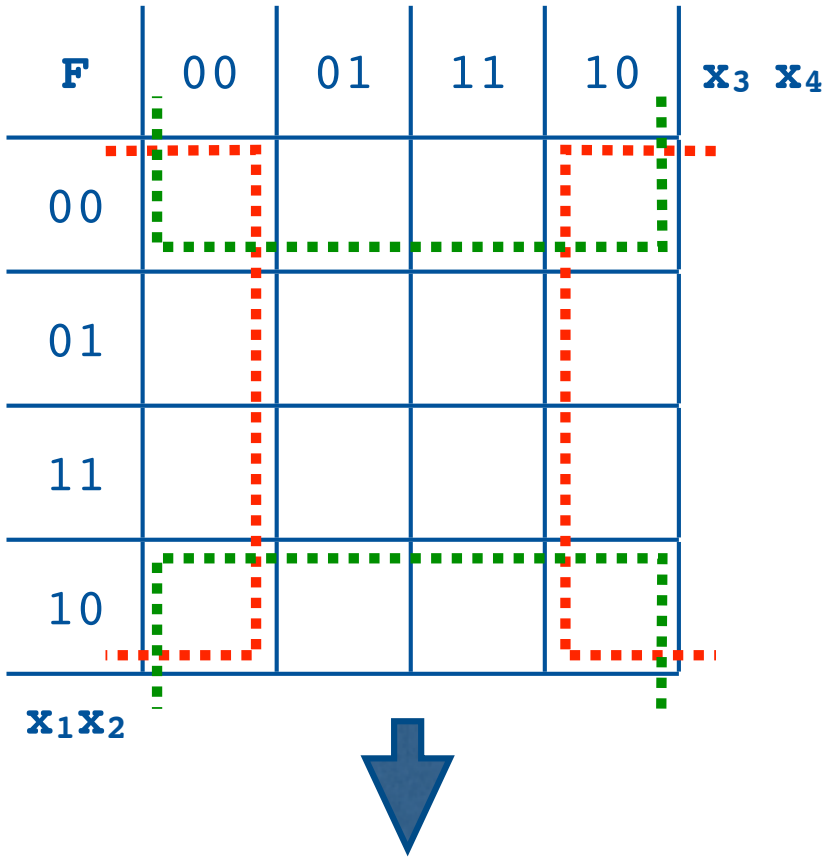
Tables de Karnaugh

Sous-cubes 8 pour une K-Map à quatre variables



x_3', x_3

x_1', x_2, x_1



x_4', x_2'

Tables de Karnaugh

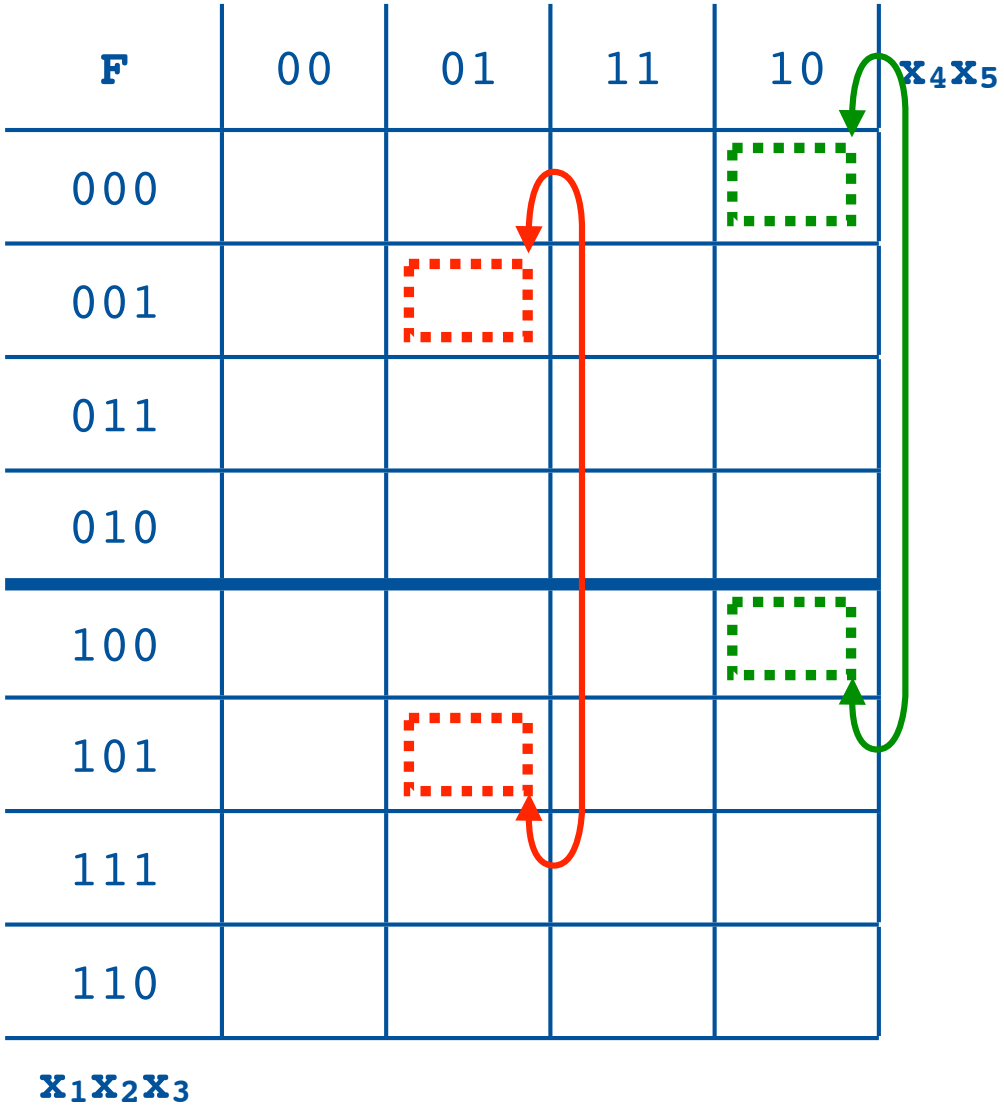
Remarque

Pour une K-Map à 4 variables un:

1. *n*-cube de taille 1 — **quatre** variables (c'est un minterme)
2. *n*-cube de taille 2 — **trois** variables (on a pu simplifier 1 var.)
3. *n*-cube de taille 4 — **deux** variables (on a pu simplifier 2 var.)
4. *n*-cube de taille 8 — **une** variable (on a pu simplifier 3 var.)
5. *n*-cube de taille 16 — la constante (1)

Tables de Karnaugh

Sous-cubes 2 pour une K-Map à cinq variables

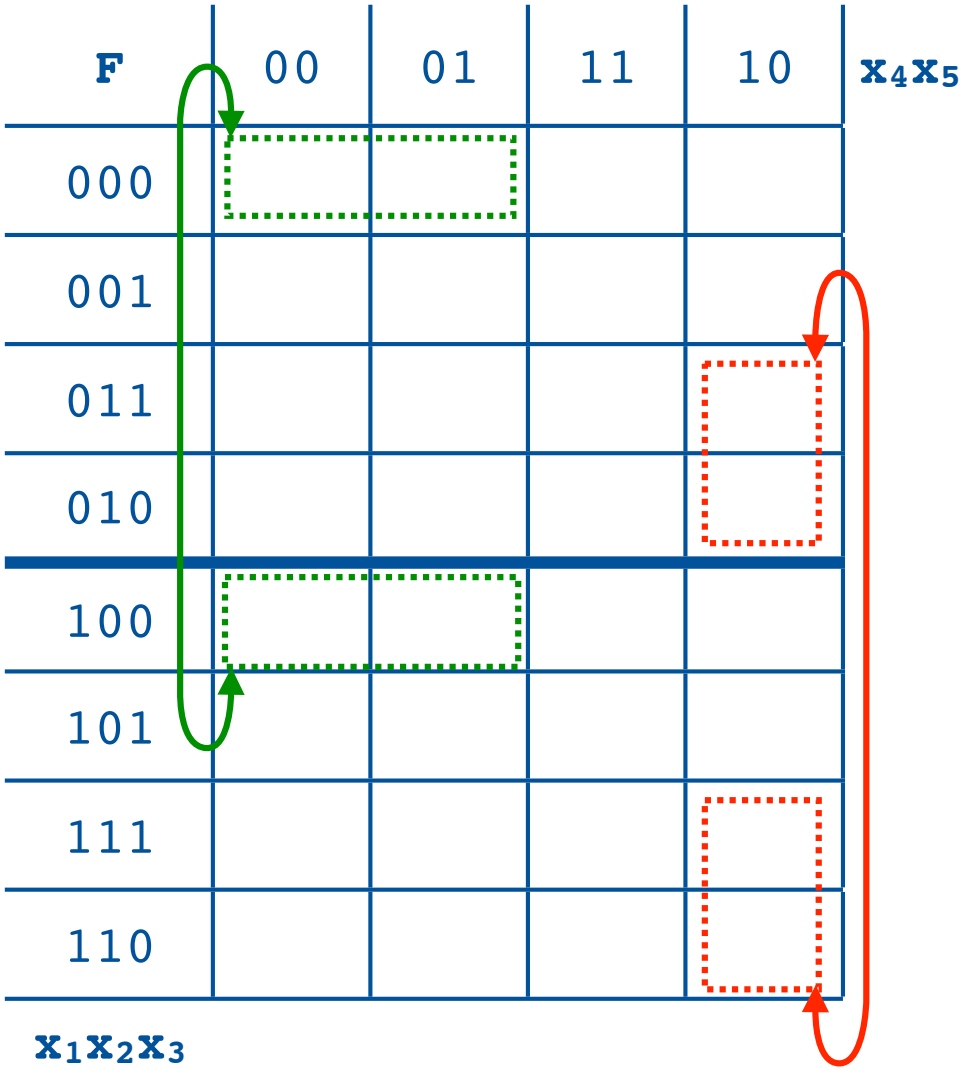


$x_2'x_3x_4'x_5$,

$x_2'x_3'x_4x_5'$

Tables de Karnaugh

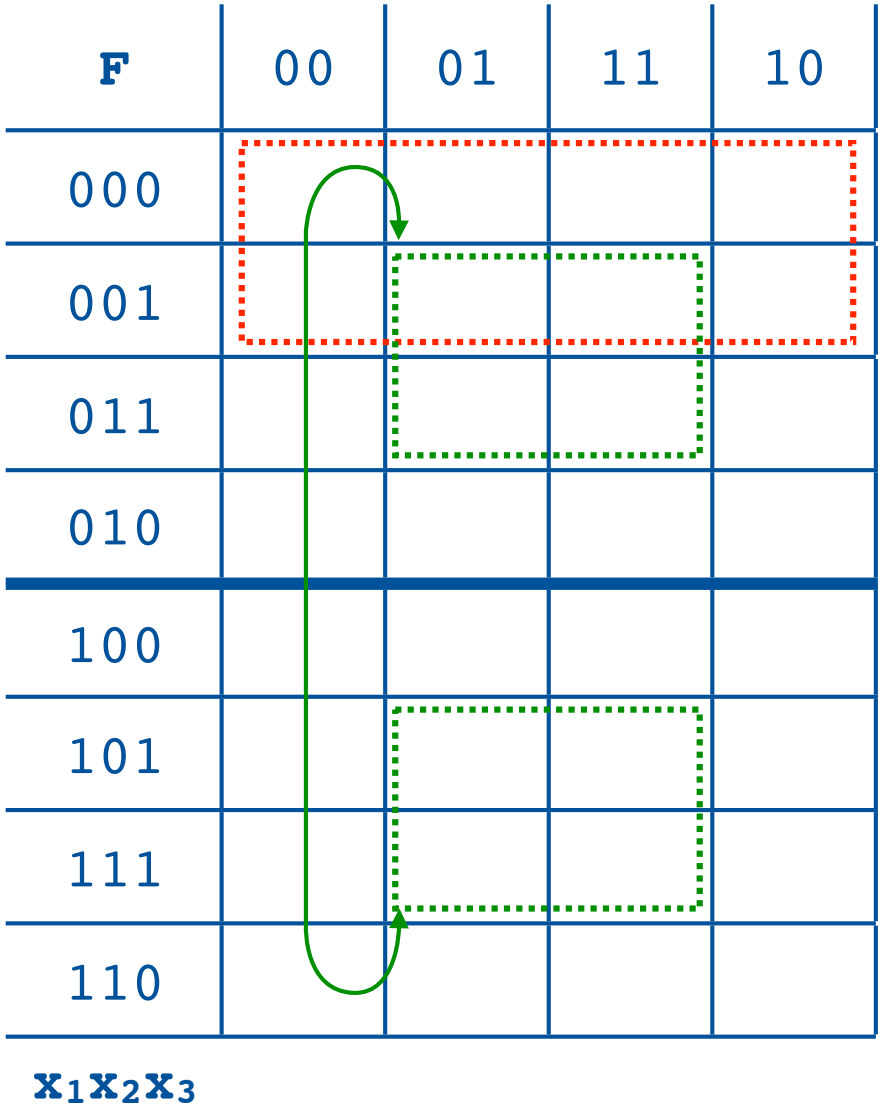
Sous-cubes 4 pour une K-Map à cinq variables



$x_2'x_3'x_4'$, $x_2x_4x_5'$

Tables de Karnaugh

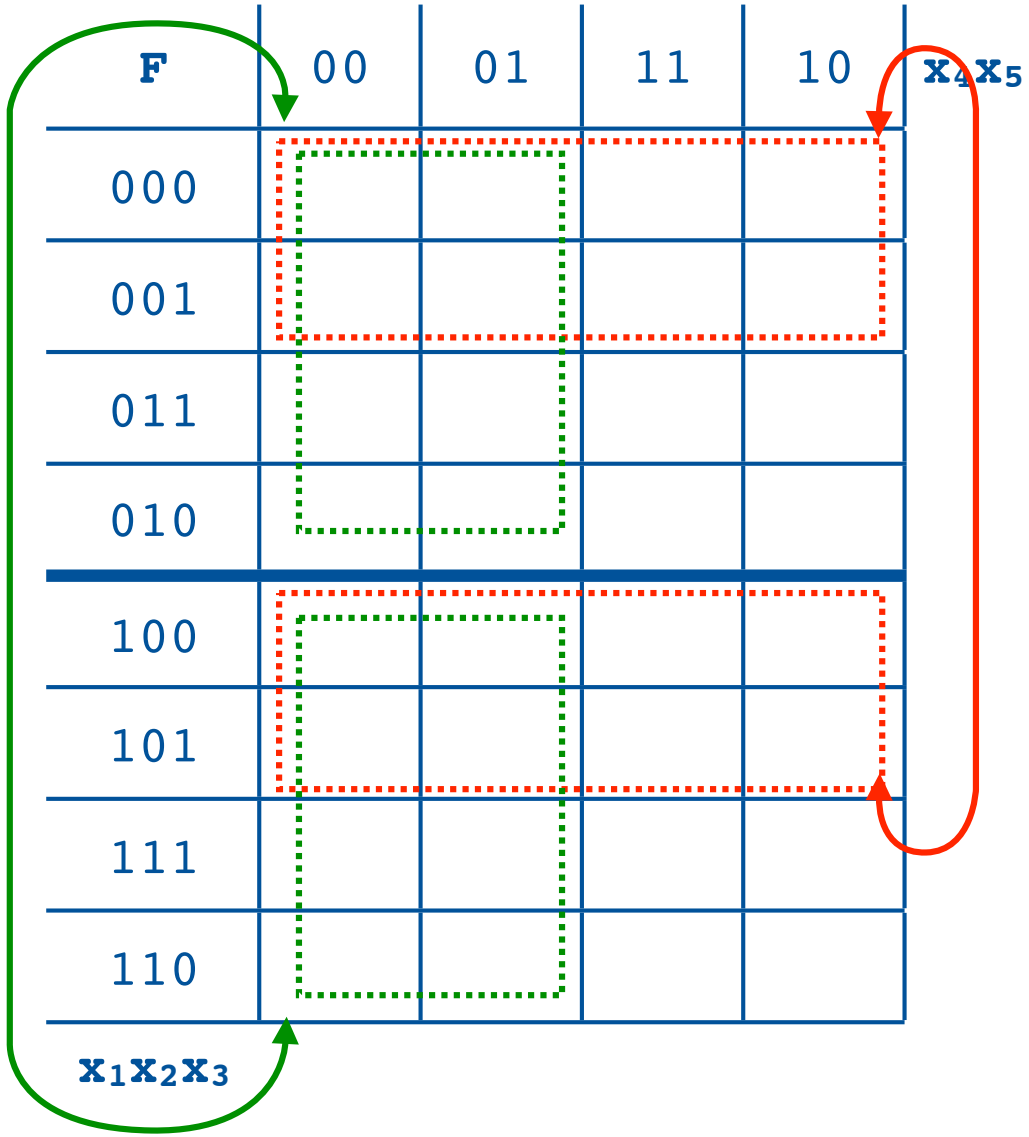
Sous-cubes 8 pour une K-Map à cinq variables



$x_1'x_2'$, x_3x_4'

Tables de Karnaugh

Sous-cubes 16 pour une K-Map à cinq variables



x_2', x_4'

Tables de Karnaugh

Remarque

Pour une K-Map à **5** variables un sous-cube :

1. *n*-cube de taille 1 - **cinq** variables (c'est un minterme)
2. *n*-cube de taille 2 - **quatre** variables (on a pu simplifier 1 var.)
3. *n*-cube de taille 4 - **trois** variable (on a pu simplifier 2 var.)
4. *n*-cube de taille 8 - **deux** variable (on a pu simplifier 3 var.)
5. *n*-cube de taille 16 - **une** variable (on a pu simplifier 4 var.)
6. *n*-cube de taille 32 - la constante (1)

Simplification des fonctions

Sous-cubes particuliers

- **Implicant premier**
  Un implicant premier est défini comme un sous-cube qui n'est pas défini au sein d'aucun autre sous-cube.

- **Implicant premier essentiel**
  Un implicant premier essentiel est un implicant premier seul à couvrir au moins un '1' de la fonction logique F.

- Les implicant premiers peuvent être trouver graphiquement dans une K-Map...

Simplification des fonctions

Fonction logique simplifiée est représentée comme une somme:

- de **tous** les **implicants premiers essentiels** et
- des **certains implicants premiers** de façon à couvrir **tous les '1'** d'une K-Map.

Dans une expression sous forme canonique non-standard, les monômes sont des implicants premiers $(m_i=1)$:

$$F = m_1 + m_2 + \ldots + m_k$$

Simplification des fonctions

Méthode de simplification (algorithme)

1. Rechercher **tous les implicants premiers** de la fonction logique écrite sous forme d'une K-Map

2. Parmi tous les implicants premiers de la fonction, **isoler** les **implicants premiers essentiels** (comme ils sont les seuls à couvrir certains '1', on est obligé de les prendre)

3. **Couvrir** le reste des '1' avec moins d'implicants premiers possible (couverture minimale i.e. simplification)

Simplification des fonctions

Simplification à l'aide des K-Maps

Remarque liée à la réalisation physique

Consiste à trouver le plus petit nombre des plus grandes sous-cubes permettant de couvrir tous les '1' dans une K-map.

Dans cette phrase:

... le plus petit nombre des sous-cubes ...

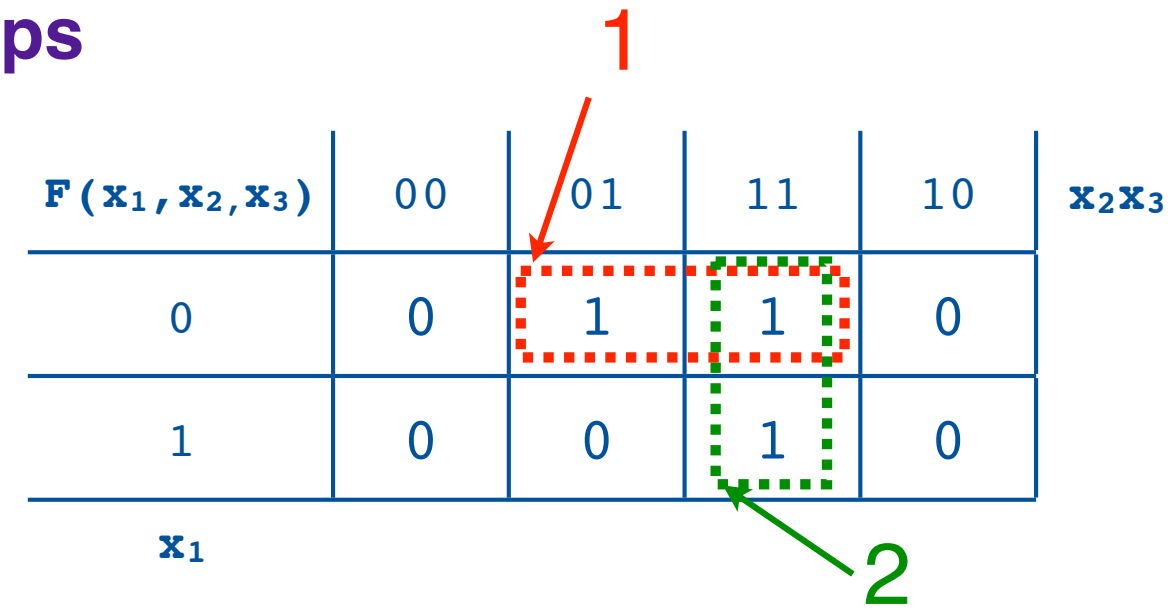
Pour une SdP ceci influence la taille de la porte OU et le nombre de portes ET.

... les plus grandes sous-cubes ...

Influence la taille de chaque porte ET

Simplification des fonctions

K-Maps



$F(x_1,x_2,x_3)$	00	01	11	10	x_2x_3
0	0	1	1	0	
1	0	0	1	0	
x_1					

Sous-cubes : 1 et 2

Implicants premiers : $x_1'x_3$, x_2x_3

Implicants premiers essentiels : $x_1'x_3$, x_2x_3

Fonction simplifiée: $F = x_1'x_3 + x_2x_3$

Simplification des fonctions

a) Remarque sur la superposition des sous-cubes

$F(x_1,x_2,x_3)$	00	01	11	10	x_2x_3
0	0	1	1	0	
1	0	0	1	0	
x_1					

On se pose la question sur la **superposition** des deux blocs vert et rouge (ils ont la combinaison 011 de variables en commun)?

On peut superposer les blocs car x+x=x. (C'est même souhaitable!)

Si l'on ne superpose pas alors: $F = x_1'x_3 + x_1x_2x_3$

Avec superposition nous avons simplifié l'expression de F car:

$F = x_1'x_3 + x_2x_3$

Simplification des fonctions

b) Remarque sur la superposition des sous-cubes

$F(x_1,x_2,x_3)$	00	01	11	10	x_2x_3
0	0	1	1	0	
1	0	0	1	1	
x_1					

On se pose la question si le sous cube est nécessaire?

NON, car avec les sous-cubes rouge et vert nous avons couvert tous les '1' de la fonction.

Attention: il y aura des cas où nous devrions rajouter ces termes... mais ce n'est pas pour tout de suite.

Exemples à 3 variables

F	00	01	11	10	x_2x_3
0	0	1	1	0	
1	0	1	0	0	
x_1					

→ $F = x_1'x_3 + x_2'x_3$

F	00	01	11	10	x_2x_3
0	1			1	
1		1			
x_1					

→ $F = x_1'x_3' + x_1x_2'x_3$

F	00	01	11	10	x_2x_3
0	1	1		1	
1	1		1	1	
x_1					

→ $F = x_3' + x_1'x_2' + x_1x_2$

Exemples à 4 variables

F	00	01	11	10	x_3x_4
00	1				
01	1			1	
11	1		1	1	
10	1				

x_1x_2

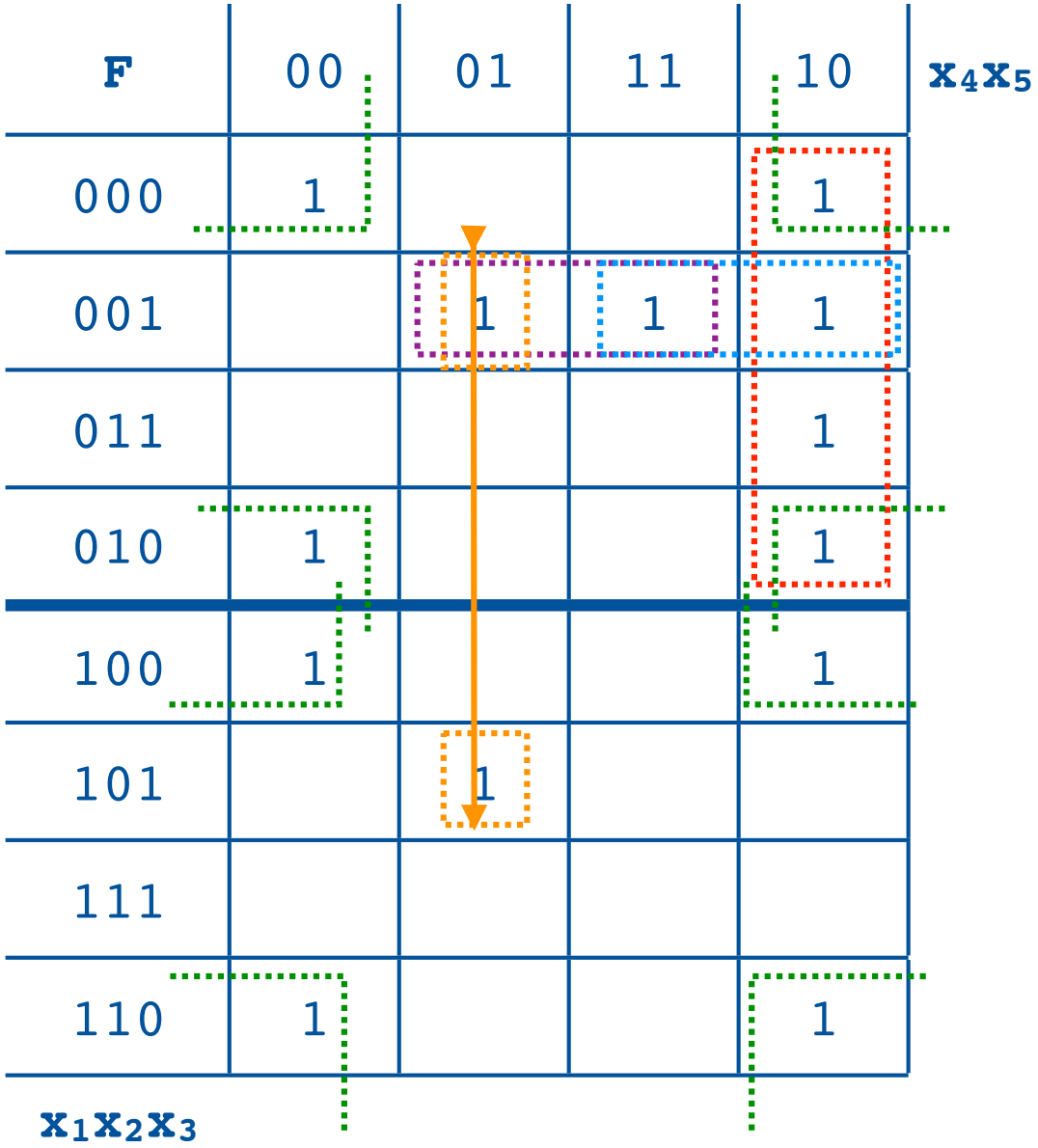
$$F = x_1'x_2' + x_1x_2x_3 + x_2x_4'$$

F	00	01	11	10	x_3x_4
00	1	1		1	
01					
11					
10	1	1		1	

x_1x_2

$$F = x_2'x_3' + x_2'x_4'$$

Exemple 1 à 5 variables



$F = x_3'x_5'$

$+ \; x_1'x_4\,x_5'$

$+ \; x_2'x_3\,x_4'x_5$

$+ \; x_1'x_2'x_3x_5$

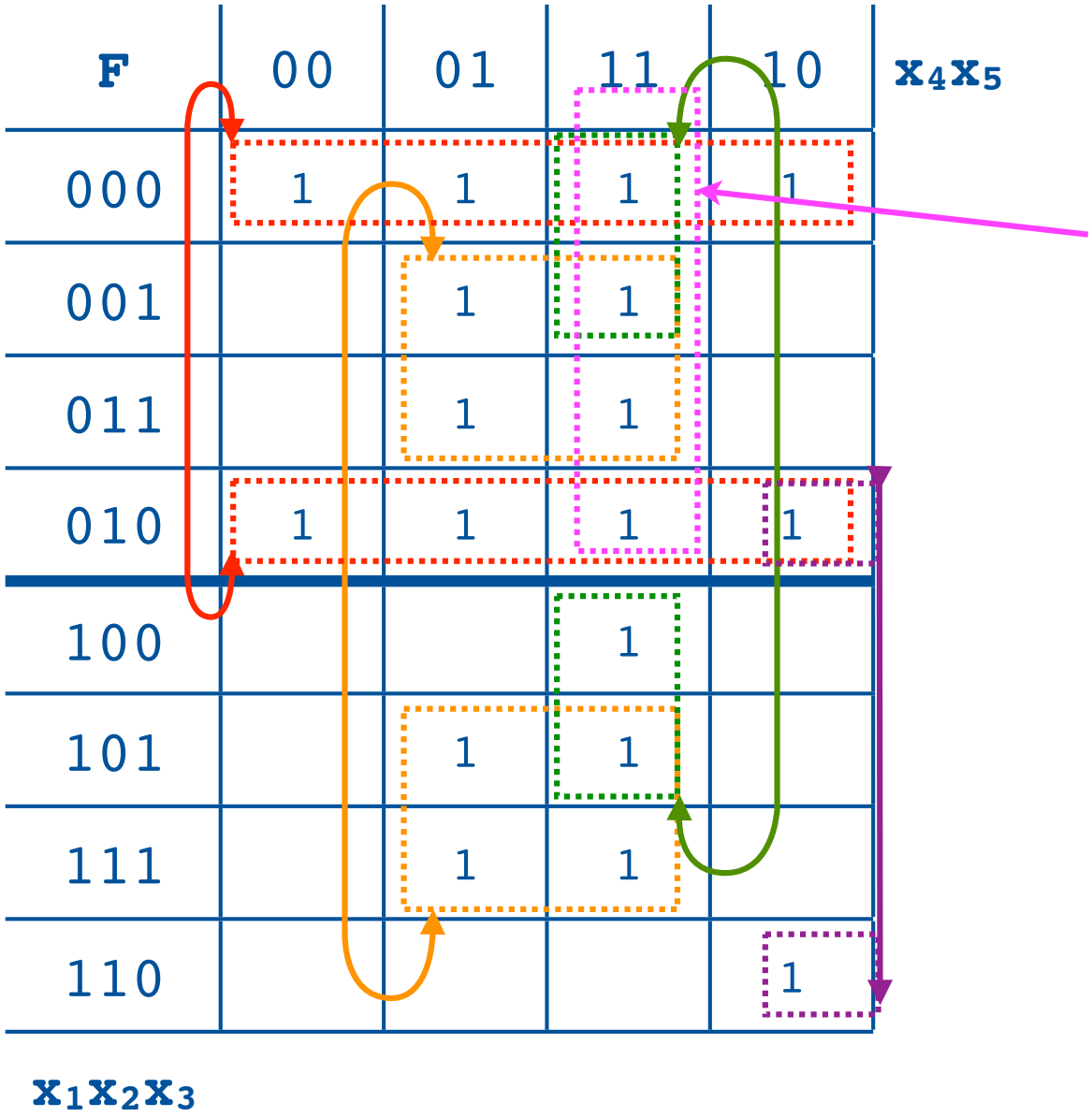
Variante pour :

$x_1'x_2'x_3x_5$

$x_1'x_2'x_3x_4$

Résultat est le même: nous avons plusieurs solutions en partant une expression.

Exemple 2 à 5 variables



Est aussi un implicant, mais on ne l'écrit pas car tous les 1 sont déjà couverts

Fonction simplifiée:

$F = x_2'x_4x_5 + x_1'x_3' + x_3\,x_5 + x_2x_3'x_4x_5'$

K-Maps: Fautes “graves”

Faute grave (Définition)

A l’examen entraîne 0 points pour l’exercice...

Fautes graves pour les K-Maps

- Ne pas respecter l'adjacence dans la création de la K-Map (lignes et colonnes ne peuvent différer que d’un seul bit)
- Faire des groupements des mintermes non-adjacents (colonne 1 et 3 p.e.)
- Faire des groupements de mintermes par 3, 5, 6, 7, ... (construire des sous-cubes qui ne sont pas x^2)

Fonctions non-complètement spécifiées

Minterm indifférent

La fonction peut être non-complètement spécifiée:

Pour certaines combinaisons des valeurs des variables, on ne se préoccupe pas de résultat de la fonction.

a. Soit parce que nous nous intéressons pas à la sortie pour cette (ou ces) combinaison(s) de variables.
   On parle alors des ***don't care***.

b. Soit parce que cette (ou ces) combinaison(s) de variables n'arrivera jamais!
   On parle alors des ***don't happen***.

Fonctions non-complètement spécifiées

- Dans une K-map (ou dans une table de vérité) nous allons marquer les ***don't cares*** ou les ***don't happen***: –
- A cet endroit la fonction logique peut prendre la valeur de `0` ou de `1`.
- Utilisation: si la fonction logique peut prendre `0` ou `1`, nous pouvons donc choisir l'une de ces deux valeurs.

A l'endroit où la fonction est non complètement spécifiée : on va choisir la valeur permettant de simplifier au mieux l'expression finale.

Fonctions non-complètement spécifiées

Différence entre les *Don't cares* et les *Don't happen*

- Interprétation:
  - ***Don't care*** - ce n'est pas grave ce que le système fera à la sortie
  - ***Don't happen*** - ce n'est pas grave ce que le système fera à la sortie, puisque **cette combinaison à l'entrée ne se présentera jamais au système**!
- Différence pour la simplification: AUCUNE!
- La différence réside dans comment le circuit en amont (ou les entrées du système) se présentent et qu'est-ce que nous souhaitons faire à la sortie

Fonctions non-complètement spécifiées

Exemple 1: une fonction à trois variables

F	00	01	11	10	x_2x_3
0	0	1	-	0	
1	0	1	1	0	
x_1					

Don't care

F	00	01	11	10	x_2x_3
0	0	1	~~0~~	0	
1	0	1	1	0	
x_1					

$F = x_2'x_3 + x_1x_3$

F	00	01	11	10	x_2x_3
0	0	1	~~1~~	0	
1	0	1	1	0	
x_1					

$F = x_3$

Fonctions non-complètement spécifiées

Exemple 2: une fonction à trois variables

F	00	01	11	10	x_2x_3
0	0	0	–	0	
1	0	1	1	0	

x_1

F	00	01	11	10	x_2x_3
0	0	0	~~0~~	0	
1	0	1	1	0	

x_1

$F = x_1x_3$

F	00	01	11	10	x_2x_3
0	0	0	~~1~~	0	
1	0	1	1	0	

x_1

$F = x_1x_3 + x_2x_3$

Fonctions non-complètement spécifiées

- Les ***don't care*** ou les ***don't happen*** nous aident à simplifier mieux les fonctions logiques
- Mais attention ce n'est pas en mettant systématiquement un *don't care* à 1 que l'on simplifie au mieux
  (cf. l'exemple du transparent précédent)
- L'ensemble B devient un ensemble à trois éléments `{0,1,-}`
- Le symbole **-** n'est pas uniquement utilisé pour designer la valeur de la sortie, il est également utilisé pour désigner **les entrées**.

Fonctions non-complètement spécifiées

- Dans une K-Map pour un impliquant on dit:
  La fonction logique vaut 1 lorsque xxx et lorsque xxx ...
  quelque soit xxx ...
- Le **... quelque soit ...** s'écrit à l'aide de symbole -
- Exemple:
  Mintermes: $x_1'x_2x_3'$; $x_1x_2x_3'$

  La fonction logique vaut 1 quelque soit la valeur de x_1...

  On écrit alors: $-x_2x_3'$

Fonctions non-complètement spécifiées

Exemple 1: fonction à quatre variables

F	00	01	11	10	x_3x_4
00	1	0	0	–	
01	–	0	–	1	
11	1	0	1	1	
10	1	0	0	0	
x_1x_2					

Liste des implicants:

x_2x_3

$x_3'x_4'$

$x_1'x_4'$

Fonction optimisé:
(on ne prends pas $x_1'x_4'$ car tous les '1' sont déjà couverts)

$$F = x_3'x_4' + x_2x_3$$

Fonctions non-complètement spécifiées

Exemple 2: fonction à quatre variables

F	00	01	11	10	x_3x_4
00	1	0	0	–	
01	–	0	0	1	
11	0	0	0	1	
10	1	0	0	–	
x_1x_2					

Liste des implicants:

$x_2'x_4'$

x_3x_4'

$x_1'x_4'$

Fonction optimisé:

$F = x_2'x_4' + x_3x_4'$