Simplification par tables de Karnaugh

Exercices Conception numérique

1 | KAR - Tables de Karnaugh

1.1 Représentation de monômes

Représenter, dans une table de Karnaugh à 4 variables, les monômes suivants:

$y_1 = \overline{b}a$ (1) $y_2 = \overline{d}\overline{a}$ (2) $y_3 = \overline{d}cb$ (3) $y_4 = dba$ (4) $y_5 = \overline{c}\overline{b}\overline{a}$ (5) $y_6 = dcb\overline{a}$ (6)

kar/karnaugh-01

1.2 Monômes

Donner l'expression algébrique des monômes représentés dans les tables de Karnaugh suivantes.

y_1

	C		D
1	1	0	0
1	1	0	0
0	0	0	0
0	0	0	0

A
B

y_3

	C		D
0	1	0	0
0	1	0	0
0	0	0	0
0	0	0	0

A
B

y_2

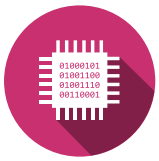
	C		D
1	0	0	1
0	0	0	0
0	0	0	0
1	0	0	1

A
B

y_4

	C		D
0	1	0	0
0	0	0	0
0	0	0	0
0	1	0	0

A
B



kar/karnaugh-02

1.3 Représentation de polynômes

Représenter, dans une table de Karnaugh à 4 variables, les polynômes suivants:

$$y_1 = \overline{b} + ac \quad (7)$$

$$y_2 = \overline{d} + \overline{a} + bc \quad (8)$$

$$y_3 = \overline{d}cb + d\overline{c}\overline{b} \quad (9)$$

$$y_4 = db + ab \quad (10)$$

$$y_5 = \overline{c}\overline{b}\overline{a} + \overline{c}\overline{b} \quad (11)$$

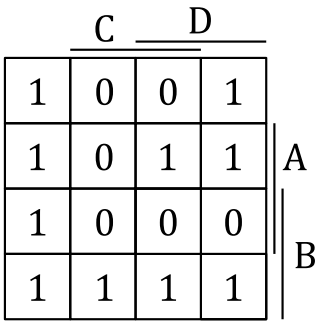
$$y_6 = dcb\overline{a} + \overline{d}cb\overline{a} \quad (12)$$

kar/karnaugh-03

2 | KAR - Simplification sous forme de somme de produits

2.1 Table de Karnaugh à 4 variables

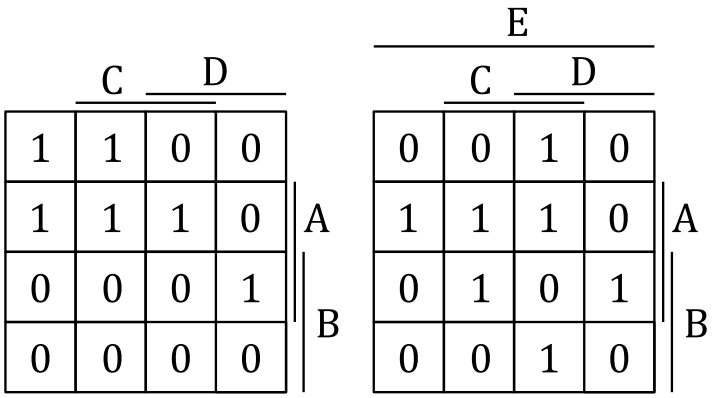
Déterminer la forme polynomiale minimale de la fonction représentée dans la table de Karnaugh de la figure suivante.



kar/productsum-01

2.2 Table de Karnaugh à 5 variables

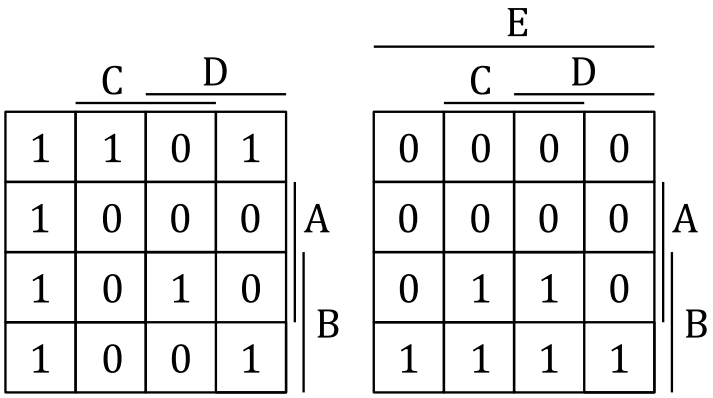
Déterminer la forme polynomiale minimale de la fonction représentée dans la table de Karnaugh de la figure suivante.



kar/productsum-02

2.3 Table de Karnaugh à 5 variables

Déterminer la forme polynomiale minimale de la fonction représentée dans la table de Karnaugh de la figure suivante.

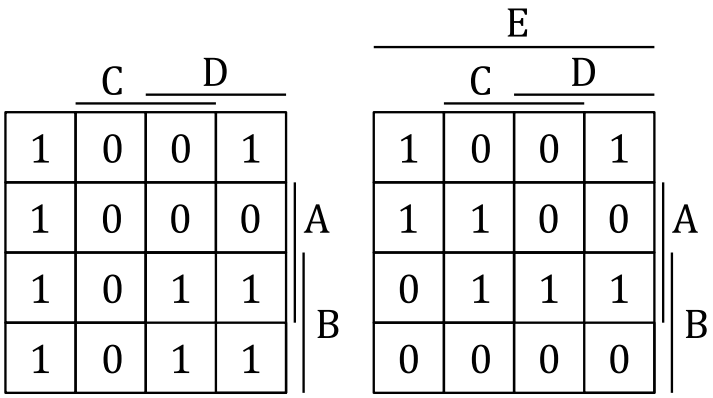


1	1	0	1		0	0	0	0
1	0	0	0	A	0	0	0	0	A
1	0	1	0		0	1	1	0	
1	0	0	1	B	1	1	1	1	B

kar/productsum-03

2.4 Table de Karnaugh à 5 variables

Déterminer la forme polynomiale minimale de la fonction représentée dans la table de Karnaugh de la figure suivante.

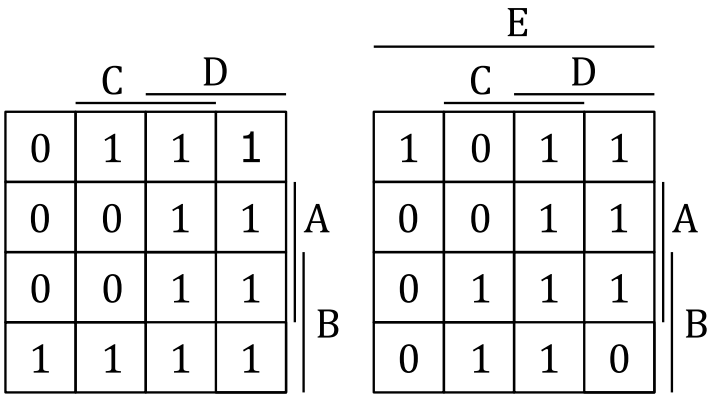


1	0	0	1		1	0	0	1
1	0	0	0	A	1	1	0	0	A
1	0	1	1		0	1	1	1	
1	0	1	1	B	0	0	0	0	B

kar/productsum-04

2.5 Table de Karnaugh à 5 variables

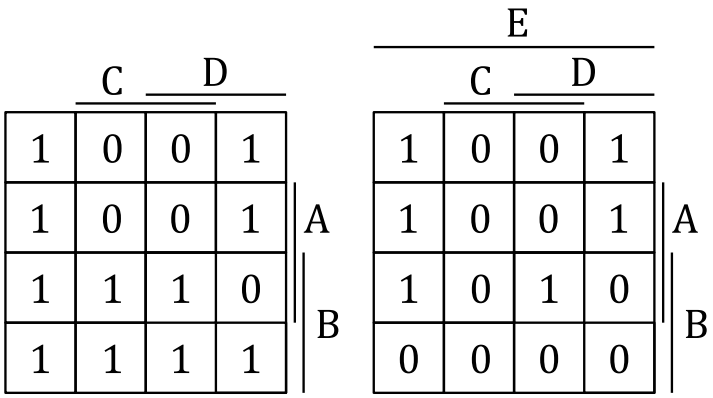
Déterminer la forme polynomiale minimale de la fonction représentée dans la table de Karnaugh de la figure suivante.



kar/productsum-05

2.6 Table de Karnaugh à 5 variables

Déterminer la forme polynomiale minimale de la fonction représentée dans la table de Karnaugh de la figure suivante. s



kar/productsum-06

2.7 Forme polynomiale minimale

Déterminer la forme polynomiale minimale de la fonction

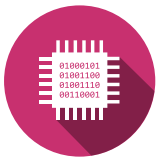
$$y = \overline{x_3\ x_2\ \overline{x_0} + \overline{x_2}\ \overline{x_1}\ \overline{x_0} + x_2\ \overline{x_1}\ x_0} \tag{13}$$

kar/productsum-07

2.8 Fonction inverse

Soit la fonction donnée par l'équation

$$y = \overline{a}\ \overline{c} + a\ \overline{b}\ \overline{e} + b\ \overline{c}\ \overline{d} + \overline{a}\ \overline{b}\ e \tag{14}$$



Donner l'expression polynomiale minimalisée de la fonction inverse $\overline{y}$.

kar/productsum-08

2.9 Forme polynomiale minimale

Pour la fonction décrite par la table de vérité ci-contre, déterminer laquelle des expressions polynomiales peut avoir le moins de termes: celle de la fonction Y ou celle de la fonction inverse $\overline{Y}$.

D	C	B	A	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

kar/productsum-09

2.10 Fonction de 5 variables

Déterminer l'expression polynomiale minimale de la fonction majorité à 5 entrées:

- la fonction retourne un **`'0'`** lorsque plus de la moitié des entrées sont à **`'0'`**,
- la fonction retourne un **`'1'`** lorsque plus de la moitié des entrées sont à **`'1'`**.

kar/productsum-10

2.11 Fonction incomplètement définie

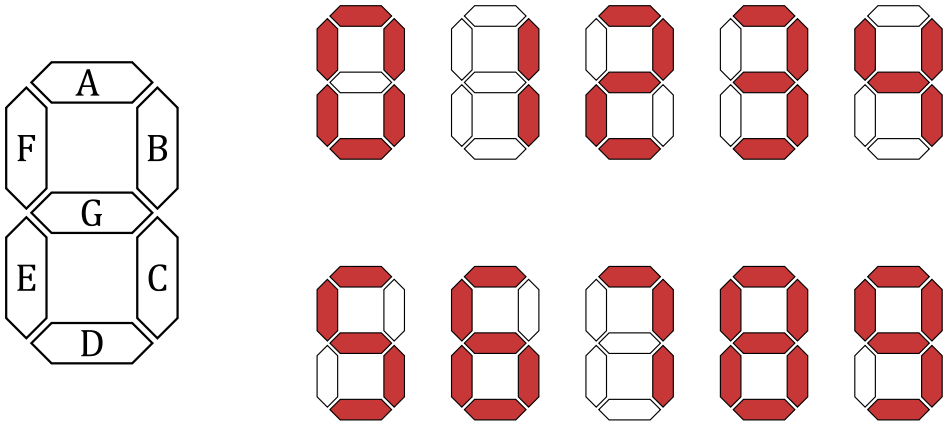
Déterminer l'expression polynomiale minimale de la fonction majorité à 4 entrées:

- la fonction retourne un **`'0'`** lorsque plus de la moitié des entrées sont à **`'0'`**,
- la fonction retourne un **`'1'`** lorsque plus de la moitié des entrées sont à **`'1'`**,
- lorsque le nombre d'entrées à **`'0'`** est identique à celui des entrées à **`'1'`**, le résultat peut prendre n'importe quelle valeur.

kar/productsum-11

2.12 Fonction incomplètement définie

Un affichage à 7 segments est utilisé pour représenter un chiffre décimal.



Concevoir un circuit logique combinatoire qui transforme un nombre binaire codé sur 4 bits dans les 7 signaux de commande pour l'allumage des 7 segments.
Dans ce système, le nombre binaire d'entrée ne peut prendre que les valeurs correspondant aux chiffres décimaux de 0_d à 9_d.

kar/productsum-12

3 KAR - Simplification de fonctions OU-Exclusif

3.1 Représentation de fonctions OU-exclusif

Représenter, dans une table de Karnaugh à 4 variables, les fonctions suivantes:

$y_1 = a \oplus b$
$y_2 = a \oplus b \oplus c$
$y_3 = a \oplus b \oplus c \oplus d$
$y_4 = \overline{a \oplus b}$
$y_5 = \overline{a} \oplus b$
$y_6 = a \oplus \overline{b}$
$y_7 = d\overline{b} \oplus \overline{d}b$
$y_8 = d \oplus b$
$y_9 = \overline{b} \oplus \overline{d}$

kar/xor-01

3.2 Forme polynomiale minimale

Déterminer la forme polynomiale minimale de la fonction

$$y = x_1 \oplus \overline{x_1}x_0 \oplus x_2\overline{x_1} \oplus x_3x_2\overline{x_0} \tag{15}$$

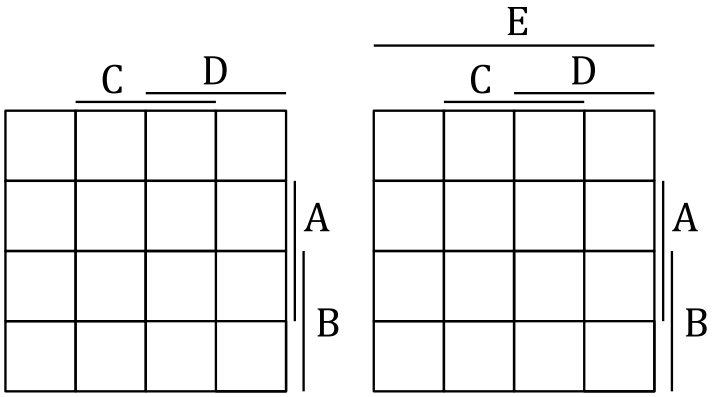
kar/xor-02

3.3 Forme polynomiale minimale

Déterminer la forme polynomiale minimale de la fonction

$$y = e \oplus \overline{d} \oplus dc \oplus d\overline{b} \oplus \overline{c}a \tag{16}$$

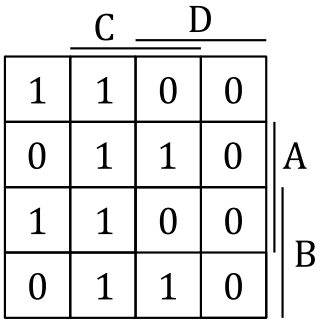
Si vous travaillez avec une table de Karnaugh, utilisez le même arrangement des variables que celui de la table de la figure suivante.



kar/xor-03

3.4 Forme ou-exclusif de produits

Ecrire sous forme de OU-exclusif de produits l'équation de la fonction représentée dans la table de Karnaugh de la figure suivante.



kar/xor-04

3.5 Forme ou-exclusif de produits

Ecrire la fonction combinatoire suivante sous forme de OU-exclusif de produits.

$$y = x_1\ x_0 + \overline{x_2}\ x_1 + \overline{x_2}\ x_0 + \overline{x_3}\ x_2\ \overline{x_1} \tag{17}$$

kar/xor-05

3.6 Additionneur

Dessiner le schéma d'un additionneur 2 bits à l'aide de portes ET et de portes OU-exclusif.
L'additionneur réalise la somme de 2 nombres codés chacun sur 2 bits. Il fournit un résultat sur 3 bits.
En considérant que toutes les portes ont un retard identique, veiller à minimiser en priorité le délai entre les entrées et les sorties. Minimiser aussi le nombre de portes logiques.

kar/xor-06

4 KAR - Fonctions avec un nombre élevé d'entrées

4.1 Comparaison de nombres

Réaliser un circuit qui indique si un nombre binaire A est plus grand que le nombre binaire B.
Les nombres sont codés sur 8 bits. Ils correspondent à des entiers positifs.

kar/manyinputs-01

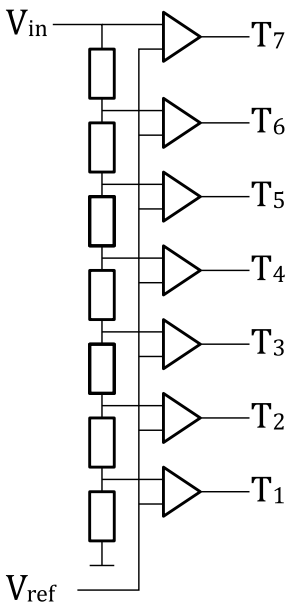
4.2 Additionneur binaire

Réaliser un circuit qui réalise la somme de deux nombres binaires codés sur 8 bits, avec un résultat sur 8 bits aussi
Veiller à minimiser le nombre de portes logiques utilisées.

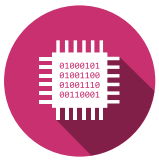
kar/manyinputs-02

4.3 Conversion de code thermomètre en code binaire

Un convertisseur analogique/numérique à 3 bits fonctionne comme suit: la tension d'entrée est divisée dans une chaîne de 7 résistances. Chaque point intermédiaire est comparé à une tension de référence. Le code binaire fourni est appelé code thermomètre.
Réaliser un circuit combinatoire qui transforme ce code thermomètre en code binaire.



kar/manyinputs-03



4.4 Transmission selon la priorité

Réaliser un circuit logique combinatoire qui, de tous ses signaux d'entrée, ne transmet que celui de priorité supérieure.
Le système comprend 8 entrées, I_1 à I_8, et délivre 8 sorties, O_1 à O_8. L'entrée I_8 a la priorité la plus élevée.
La table suivante donne quelques exemples de fonctionnement:

$I_1 ... I_8$	$O_1 ... O_8$
00000000	00000000
00010000	00010000
11000000	01000000
11010000	00010000
11010010	00000010

kar/manyinputs-04

4.5 Logique pour compteur sans retour à zéro

Pour la réalisation d'un compteur à 16 bits, concevoir un circuit combinatoire qui délivre la valeur suivante du compteur, y, en fonction de la valeur présente, x.
La fonction à réaliser est $y = x + 1$ tant que x n'est pas à sa valeur maximale: $x = \mathrm{FFFF}_h$ (tous les bits à 1). Dans ce dernier cas, le circuit délivre $y = \mathrm{FFFF}_h$, dans le but d'éviter un retour à zéro.

kar/manyinputs-05

4.6 Additionneur avec saturation

Proposer le schéma d'un aditionneur dont les entrées, ainsi que le résultat, sont représentés sur 16 bits et où le dépassement de capacité est traité par une saturation. Le calcul se fait avec des nombres non-signés.
La saturation se définit comme suit: si la somme des 2 nombres dépasse la valeur maximale $y = \mathrm{FFFF}_h$ (tous les bits à 1), le circuit délivre cette valeur maximale en sortie.

kar/manyinputs-06

4.7 Nombres en code BCD

Proposer un circuit qui réalise la somme de 2 nombres codés en BCD (Binary Coded Decimal).
Les 2 nombres sont codés sur 3 chiffres BCD, soit sur 12 bits. Le résultat est à fournir sur 4 chiffres BCD, soit sur 16 bits.

kar/manyinputs-07

4.8 Fonction majorité à 7 entrées

Donner le schéma à inverseurs, portes ET, OU et OU-exclusif du circuit qui détermine la majorité de 7 entrées.

kar/manyinputs-08

4.9 Unité arithmétique et logique

Dessiner le schéma d'une unité arithmétique et logique (Arithmetic and Logic Unit, ALU) d'un microprocesseur.
L'ALU fonctionne sur 8 bits. Les opérations possibles sont:

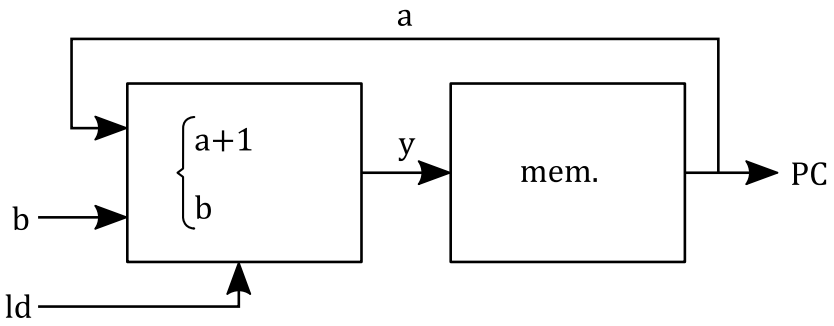
Command	Opération	
00	Addition	$y = a + b$
01	Soustraction	$y = a - b$
10	ET	$y = a$ AND b
11	OU	$y = a$ OR b

Les opérations logiques sont effectuées bit à bit, exemple: $y_i = a_i \text{AND} b_i, \forall i$.

kar/manyinputs-09

4.10 Logique pour compteur de programme

Pour le compteur de programme d'un microprocesseur (Program Counter, PC), on a besoin d'un bloc capable soit d'incrémenter le compteur (passage à la prochaine instruction) soit d'y charger une nouvelle valeur (saut).

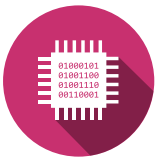


Concevoir un circuit combinatoire qui calcule

$$y = \begin{cases} \text{ld} = 0 \Rightarrow a + 1 \\ \text{ld} = 1 \Rightarrow b \end{cases} \tag{18}$$

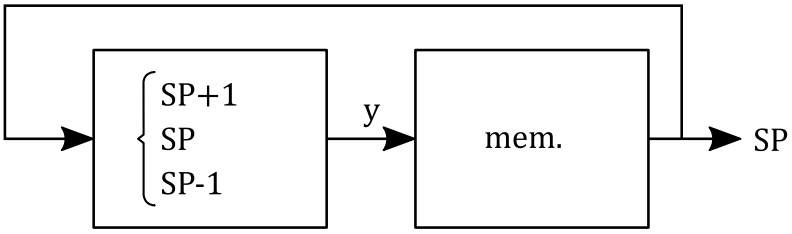
avec les nombres a, b et y représentés sur 16 bits.

kar/manyinputs-10



4.11 Logique pour pointeur de pile

Pour le pointeur de pile (Stack Pointer, SP) d'un microprocesseur, on a besoin d'un bloc capable d'incrémenter le pointeur (mettre une donnée sur la pile), de décrémenter cette valeur (enlever une donnée de la pile) ou de ne pas la modifier (aucune action avec la pile).



Le pointeur de pile est représenté sur 16 bits.

kar/manyinputs-11