

# Architecture des ordinateurs

## Corrigé du TD – Table de Karnaugh et circuits combinatoires

1. Considérer les fonctions logiques suivantes. Pour chacune d'elles,
  - (a) Construire le diagramme de Karnaugh
  - (b) Utiliser le diagramme pour simplifier les fonctions.

$$F_1(A, B, C) = A \cdot \bar{B} \cdot C + A \cdot B \cdot \bar{C} + A \cdot B \cdot C$$

$$F_2(A, B, C, D) = B \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot B \cdot \bar{D} + A \cdot B \cdot C \cdot \bar{D}$$

$$F_3(A, B, C, D) = \bar{A} + A \cdot B + A \cdot \bar{B} \cdot C + A \cdot \bar{B} \cdot C \cdot D$$

$$F_4(A, B, C, D) = \bar{A} \cdot \bar{B} \cdot \bar{D} + \bar{A} \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot B \cdot C \cdot \bar{D} + A \cdot B \cdot D + \bar{B} \cdot \bar{C} \cdot \bar{D} + A \cdot \bar{B} \cdot C \cdot \bar{D}$$

**Correction :** Chaque colonne de la table de Karnaugh doit différer de ses voisines d'un et d'un seul littéral. Les regroupements se font selon les principes suivants :

- (a) faire les plus grands morceaux possibles ;
- (b) faire le moins de morceaux possibles ;
- (c) le nombre de 1 dans un morceau doit être une puissance de 2 ;
- (d) ne faire un nouveau morceau que s'il permet de regrouper des 1 qui n'ont pas encore été regroupés, en se rappelant que la ligne du bas et la ligne du haut sont considérées comme adjacentes, et qu'il en est de même pour la colonne la plus à droite et la colonne la plus à gauche.

$$F_1 = A \cdot B + A \cdot C$$

$$F_2 = B \cdot \bar{D}$$

$$F_3 = B + \bar{A} + C$$

$$F_4 = \bar{A} \cdot \bar{D} + A \cdot B \cdot D + \bar{B} \cdot \bar{D}$$

2. Soit deux nombres  $a$  et  $b$  codés sur 2 bits  $a_1a_0$  et  $b_1b_0$ , dresser les tables de vérité des opérations  $a < b$  et  $a \leq b$ . Après simplification par tables de Karnaugh, donner une expression en sommes de produits de ces opérations.

**Correction :** Table de vérité :

$<$	00	01	10	11
00	0	0	0	0
01	1	0	0	0
10	1	1	0	0
11	1	1	1	0

$\leq$	00	01	10	11
00	1	0	0	0
01	1	1	0	0
10	1	1	1	0
11	1	1	1	1

$$a < b = \bar{a}_1 \cdot b_1 + \bar{a}_1 \cdot \bar{a}_0 \cdot b_0 + \bar{a}_0 \cdot b_1 \cdot b_0$$

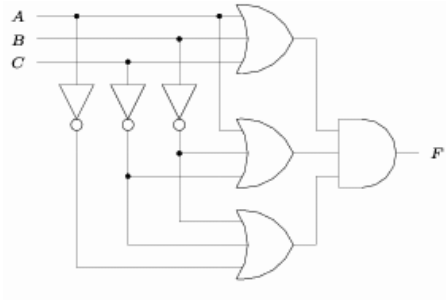
$$a \leq b = \bar{a}_1 \cdot b_1 + \bar{a}_1 \cdot b_0 + \bar{a}_1 \cdot \bar{a}_0 + b_1 \cdot b_0 + \bar{a}_0 \cdot b_1$$

3. Réaliser le circuit logique qui implémente la fonction  $F$ .

$$F = (A + B + C) \cdot (A + \bar{B} + \bar{C}) \cdot (\bar{A} + \bar{B} + \bar{C})$$

(les portes logiques utilisées pourront admettre plusieurs entrées).

**Correction :** Voir figure



4. Un *générateur de parité impaire* est une fonction logique qui retourne 1 si le nombre de ses arguments positionnés est impair et qui retourne 0 si le nombre de ses arguments positionnés est pair. Définir cette fonction pour un mot de quatre bits. Réaliser le circuit logique qui implémente cette fonction.

**Correction :** La table de vérité d'un générateur de parité conduit à une expression qui ne se simplifie pas par une table de Karnaugh. En revanche, en remarquant que pour 2 bits  $A$  et  $B$ ,  $P = A \oplus B$ . On en déduit les circuits suivants :

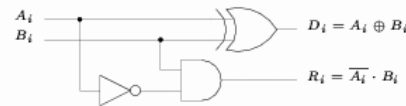


5. Le soustracteur.

- (a) Réaliser un demi-soustracteur.
- (b) Réaliser un soustracteur binaire complet (ou *étage de soustracteur*) selon deux modes :
  - i. avec deux demi-soustracteurs,
  - ii. avec un demi-additionneur et un demi-soustracteur.
- (c) Donner le schéma d'un soustracteur à  $n$  bits

**Correction :** En appelant  $D_i$  la différence et  $R_i$  la retenue, on obtient la table de vérité :

$A_i$	$B_i$	$D_i$	$R_i$
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



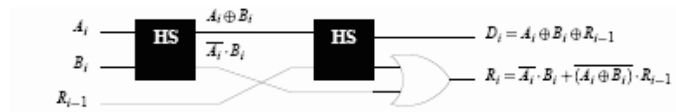
Ainsi,  $D_i = A_i \oplus B_i$  et  $R_i = \overline{A_i} \cdot B_i$ . On obtient donc le circuit suivant :  
En tenant compte de la retenue, la table de vérité devient :

$R_{i-1}$	$A_i$	$B_i$	$D_i$	$R_i$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Donc :

$$\begin{aligned}
 D_i &= \overline{R_{i-1}} \cdot (A_i \oplus B_i) + R_{i-1} \cdot \overline{(A_i \oplus B_i)} \\
 &= (A_i \oplus B_i) \oplus R_{i-1} \\
 R_{i-1} &= \overline{R_{i-1}} \cdot \overline{A_i} \cdot B_i + R_{i-1} \cdot \overline{A_i} \cdot \overline{B_i} + R_{i-1} \cdot \overline{A_i} \cdot B_i + R_{i-1} \cdot A_i \cdot B_i \\
 &= \overline{A_i} \cdot B_i \cdot (\overline{R_{i-1}} + R_{i-1}) + (\overline{A_i} \cdot \overline{B_i} + A_i \cdot B_i) \cdot R_{i-1} \\
 &= \overline{A_i} \cdot B_i + (A_i \oplus B_i) \cdot R_{i-1}
 \end{aligned}$$

D'où le circuit :



Ce schéma correspond à :

- (a)  $A_i - B_i$  (premier demi-soustracteur)
- (b) retrancher  $R_{i-1}$  à la différence obtenue

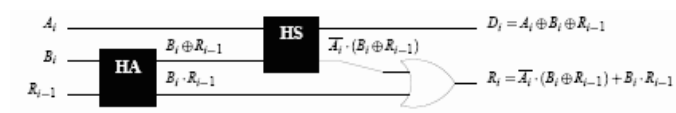
Une autre manière consiste à :

- (a) ajouter  $R_{i-1}$  à  $B_i$  (demi-additionneur)
- (b) puis à retrancher le résultat obtenu de  $A_i$

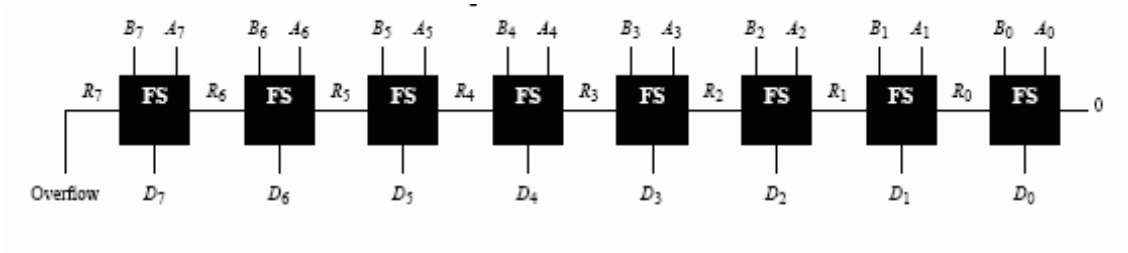
Cela correspond à une autre organisation des expressions booléennes :

$$\begin{aligned}
 D_i &= A_i \oplus (B_i \oplus R_{i-1}) \\
 R_{i-1} &= \overline{A_i} \cdot (B_i \cdot \overline{R_{i-1}} + \overline{B_i} \cdot R_{i-1}) + (\overline{A_i} + A_i) \cdot B_i \cdot R_{i-1} \\
 &= \overline{A_i} \cdot (B_i \oplus R_{i-1}) + B_i \cdot R_{i-1}
 \end{aligned}$$

d'où le circuit :



Ce qui nous amène au soustracteur complet :



6. Un *incrémenteur* est un circuit logique qui ajoute 1 à un entier. L'incrémentation peut être réalisée par un circuit additionneur mais cette opération très fréquente peut donner lieu à un circuit plus simple. Soit un entier non signé de 4 bits  $x_3x_2x_1x_0$  et le résultat  $y_3y_2y_1y_0$  après incrémentation de 1, trouver une expression de  $y_i$  en fonction des  $x_j$  avec  $j \leq i$ . Réaliser le circuit de l'incrémenteur opérant sur des entiers codés sur 4 bits.

**Correction :** Si  $x_0 = 0$ ,  $y_0 = 1$ , sinon  $y_0 = 0$  et on propage. De manière générale,

$$y_0 = \overline{x_0}$$

$$y_i = \overline{x_i} \text{ si } x_{i-1} = x_{i-2} = \dots = x_0 = 1$$

$$= x_i \text{ sinon}$$

$$\text{donc } y_i = x_i \oplus (x_{i-1} \cdot x_{i-2} \cdot \dots \cdot x_0)$$

On en déduit le circuit suivant :

